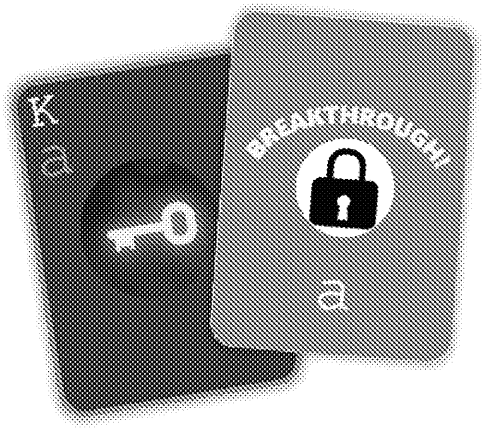
Zig Zag Education

Computer Science

A Level | AQA | 7517

2015 specification for the 2022 exam

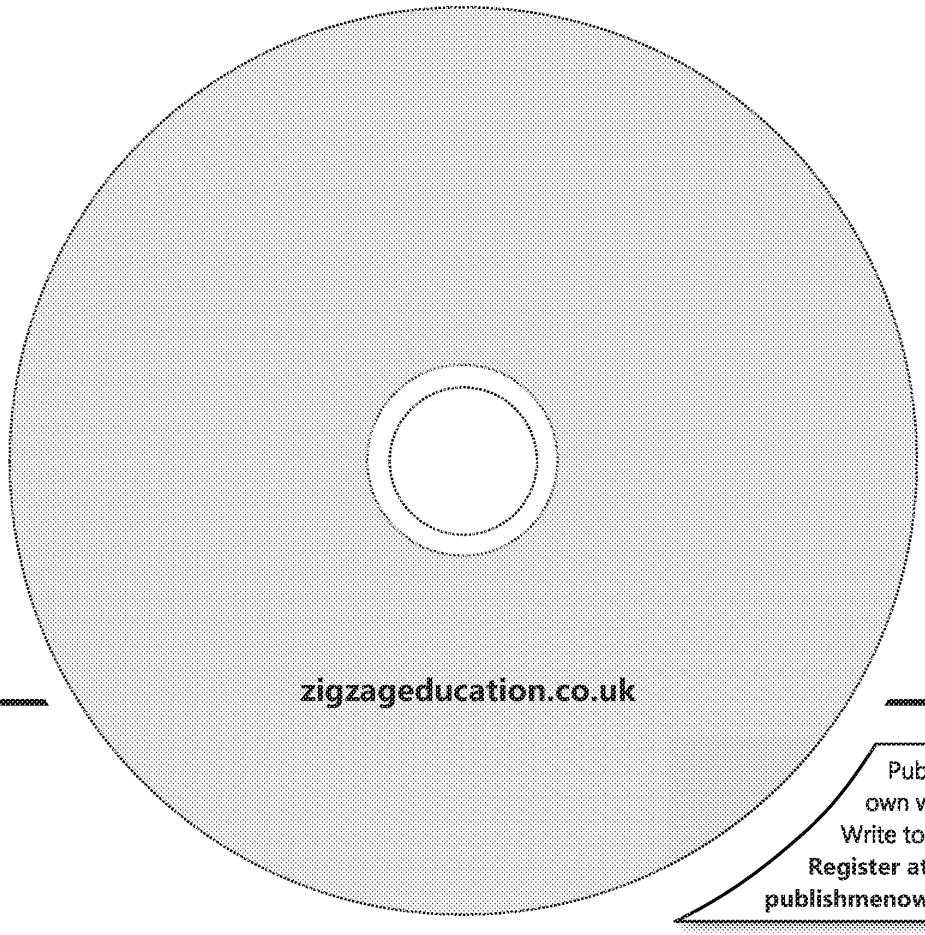


PAPER 1 EXAM RESOURCE PACK 2022

BREAKTHROUGH!

for A Level AQA Computer Science

JAVA EDITION



DH7/
11148

POD
11148

Publish your own work...
Write to a brief...
Register at
publishmenow.co.uk

Contents

Printouts of CD resources (for reference)

- Code Breakdown (10 pages)
- UML Class Diagram – Complete (1 page)*
- Theory Questions: Write-on version (9 pages)
- Theory Questions: Non-write-on version (4 pages)
- Coding Tasks (15 pages)
- Additional Tasks (Extension) (1 page)
- Theory Questions: Mark Scheme (5 pages)
- Programming Tasks: Mark Scheme (34 pages)
- Electronic Answer Document (4 pages)

** Note there are also electronic copies of the UML Diagrams ('Complete' & 'Activity' versions) on the CD – which can be printed in A3, making them much more usable (especially when used as activities)*

Teacher's Introduction

This resource pack is designed to help you support your students taking the **A Level Computer Science Paper 1** exam. It is based on the ***Breakthrough!*** preliminary material (Java) – for examination summer 2022.

On the CD, you will find the following:

`Breakthrough`	this folder contains all of the content (PDF/DOCX) accessible via a HTML interface
`Passwords.txt`	for teacher use – this file contains all of the passwords for the protected PDFs (also listed below)

* PRINTED COPIES OF ALL THE MATERIALS IN THIS DIGITAL RESOURCE PACK ARE INCLUDED FOR REFERENCE.

Installation: Copy the entire `Breakthrough` folder onto a network location that is accessible for students, and provide them with a shortcut to the index.html file. All content can be accessed from this page.

Passwords: All of the PDFs accessible via the *Solutions* web page are password-protected, so that students can only access them with your permission. Each password is a four-digit code, as follows:

The resource pack consists of the following:

① **Code Breakdown**
This document gives a detailed technical overview of the skeleton program, describing in detail each class and method in turn – including their purpose/function, parameters and return values.

> **Note:** although this section is intended to give extra support to teachers and students, it should in no way be seen as a substitute to a student exploring the code for themselves.

② **Class Diagrams**
Three UML Class Diagrams help students explore the skeleton program; there is a completed version, a partially-complete version (gap-fill), as well as a mostly blank template. The completed version is password-protected and accessible via the *Solutions* web page.

③ **Video**
Quick video going over the *Breakthrough!* card game mechanics – intended as a visual aid to accompany the notes in the official AQA preliminary material.

④ **Written Questions**
Theory questions testing students' understanding of the skeleton program. These questions require access to the program, but no modifications need to be made to the program. Write-on (with answer lines) and non-write-on versions are available. Suggested answers are provided via the *Solutions* web page as a password-protected PDF.

⑤ **Coding Tasks**
Fifteen modification exercises put students' programming skills to the test. Example solutions with suggested mark schemes are provided via the *Solutions* web page as a password-protected PDF. Note that these are example solutions and you must use your discretion to award marks accordingly where there are valid alternative solutions.

An **Electronic Answer Document (EAD)** is provided should you wish students to use it for ③ and/or ④ above.

BREAKTHROUGH

Skeleton Code Breakdow

Note: In the skeleton code released by AQA all parameters are

Class: Breakthrough

Identifier / Data		Description
<<constru	Breakthrough	
Parameter	n/a	Initialises several private attributes incl • deck to a new CardCollection • hand to a new CardCollection • sequence to a new CardColle • discard to a new CardCollecti • score to 0 • gameOver to False • locks to an empty list • currentLock to an empty Loc • lockSolved to False Invokes the LoadLocks() method to lo 'locks.txt'.
Return values	n/a	
addDifficultyCardsToDeck (private)		
Parameters	n/a	Add five DifficultyCards to the deck.
Return values	n/a	
checkIfLo	llengeMet (private)	
Parameters	n/a	Iterates through the sequence CardC together the string sequenceAsString the separator between each card desc As a new element from sequence is c sequenceAsString, the string is comp conditions using the checkIfCondition lock to check whether a challenge has incrementally because challenges can challenge has been met, true is return
Return values	Boolean	
checkIfPlayerHasLost (private)		
Parameters	n/a	Checks to s e are any cards le none ar appriate message is displa with the final score; the game is over a If there are cards still left in the deck, t false is returned, allowing the player to
Return values	Boolean	
createSta	Deck (private)	
Parameters	n/a	Used by the setupGame() method to p the correct Files, Picks and Keys for e 5 Picks from toolkits a, b and c are ad Files and 3 Keys from toolkits a, b and
Return values	n/a	

Identifier / Data		Description
getCardChoice (private)		
Parameters	n/a	Used by the playGame() method t their hand they would like to use.
Return values	value : Integer	Contains error handling to catch n not catch a value out of range.
getCardFromDeck (private)		
Parameters	cardChoice : Integer	Used to get the next card from the add it to the hand.
Return values		If the deck CardCollection has a system will then check if the card DifficultyCard. If a DifficultyCard they would like to lose a 'Key' card cards from the deck. The Difficul discard CardCollection and the on the DifficultyCard passing the parameters.
		The system then performs a check repopulating the hand with cards If another Difficulty card is found Difficulty card (or cards if there is the deck) is move automatically in CardCollection rather than into th
		If the deck has run out of cards, the
getChoice (private)		
Parameters	n/a	Used by the playGame() method t like to use a card from their hand CardCollection on the screen.
Return values	choice : String	
getDiscardPlayChoice (private)		
Parameters	n/a	Used by the playGame() method t like to play the selected card from discard the selected card from the CardCollection.
Return values	choice : String	
getRandomLock (private)		
Parameters	n/a	Returns a randomly selected lock locks.
Return values	lock	
loadGame (private)		
Parameters	fileName : String	Uses the fileName parameter to lo imports the current score, challe the hand, sequence, discard and
Return values	Boolean	true is returned if the file is loaded error occurs, an error message is returned.

Identifier / Data		Description
LoadLocks (private)		
Parameters	n/a	Uses a hard-coded 'lo locks available for the file contains the challe 'i' from the file is spl challenges, using a s Each Challenge is the as a delimiter into the The conditions are th variable – lockFromF private attribute locks If an error occurs, an advise that the locks correctly.
Return values	n/a	
moveCard (private)		
Parameters	fromCollection : CardCollection toCollection : CardCollection cardNumber : Integer	Moves a card at the p the CardCollection f CardCollection toCo If the fromCollection toCollection is the s been chosen (i.e. not score is updated appr ed. For all other n another, score is not score is returned.
Return values	score : Integer	
playCardToSequence (p		
Parameters	cChoice : Integer	This method is used t to the sequence to te The system tests to s one card in the CardC system then checks t by the user is a differe played card. If the too can be played and the to the sequence and appropriately for that then gets a new card hand. If the sequence does , the system moves t sequence and the sc The system then uses checkIfLockChallen the new card added t Challenge to be met appropriate message the player score by 5
Return values	n/a	

Identifier / Data		Description
playGame (public)		
Parameters	n/a	This contains the main game loop.
Return values	n/a	Checks to confirm if the private list attribute locks the LoadLocks() method. If none have been load screen and the program ends. If the list does contain locks, it initialises the follow • lockSolved to false • Invokes the setupGame() method to set The main game loop runs while the private attribu There is then an inner loop which runs while gam attribute lockSolved is also false. The inner game loop displays the current user sc current lock and the contents of the hand, and se Using the getChoice() method to display a choic loop then uses selection to either display the disc card in the game. If the user selects to use a card, the system uses to select a card. It then uses the getDiscardOrP if the user wants to play or discard the chosen ca the system moves the selected card from the han CardCollection and gets a new card from the de If the user selects play, the system uses the play move the chosen card from the hand to the sequ Once a card has been played or discarded, the m getLockSolved() method on the currentLock to challenges have been met. If they have, the lock and a new lock is generated. If a lock has been solved, the inner loop returns b which checks if the game is over by invoking the method. If this returns true the game ends.
processLockSolved (private)		
Parameters	n/a	Increments the score by 10 and displays the use
Return values	n/a	Uses an indefinite loop to iterate through the disc all of the cards back to the deck. Reshuffles the deck using the shuffle() method a the getRandomLock() method with the private a

Identifier / Data		Description
setupCardCollectionFromGameFile (private)		
Parameters	lineFromFile : String cardCol : CardCollection	Used for processing lines 4 to file which are for processing t CardCollections (namely the hand a equence). Re ves a single line of text parameter) from the external processes it into a CardColle lineFromFile contains text, it splitLine, using the comma a The splitLine list is then proc card number and card type in CardCollection. If a Difficul instead of a normal ToolCard
Return values	n/a	
setupGame (private)		
Parameters	n/a	Called from the playGame() message of the game on the would like to load in an extern game. If the player chooses t system attempts to load the fi be loaded the game quits. If the player chooses to play a gener s a new deck using me ho and then shuffles it b method. It then moves 5 card start the player off. The syste addDifficultyCardsToDeck() DifficultyCards into the deck ensure they are in random lo assigns a new lock at random currentLock using getRand
Return values	n/a	
setupLock (private)		
Parameters	line1 : String line2 : String	Used for processing lines 2 a file which contain the challen The parameter line1 contains and the parameter line2 cont Each line is split into a string delimiter. The line1 parameter is then the de r to add a new ch sin e may contain multip parameter is split using a sem populate the met status for e setChallengesMet() method
Return values	n/a	

Class: Challenge

Identifier / Data		Description
<<constructor>> Challenge		
Parameters	n/a	Initialises the following p • met to false • conditions to an
Return values	n/a	
getCondition (public)		
Parameters	n/a	Returns a list of strings o challenge in the lock.
Return va	ondition : List (String)	
getMet (pu		
Parameters	n/a	Returns the value of the
Return values	met : Boolean	
setCondition (public)		
Parameters	newCondition : List (String)	Sets the value of the pro condition from the para
Return values	n/a	
SetMet (public)		
Parameters	newValue : Boolean	Sets the value of the pro parameter newValue.
Return values	n/a	

Class: Lock

This class does not have a specific constructor and the

Identifier / Data		Description
addChalleng	(pu	
Paramete	condition : List (String)	Initialises a new challeng condition from the para
Return values	n/a	Appends the new challe protected attribute.
checkIfConditionMet (public)		
Parameters	sequence : String	Returns true and sets th SetMet() if the sequenc challenge, otherwise it re
Return values	Boolean	
convertConditionToString (private)		
Parameters	c : List (String)	Converts list of condition displaying on the screen parameter c, concatenat conditionAsString() usi the delimiter.
Return values	conditionAsString : String	
getChallengeMet (pu		
Paramete	pos : Integer	Returns the met status o pos in the challenges li
Return va	Boolean	

Identifier / Data		Description
getLockDetails (public)		
Parameters	n/a	Used for displaying a challenge's c through the challenges protected together the output string lockDet version of all the challenges for th beer[illegible] not
Return values	lockDetails: String	
getLockSolved (public)		
Parameters	n/a	Returns the status showing if a loc through the challenges protected there are any unmet ones, otherw
Return va[illegible]	[illegible]ean	
getNumberOfChallenges (public)		
Parameters	n/a	Returns the number of Challenge number of challenges in this lock)
Return values	Integer	
setChallengeMet (public)		
Parameters	pos : Integer value : Boolean	Uses the SetMet() method in the C attribute of a challenge at the pos list to met or not met using the va
Return values	n/a	

Class: Card

Identifier / Data		Description
<<constructor>> Card		
Parameters	n/a	Initialises the cardNumber p static attribute (class variab increments the static attribu nextCardNumber which m and updated for all objects c Initialises the score protect
Return values	n/a	
getCardNumber (public)		
Parameters	n/a	Returns the value of the pro
Return values	cardNumber : Integer	
getDescription (public)		
Parameters	n/a	Returns the protected attrib string.
Return values	cardNumber: String	
getScore (public)		
Parameters	n/a	Retur[illegible]e protected attrib
Return values	score : Integer	
process (public)		
Parameters	dec[illegible]ection [illegible]d : CardCollection hand : CardCollection sequence : CardCollection currentLock : Lock choice : String cardChoice : Integer	Base class method for the p classes to override.
Return values	n/a	

Class: ToolCard (inherits from Card)

Identifier / Data		Description
<<constructor>> ToolCard		
Parameters	t : String k : String cardNo : Integer	Initialises the following protected att • toolType from parameter t • k from parameter k • cardNumber from paramet Invokes the setScore() method to a base class for the toolType.
Return values	n/a	
<<constructor>> ToolCard		
Parameters	t : String k : String	Initialises the following protected att • toolType from parameter t • kit from parameter k Initialise cardNumber by calling the Invokes the setScore() method to a base class for the toolType.
Return values	n/a	
getDescription (public)		
Parameters	n/a	Overrides the getDescription() me return a concatenated string of the for this ToolCard
Return values	String	
setScore (public)		
Parameters	n/a	Assigns the correct score from the
Return values	n/a	

Class: DifficultyCard (inherits from Card)

Identifier / Data		Description
<<constructor>> DifficultyCard		
Parameters	n/a	Initialises the protected attribute ca
Return values	n/a	Initialises cardNumber by calling th
<<constructor>> DifficultyCard		
Parameters	cardNo : Integer	Initialises the protected attribute ca
Return values	n/a	Initialises cardNumber from param
getDescription (public) <<override>>		
Parameters	n/a	Overrides the getDescription() me return the protected attribute cardT
Return values	String	

Identifier / Data		Description
process (public) <<override>>		
Parameters	deck : CardCollection discard : CardCollection hand : CardCollection sequence : CardCollection currentLock : Lock choice : String cardChoice : Integer	Overrides the process() me process the user choices fr user receives a difficulty ca like to discard a key or 5 ca Choosing the option to d select a key. This method th parameter is valid. ***Althoug*** ***errors in this check, AQA*** ***code is written as it was i*** If the choice parameter con converted to an index by su a 'key' ToolCard in the play from the hand and placed i If the choice parameter doe through deliberate user cho removed from the deck and CardCollection.
Return values	n/a	

Class: CardCollection

Identifier / Data		Description
<<constructor>> CardCollection		
Parameters	n : String	Initialises the following protected • name from parameter n • cards to an empty list
Return values		
getCardDescriptionAt (public)		
Parameters	x : Integer	Returns a string containing the d index x in the cards list by invok getDescription() method in Car
Return values	String	
getCardNumberAt (public)		
Parameters	x : Integer	Returns the cardNumber attribu the cards list.
Return values	Integer	
getName (public)		
Parameters	n/a	Returns the value of the protecte
Return values	name : String	
addCard (public)		
Parameters	c : Card	Appends the value of parameter cards.
Return values	n/a	

Identifier / Data		Description
createLineOfDashes (private)		
Parameters	size : Integer	Used in formatting a CardCollec
Return values	lineOfDashes : String	Returns an appropriately sized l elements in a CardCollection o CardC tion is greater than t si e)
getCardDisplay (public)		
Parameters	n/a	Used in formatting a CardCollec display output of a CardCollecti the collection name and card de list attribute cards. If there are n collection name and 'empty' is re If there are cards in the collectio which is either appropriately size the collection or is fixed at 10 if t collection is greater than 10. Thi fits correctly in the terminal wind It then uses indefinite iteration to using the getDescription() meth the card at each element and co the | (pipe) symbol to create a vi It then create a second line of d under th 'line of cards' an
Return va	cardDisplay : String	
getNumberOfCards (public)		
Parameters	n/a	Returns the number of cards in t cards.
Return val		
removeC blic)		
Parameters	cardNumber : Integer	Returns the card from cards list removes it from cards. If cardNumber is not a valid ind
Return values	cardtoGet : Card	
shuffle (public)		
Parameters	n/a	Uses definite iteration to perform from one random position to and attribute cards in order to gener
Return values	n/a	

BREAKTHROUGH!

UML class Diagram

Zig Zag Education

BREAKTHROUGH

Theory Questions

These questions refer to the **Preliminary Material** and the **S**
but **do not** requi[illegible] additional programmin

TOTAL MARKS: 80

1 Exami[illegible] private method moveCard. Currently this method return

(a) State a more appropriate name for this local variable.

.....

(b) Currently the moveCard method returns an integer which represe
that was moved. Sometimes this return value is ignored.

Evaluate the choice (of the programmer) to ignore the return valu
return 0 in some cases, and suggest an alternative implementatio

.....

.....

.....

.....

.....

.....

.....

2 The class CardCollection currently contains an interface that expose
structure of a list. For the sequence and the discard pile, a more appr
would be either a queue or a stack.

(a) Justify whether you would use a queue or a stack. When giving y
functionality of the data structure to the behaviour of the game.

.....

.....

.....

.....

.....

.....

.....

(b) In order to implement a stack or a queue for the sequence, justify
thereof) that you would make to the inheritance structure.

....

....

....

....

(c) How would ... a new class to handle a CardCollection that
in... encapsulation?

....

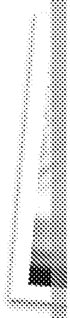
....

....

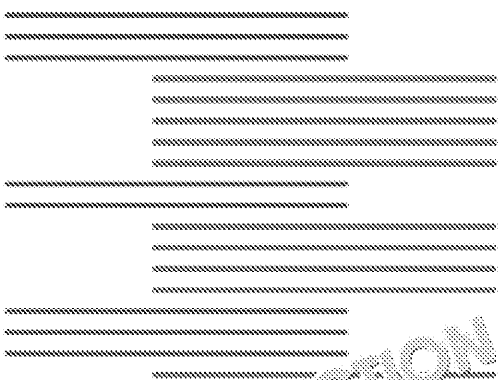
....

3 The shuffle method of the CardCollection class currently swaps 10,0
cards in order to shuffle the deck.

Another way of shuffling the deck is to use a method that humans would normally use called a 'riffle shuffle'. This involves splitting the deck into two approximately even piles and then flicking through each pile from the bottom while combining the halves together into a single deck. Another way of thinking of this would be to imagine pushing the two halves together and having a random number of cards between each card from each half as they recombine.



For example, a deck combined from a blue half and an orange half might look something like this:



Note that in the perfect case a riffle shuffle would use one card from e
is not desired, and in reality, between 0 and 5 cards will normally inter
other half at any time.

a) Write a detailed algorithm for riffle shuffle in any format you choos
pseudocode, flow chart).

b) Explain the space complexity of your algorithm.

.....

.....

4 Examine the checkIfLockChallengeMet method of the Breakthroug
checkIfConditionMet method of the Lock class.

```
Lock:
Challenge Met: P c, F c, K c
Not met: P a, F a, P a
Sequence: P c, F c, K c, P a, F a, P a
```

(a) For the above sequence and lock, complete the trace table below
checkIfLockChallengeMet method of the Breakthrough class.

count	sequenceAsString	Return valu
	""	
5		

(b) If the above lock had a third challenge as below, then how would (please fill it in below)?

```
Not met: F a, P a
```

count	sequenceAsString	Return value
	""	
5		

5 Examine the processLockSolved method in the Breakthrough class methods called by that method.

(a) When a new lock is set, if that lock has been solved before, it will automatically replaced with a new lock the following turn (and trea just solved the first new lock) but reward the player for solving the

.....

.....

.....

.....

(b) D e logical change you would make to the code (no nee al you can) to ensure that this no longer happens.

.....

.....

.....

.....

6 Examine the shuffle method in CardCollection. This method will ma of cards in the deck.

(a) Explain how the effectiveness and eff nc of his algorithm dec cards in the deck reduces.

.....

.....

.....

.....

.....

(b) Other than introducing a riffle shuffle, justify how you could impro efficiency of the algorithm by describing any changes below.

.....

.....

.....

.....

7 ToolC ca be instantiated with either two or three arguments.

(a) Explain what happens in the case where a third argument is supp where only two arguments are supplied.

.....

.....

.....

.....

.....

(b) State the purpose of a constructor.

.....

.....

8 Examin e classes Card, ToolCard and DifficultyCard.

(a) Using evidence from these classes in the program, explain the di abstract and a concrete class.

.....

.....

.....

.....

.....

.....

.....

(b) Using evidence from the Card method, explain the difference bet
(static) and an attribute.

.....

.....

.....

.....

9 Find an example code for each of the following. Only write out th
line/s

(a) Inheritance

.....

.....

(b) Aggregation association

.....

.....

(c) A dynamic data structure

.....

.....

10 This q refers to the concept of polymorphism and how it is use

(a) Choose and then write out one or more lines of the skeleton prog
polymorphism and justify why this is an example of polymorphism

.....

.....

.....

.....

.....

.....

(b) Define the term 'polymorphism'.

.....

.....

.....

.....

11 A suggestion has been made to introduce a new AdvancedLock that
challenge which is only revealed once the basic challenges have been

Explain the steps that you would take in order to do this, i.e. the logical change/addition and the reason for each step.

You are not required to implement this or to write any actual code.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

12 Examine the process method in the DifficultyCard class and the playCardFromDeck method of the Breakthrough class.

Using the sequence below:

```
Not m    P a, F a, K a
Sequence: P a, F a
Hand: P b, K a, F b, K c, P a
```

The player plays the 'K a' card to the sequence and then draws a difficulty card that require them to either discard a key or five cards from the deck. The player chooses to discard the 'K c' from their hand, which is currently in position 4.

Explain what will happen when the process method is called under this scenario, including specific references to the lines of code executed and in which order, and the values of variables, especially choiceAsInteger.

You will need to ensure that you look at the playCardToSequence and process methods in Breakthrough to be certain of the state of the hand and sequence when the DifficultyCard is drawn.

13 The terms 'HAND', 'SEQUENCE', 'DECK' and 'DISCARD' all appear a
and in some cases more than once. This is an example of hard-coded
difficult to maintain and understand and can make it more prone to er

(a) Describe one method of avoiding hard-coding values that makes

(b) Explain why using hard-coded values makes the code more pron
understand.

14 Exception handling is used in several places in the skeleton code; two
the use of file handling.

(a) Describe why it is important to always use exception handling wh

.....

.....

.....

.....

(b) G example of another situation (not file handling) where exc
(it doesn't have to be from the skeleton code) and explain why.

.....

.....

.....

.....

15 This question refers to the playGame method of the Breakthrough c

Explain the use of the private attribute gameOver in this method, spe
is set and why it is used as the condition for iterative statements.

.....

.....

.....

.....

END OF QUESTIONS

BREAKTHROUGH

Theory Questions

These questions refer to the **Preliminary Material** and the **S**
but **do not** requi[illegible] additional programmin

TOTAL MARKS: 80

1 Exami[illegible] private method moveCard. Currently this method return

(a) State a more appropriate name for this local variable.

(b) Currently the moveCard method returns an integer which repres
that was moved. Sometimes this return value is ignored.

Evaluate the choice (of the programmer) to ignore the return valu
return 0 in some cases, and suggest an alternative implementatio

2 The class CardCollection currently contains an interface that expose
structure of a list. For the sequence and the discard pile, a more appr
would be either a queue or a stack.

(a) Justify whether you would use a queue or a [illegible]k. When giving y
functionality of the data structure to the behaviour of the game.

(b) In order to implement a [illegible] queue for the sequence, justify
thereof) that you [illegible]ake to the inheritance structure.

(c) H[illegible]ui[illegible]ating a new class to handle a CardCollection that
im[illegible]encapsulation?

3 The shuffle method of the CardCollection class currently swaps 10,0
cards in order to shuffle the deck.

Another way of shuffling the deck is to use a method that humans would normally use called a 'riffle shuffle'. This involves splitting the deck into two approximately even piles and then flicking through each pile from the bottom while combining the halves together into a single deck. Another way of thinking of this would be to imagine pushing the two halves together and having a random number of cards between each card from each half as they re[illegible]mbine.

For example, a deck combined from a bl[illegible]e h[illegible]and an orange half might look someth[illegible]is:

Note that in the pe[illegible]a riffle shuffle would use one card from e
is not d[illegible]ed, [illegible]reality, between 0 and 5 cards will normally inter
other [illegible]any time.

a) Write a detailed algorithm for riffle shuffle in any format you choos
pseudocode, flow chart).

b) Explain the space complexity of your algorithm.

4 Examine the checkIfLockChallengeMet method of the Breakthroug
checkIfConditionMet method of the Lock class.

```
Lock:
Challenge Met: P c, F c, K c
Not met: P a, F a, P a
Sequence: P c, F c, K c, P a, F a, P a
```

(a) For the above sequence and lo[illegible] complete a trace table like the
checkIfLockChalleng[illegible]od of the Breakthrough class.

count	se[illegible] AsString	Return value
	""	
5		

(b) If the above lock had a third challenge as below, then how would
(Complete an updated trace table)

```
Not met: F a, P a
```

count	sequenceAsString	Return value
	""	
5		

5 Examine the processLockSolved method in the Breakthrough class
methods called by that method.

(a) When a new lock is set, if that lock has been solved before, it will
automatically replaced with a new lock the following turn (and trea
just solved the first new lock) but reward the player for solving the

(b) Describe the logical change you would make to the code (no nee
although you can) to ensure that this no lon[illegible] happens.

6 Examine the shuffle m[illegible]od in CardCollection. This method will ma
of cards in the [illegible]

(a) E[illegible]how the effectiveness and efficiency of this algorithm dec
car[illegible]s in the deck reduces.

(b) Other than introducing a riffle shuffle, justify how you could impro
efficiency of the algorithm by describing any changes below.

7 ToolCards can be instantiated with either two or three arguments.

(a) Explain what happens in the case where a third argument is supp where only two arguments are supplied.

(b) State the purpose of a constructor.

8 Examine the classes Card, ToolCard and D fficultyCard.

(a) Using evidence from these classes in the program, explain the dif abstract and a con ete class.

(b) Usi evidence from the Card method, explain the difference bet (s nd an attribute.

9 Find an example in the code for each of the following. Only write out th line/s of code.

(a) Inheritance

(b) Aggregation association

(c) A dynamic data structure

10 This question refers to the concept of polymorphism and how it is use

(a) Choose and then write out one or more lines of the skeleton prog polymorphism and justify why this is an exam of polymorphism

(b) Define the term 'polymorphism'.

11 A suggestion has been made to introduce a new AdvancedLock that challe hich is only revealed once the basic challenges have been

Explain steps that you would take in order to do this, i.e. the logica change/addition and the reason for each step.

You are not required to implement this or to write any actual code.

12 Examine the process method in the DifficultyCard class and the pla getCardFromDeck methods of the Breakthrough class.

Using the scenario below:

```
Not met: P a, F a, K a
Sequence: P a, F a
Hand: P b, K a, F b, K c, P a
```

The player plays the 'K a' card to the sequence and then draws a diffi require them to either disc rd or five cards from the deck. The p discard the 'K c' fro hand, which is currently in position 4.

Explai t will happen when the process method is called under th includi ecific references to the lines of code executed and in whic values of variables, especially choiceAsInteger.

You will need to ensure that you look at the playCardToSequence an methods in Breakthrough to be certain of the state of the hand and s the DifficultyCard is drawn.

13 The terms 'HAND', 'SEQUENCE', 'DECK' and 'DISCARD' all appear a
and in some cases more than once. This is an example of hard-code
difficult to maintain and understand and also make it more prone to er

(a) Describe one method of avoiding hard-coding values that makes

(b) Explain why using hard-coded values makes the code more pron
understand.

14 Exception handling is us in several places in the skeleton code; two
the use of file ha

(a) D e why it is important to always use exception handling wh

(b) Give an example of another situation (not file handling) where ex
(it doesn't have to be from the skeleton code) and explain why.

15 This question refers to the playGame method of the Breakthrough c

Explain the use of the private attribute gameOver in this method, spe
is set and why it is used as the condition for two iterative statements.

END OF QUESTIONS

BREAKTHROUGH

Programming Tasks

These questions require you to load the **Skeleton Program** and to make

Note that any alternative or additional changes that you deemed appropriate
– ensuring that it is clear elsewhere in the Skeleton Program those chang

Task 1

This question refers to the playGame method of the Breakthrough class.

The number of cards left in the deck should be printed out after the curren
cards in the player's hand each turn.

Test the changes you have made:

Run the game and play two turns, showing the number of cards in the deck

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playGa
- SCREEN CAPTURE(S) showing the required test

Task 2

This question refers to the playGame and getChoice methods of the Bre
creation of a new attribute (with accessor methods), peekUsed in the Loc

Introduce a **(P)eek** option. This can be used once per lock, and allows a p
see the next three upcoming cards. There should be a new command in p
the 'deck peek' is still available.

Create a new attribute in the [illegible] class called peekUsed. Create access
to update and read the peekUsed attribute (get/set).

Update the getChoice() method in the Breakthrough class to give the use
menu option should only appear if the peekUsed attribute is false.

Introduce an option to the menu in the playGame() method to accept 'P' a
This menu option should only appear if the peekUsed attribute is false. D
the deck using the getCardDescriptionAt() method. Set the peekUsed a
peek option has been chosen by the user.

When the player is given a new lock, set the peekUsed attribute appropria
the peek option again.

Test the changes you have made:

Run the game and peek (peek is an option, it works) and then it's no longe
make sure it doesn't work even though the option isn't displayed. Solve a l
now an option again.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playGa
- PROGRAM SOURCE CODE showing changes made to the getCho
- PROGRAM SOURCE CODE for the new peekUsed attribute
- SCREEN CAPTURE(S) showing the required test.

Task 3

This question refers to the playCardToSequence method of the Breakth

Under the rules of the game, a player cannot play two cards of the same ty
there is no error message warning the player when they attempt to do this

Modify the playCardToSequence method in the Breakthrough class to in
which tells the user that they cannot play two cards of the same type sequ

Use the getCardDescription method to highlight to the user which card
and explain why it is the same as the type just played.

Test the changes you have made:

Run the game and show at least one turn played where the error does not
shows the new error message under the correct conditions of playing a du
show that (1) the error message is displayed and (2) the card is not played

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playCar
- SCREEN CAPTURE(S) showing the required t

Task 4

This question refers to the playGame and getChoice methods and the cr attribute, mulliganUsed of the Breakthrough class.

Each player gets 1 'mulligan' per game where they can take all the cards i discard pile and the sequence, put them together a sh affle up and deal drawn (when repopulating the player's h d) should be sent to the discard the current lock including any solv challenges will remain unchanged.

Create a ne attri the Breakthrough class called mulliganUsed w mulliganU false then display an additional **(M)ulligan** option each t mulligan has been used, set the mulliganUsed attribute to true, which sh option is no longer displayed or usable.

Test the changes you have made:

Run the game, solve one challenge, use mulligan, play one card to the se attempt to mulligan again despite no menu option).

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing chan de to the playGa
- PROGRAM SOURCE CODE sh changes made to the Breakt
- PROGRAM SOU E showing changes made to the getCho
- SCR CA URE(S) showing the required test

Task 5

This question refers to the playGame and getChoice methods of the Bre

The player will have a new option in playGame to **(Q)uit**, and for this they score for each card remaining in the deck. Print out their final score as the

Note that the code should exit cleanly/nicely without using any System.ex break/continue are allowed of course.

Test the changes you have made:

Play one turn of a game, choose quit.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playGa
- PROGRAM SOURCE CODE showing changes made to the getCho
- SCREEN CAPTURE(S) showing the required test

Task 6 Diffi

This question refers to the getCardFromDeck method of the Breakthrou
new method, displayStats, modifying two existing methods, addCard and
adding three new attributes, numPicks, numFiles and numKeys, in the C

Introduce a stats / card count to the CardCollection class which keeps tra
out of the deck and calculates the % chance that the next card tile in the d

Introduce three new attributes to the CardCollection class called numPic
which will be updated every time a ToolCard is added to or removed from

Create a new method in the CardCollection class called displayStats. Th
the percentage chance of the next card being a key, pick or file based on t
the number of cards left in the deck.

When the player receives a difficulty card, use the displayStats method to
getNumberOfCards method in the CardCollection class to display the fo
they choose 'lose a key or discard 5 cards from the deck'.

*There is a X% chance that the next card will be a key, a Y% chance that it
that it will be a pick.*

The percentages should be displayed to two decimal places.

Replace X, Y and Z with the appropriate values. Note that they will not nor
because there are also difficulty cards in the deck.

Test the changes you have made:

Run the game until a difficulty card is drawn and show the printout of the s
(after the hand and before asking which card).

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the getCar
- PROGRAM SOURCE CODE showing changes made to the CardCo
- SCREEN CAPTURE(S) showing the required test

Task 7 Diffi

This question involves the createStandardDeck, processLockSolved an methods of the Breakthrough class, as well as the creation of a new setC Card, ToolCard and CardCollection classes.

Introduce three new 'multi-tool' cards – a **multi-[illegible]**), a **multi-key (K**

At the start of a standard game (not [illegible]ading a save game file), the d of these new types of card[illegible]ti-tool cards can be dealt to the player's ha cards are.

On playing [illegible]lti-tool card, the player should be given the option to choos assign the card to before it is added to the sequence, therefore allowing a any lock challenge of that type.

When a lock has been solved, three new multi-tool cards (one of each type available for the next lock and the deck is reshuffled (as normal).

Test the changes you have made:

Play the game and show the use of at least one multi-tool card, the print s sequence both before and after the multi-tool is played.

Evidence that you need to provide:

- PROGRAM SOURCE CODE sh[illegible] changes made to the createS
- PROGRAM SOU[illegible] showing changes made to the process
- PRO[illegible] SOURCE CODE showing changes made to the playCar
- PROGRAM SOURCE CODE for the new setCardToolkit method (in CardCollection classes)
- SCREEN CAPTURE(S) showing the required test.

Task 8 Diffi

This question refers to the getLockDetails method of the Lock class and Breakthrough class.

Challenges are to be marked as 'partially met' (rather than just 'met' or 'no solved. A challenge is partially met if the end of the sequence (last one or an unsolved challenge.

Modify the call to getL... from playGame to pass in the sequenc

Modify get... etails so that if the challenge is not met then it checks to For challenges of three cards, only check the last two cards and it become of the sequence matches the first card of the challenge or the second last the first card of the challenge and the last card of the sequence matches t challenge.

In general, check N-1 cards where N is the number of cards in the challen challenges of one card cannot be partially met. You only need to solve the three cards exactly.

Test the changes you have made:

Run the game and play one card to the sequence that doesn't match any one towards one of the three card challenges that matches the first card fo screen showing this entire turn.

Then play a second card to sequence that matches the second card o

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playGa
- PROGRAM SOURCE CODE showing changes made to the getLoc
- SCREEN CAPTURE(S) showing the required test

Task 9 Diffi

This question refers to the playGame method of the Breakthrough class.

Introduce a bonus for solving locks using fewer cards. Once the first card is sequence for a new lock, a counter starts and one is added every time a p (discarding or playing to the sequence).

Once a lock is solved (all the chambers), a player receives an extra point the counter, after which it is reset. The player simply receives 0 if the coun message confirming the bonus points that were awarded (including 0 if tha

Test the changes you have made:

Run the game and play two locks, one solved in under 20 cards to show a in over 20 cards to show a bonus score of 0.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playGa
- SCREEN CAPTURE(S) showing the required test

Task 10 Diffi

This question refers to the processLockSolved, setupGame and getCar
as the creation of a new method, addGeniusCardToDeck of the Breakth
of a new class called GeniusCard.

Introduce a new 'Genius Card' which is added to de at the start of a lock
chance of having a 'Genius Card' in a de

A player can choose to use the 'Genius Card' when they draw it to solve a
ask which challenge it will be discarded and then reshuffled into the de
from the di pile.

Note that if a GeniusCard is drawn when filling up the hand it should be d
message should be printed to this effect.

Create a method called addGeniusCardToDeck which has a 25% chance
to the deck. This should be called from processLockSolved and setupG

Create a new class for the GeniusCard which inherits Card with cardTyp
the getCardFromDeck method of Breakthough to ensure that the card is
drawn.

Test the changes you have made:

Run the game and play until a 'Genius Card' is drawn, then choose yes an
challenge in the current lo

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the proces
- PROGRAM SOURCE CODE showing changes made to the setupG
- PROGRAM SOURCE CODE showing changes made to the getCar
- PROGRAM SOURCE CODE for the new GeniusCard class
- PROGRAM SOURCE CODE for the new addGeniusCardToDeck m
- SCREEN CAPTURE(S) showing the required tes

Task 11 Di

This question refers to the addition of a new attribute in the Breakthrough c getCardFromDeck method of the Breakthrough class as well as the creati printToolsAvailable, for the CardCollection class.

Introduce the concept of 'Buying a tool' from the dec

Add a new attribute, credits, to the Br ough class which contains the n currently has. At the start of the gan e player has 10 credits. When a play sequence or discarded ey have at least 2 credits remaining, they s like to buy (y) ore their hand is refilled from the deck. If they choos card as no ut otherwise the new card will be the tool card that they purc

Players can 'buy' a 'Key' card at the cost of 3 credits, and 'file' or 'pick' cards a When the player chooses 'y' to buy a tool, they should be prompted with the f have 0 availability should not be listed).

```
1.   F a (1 available)
2.   F b (1 available)
3.   F c (1 available)
4.   P a (1 available)
5.   P b (1 available)
6.   P c (1 available)
7.   K a (1 available)
8.   K b (1 available)
9.   K c (1 available)
10.  No Tool (buy nothing)
```

***Note**: the actual number a*

***Note**: keys (items 7–9) sho at least 3 credits left. All m numbers given above even e.g. item 10 should always the player changed their m*

The new printToolsAvailable meth s and take one parameter, keysAva player has at least 3 credits e otherwise is false. It should return an array c available tool card of each type, for example, if the deck contains three files fr at index 3 deck, no files from toolkit b and one file from toolkit c which is the first par e array returned would look like this:

```
[ 3, -1, 12, ...
```

***Note:** -1 is used to indicate*

Test the changes you have made:

1. Run the game and play any card to the sequence, then choose 'y' whe a tool. Select any tool listed as available, play it to the sequence and t asked if you would like to buy a tool; show all the output produced incl and the tool card being added to the player's hand each time.
2. Continue playing the game and buying tools until you have spent a tota pick/file and 2 keys) and then show the printed li tools available whe

Evidence that you need to pr

- PROGRAM SOU CODE showing changes made to the getCardFr
- PRO SOURCE CODE for the new credits attribute
- PROGRAM SOURCE CODE for the new printToolsAvailable method
- SCREEN CAPTURE(S) showing the required test

Task 12

This question refers to checkIfLockChallengeMet method of the Breakth

Create an 'Advanced' mode where, for any challenge that requires three c once the challenge is solved move the cards used to solve it from the seq exposing the previous card on the sequence, which could then possibly be challenge.

For example, if the sequence contains:

```
Fa, Kc,
```

and the current challenge is Pb, Kb, Fb. Suppose you play Kb and Fb to th current challenge but instead of the sequence extending to:

```
Fa, Kc, Pb, Kb, Fb
```

it will be contracted to:

```
Fa, Kc
```

and the Pb, Kb and Fb cards from the challenge that was just solved will b

Test the changes you have made:

Run the game and restart until you get a Lock of at least one challenge of three cards. Play until you solve the three card challenge and then play challenge. The screen capture(s) should show the Lock, Sequence and Ha card to solve the three card challenge and the Lock and Sequence after yo

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the checkIfLo
- SCREEN CAPTURE(S) showing the required test

Task 13

This question refers to playGame and getChoice methods and to the cre
method in the Breakthrough class. It also requires the creation of new ge
getChallengesMetAsString methods in the Lock class.

The playGame menu should have a **(S)ave** optic ici will save the gam
it to be reloaded (from the main menu y first start the game).

In order to understand th at the save game file, you will have to in
the loadGa me in the Breakthrough class.

Print out a le message stating whether the game was saved succes

Test the changes you have made:

1. Take a copy of the **game1.txt** file and rename it **backup.txt**.
2. Run the game until you get a lock with at least two challenges. Sol
save the game as '**game1.txt**' (it shouldn't prompt you). Load the g
of it has been correctly restored.
3. Restore the original **game1.txt** from **backup.txt**.

Evidence that you need to provid

- PROGRAM SOURCE CODE showing changes made to the playGa
- PRO M SOURCE CODE showing changes made to the getCho
- PRO M SOURCE CODE for the new saveGame method
- PROGRAM SOURCE CODE for the new getChallengesAsString m
- PROGRAM SOURCE CODE for the new getChallengesMetAsStri
- SCREEN CAPTURE(S) showing the required test

Task 14

This question refers to playGame and playCardToSequence methods an attribute, bonusPool, in the Breakthrough class. It also requires the crea isPartial, in the Lock class, which takes sequence as a parameter.

Introduce a bonus for playing consecutive cards towards that solves a cha card played in a row that goes towards solving a challenge will add 5 to th pool will be added to the score for the card.

For example, the bonus pool is 0 and a player plays a card towards challe added to the score along with their normal score and the bonus pool is in does anything except play another correct card towards challenge 1, then 0; otherwise they will get the score for the card played as normal, plus the bonus pool will be increased to 10 and so on.

Test the changes you have made:

Run the game and keep discarding until you have all three cards required solve it one card after another; continue playing and play a card to a challe sequence that is not part of the challenge.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the playGa
- PROGRAM SOURCE CODE showing changes made to the playCa
- PROGRAM SOURCE CODE for the new bonusPool attribute
- PROGRAM SOURCE CODE for the new isPartial method
- SCREEN CAPTURE(S) showing the required test

Task 15 Difficu

This question refers to the processLockSolved, getCardFromDeck and
methods, and to the creation of new private generateSolubleLock and g
and a new private attribute finalLock in the Breakthrough class. It also r
public isSoluble method in the Lock class that take
deck and hand

EXTRA FILE NEEDED: **game2.txt**

Every lock presented mu
soluble based on the cards left in the deck
exhaust the deck. If the lock cannot be solved, then choose a new random
in a row (w
a suitable lock being found) then display a message 'Fina
generate a lock with two challenges that can be solved.

Once those challenges are solved, there should be a message from chec
of saying the player lost, prints out 'You have solved the final lock. Your fi

When approaching this task you should ignore the effect of Difficulty card
check that the deck and hand combined contain the requisite number of e
lock.

The attribute finalLock should be set to 0 at the start and then set to 1 in
final lock is set. When checkIfPlayerHasLost runs, it should set finalLoc
the final turn is played). If finalLock is 1 and there are no cards left in the
lose until all the cards from their hand are gone.

Test the changes you have ma

1. Change the
oad the file **game2.txt** instead of **game1.txt**
loa
e.
2. Play the game until the message 'Final Lock' is displayed, then sol
final turn.

Evidence that you need to provide:

- PROGRAM SOURCE CODE showing changes made to the processL
- PROGRAM SOURCE CODE showing changes made to the checkIfP
- PROGRAM SOURCE CODE showing changes made to the getCardF
- PROGRAM SOURCE CODE for the new gener
lubleLock meth
- PROGRAM SOURCE CODE for the
erateChallenge method
- PROGRAM SOURCE CODE for the new IsSoluable method
- PROG
CODE for the new finalLock attribute
- SCREE
APTURE(S) showing the required test

BREAKTHROUGH

Possible Additional Programmin

1. Create an extra toolkit (e.g. 'd') and ad[illegible] involving this to the l
2. Introduce a *Swiss Army K[illegible]* ca[illegible] which can be used as any singl toolkit.
3. Add [illegible]ful [illegible]ality to allow a player, at the start of a turn, to (U)n rev[illegible]rything – including the score, sequence, hand and discar the start of the previous turn. There should be one undo available p possible to use it on the first turn of a new lock.
4. Add a *High Scores* file and ability to view this from a main menu.
5. Add levels so that different locks have different challenges which w depending on the current level. This could be linked to (11 – comp number of toolkits used, e.g. 2, 3 or 4.
6. Add a *Mighty Hammer* card that can smash (solve) the current lock your hand and play it later.
7. Introduce a user-defined locks option. This generates a rough pseu one player can choose a lock sequence and another has to try to u original game of *Mastermind*). A user-defined lock must follow the c at least two must be files, and at least one must be a pick.
8. Introduce a second type of lock – 'Maths Lock', whereby the play the way they are now and [illegible] new maths locks. This will involve each card call[illegible]'number'. The value of 'number' is displayed in a (pic[illegible]e[illegible]y currently displayed. Cards can be used for their ma the[illegible]attribute. For example, if a lock contains four files – each that gives a total lock value of 20. The player needs to play a sequ a total value of 20. For example, if the player plays two picks, each then the lock will open. These new 'Maths Locks' are solved only u are independent of the tool type and tool kit.
9. Receive a bonus of 50 if you quit and the current challenge could n and deck as they are currently).
10. Add an *Autoplay* mode which shows a computer simulation of the g
11. Design a formula to compute a complexity value for a lock.
12. Validation of card to play (with exception handling) for choosing wh response to a difficulty card.
13. Validation on entry of choice (or an[illegible]e[illegible]tr[illegible]) so that the player can c
14. Be able to sacrifice a card [illegible]moved from the game) in order to cha
15. Exami[illegible]e th[illegible]1.txt file (or game) closely and draw a flow diagr pla[illegible]ul[illegible]make to move from the 'saved state' to 'end of the gam look[illegible]at the data in the game1.txt file rather than playing the gam

BREAKTHROUGH!

Theory Questions (Mark Scheme)

Question		Suggested Solution	Total Marks	Marking Guidance
1	(a)	e.g. additionalScore // cardScore	1 mark	**A:** Similar names with meaning to explain the score. **R:** Spaces in names. **I:** case.
	(b)	**4 marks:** Ignoring a return value is not good practice **[1]**… One alternative would be to create a new method called *moveCardWithScore* **[1]**… which takes the current player *score* as a parameter and returns the updated *score* **[1]**… and to remove the return value from the current method **[1]**. **Examples of answers worth less than full marks:** **3 marks:** make the return type void **[1]**… and move the logic to the place where the card is played to the sequence **[1]**… which is the only time the score is needed **[1]**. **2 marks:** remove the scoring **[1]**… and create a separate *getScore()* method **[1]**. **1 mark:** always check the return value as ignoring it is bad practice **[1]**.	4 marks	**A:** any reasonable suggestion. **A:** answers without passing *score* as a parameter and dealing with the extra score as per now. **A:** answers where there is a scoring method in *CardCollection* which 'knows' whether to score or not. **A:** passing score in by reference and having a new attribute on *CardCollection* to indicate if a card added/played should affect the score.
2	(a)	The sequence only allows cards to be added to the end and taken from the same end which is a LIFO structure **[1]**… and a stack is a LIFO structure that would be appropriate **[1]**… For the discard pile, sometimes you need to peek at the whole stack but generally just play cards to the top and then the whole pile is shuffled back in **[1]**… a stack could be suitable for this	4 marks	1 mark for each point (MAX. 4) **A:** stack for discard pile. **R:** queue for either.

<table>
<tr><th>Question</th><th colspan="2">Suggested Solution</th><th>Total Marks</th><th>Marking Guidance</th></tr>
<tr><td rowspan="2">3</td><td>(a)</td><td>Splitting the deck into two halves (or using pointers to do the same thing) [1]… Choosing a number of cards from one half and add them to the combined deck (A: circular deck with counting solutions) [1]… Choosing a number of cards from the other half and add them to the combined deck [1]… Using a random number of cards (e.g. 1 to 5, A. 1 to 5) [1]… Taking cards from the bottom of the split decks rather than the top. Repeating until the deck is fully combined [1].</td><td>6 marks</td><td>1 mark for each point</td></tr>
<tr><td>(b)</td><td>Most algorithms will have a space complexity twice that of the random shuffle that existed before, by storing the deck split and a new merged/combined deck.</td><td>1 mark</td><td>A: any version of the idea for 1 mark.
A: circular (e.g. solutions with space complexity can be same as the storage for the deck as long as they are explained properly.</td></tr>
<tr><td>4</td><td>(a)</td><td><table><tr><th>count</th><th>sequenceAsString</th><th>Return value</th></tr><tr><td></td><td>“”</td><td></td></tr><tr><td>5</td><td>“P a”</td><td></td></tr><tr><td>4</td><td>“, P a”</td><td></td></tr><tr><td></td><td>“F a, F a”</td><td></td></tr><tr><td>3</td><td>“, F a, F a”</td><td></td></tr><tr><td></td><td>“P a, F a, n”</td><td>True</td></tr></table></td><td>5 marks</td><td>1 mark for the count column (I: spaces)
1 mark for a final return value of True
1 mark for the first value in the sequenceAsString column
1 mark for the last value in the sequenceAsString column
1 mark for the correct middle values in the sequenceAsString column (between the first and last value)
DPT: -1 only for a missing space</td></tr>
<tr><td></td><td>(b)</td><td><table><tr><th>count</th><th>sequenceAsString</th><th>Return value</th></tr><tr><td></td><td>“”</td><td></td></tr><tr><td>5</td><td>“P a”</td><td></td></tr><tr><td>4</td><td>“, P a”</td><td></td></tr></table></td><td>3 marks</td><td>1 mark for each column
DPT: -1 only for a missing space (note that this is across parts (a) and (b) combined, total of -1 for a missing space across the two</td></tr>
</table>

Question		Suggested Solution	Total Marks	Marking Guidance
5	(b)	**Either:** Once a lock is solved, remove it from the list of locks so that it cannot be selected again when a new random lock is chosen. **Or:** Once a lock is solved, iterate through all the challenges and set the attribute *met* for each to *false*.	2 marks	2 marks available for either solution as long as the details are clearly explained, otherwise award 1 mark **A:** any reasonable solution including the idea of moving *lockSolved* into the *Lock* class and checking it when choosing a new lock.
6	(a)	As the number of cards in the deck gets smaller then there will be more swaps than possible arrangements of the cards, which makes the additional swaps redundant and inefficient **[1]** …For example, with 2 cards there are only two combinations and with 6 only 720 but there are still 10 000 swaps **[1]** …There is also the chance of the algorithm causing an exception with 0 or 1 cards but there is no check for this **[1]**.	3 marks	Award 1 mark for each point **A:** any expression of the idea that 10000 is more swaps than you need, which makes the extra ones unnecessary for the first mark. **A:** any reference or example to decreasing combinations for the second mark.
	(b)	The number of swaps could be a measure of deck size or could be set to a lower threshold when the deck is small **[1]**. This would be done by using a selection statement and setting a swaps variable to a lower value if the deck is short or to 10000 as now if the deck is large **[1]**.	2 marks	Award 1 mark for each point **A:** any expression of each concept for each mark. **A:** any other reasonable suggestions that would give 10000 for large decks and a much lower number for small decks.
7	(a)	When two arguments are supplied they are used to set the *toolType* and *kit* respectively **[1]**… The *cardNumber* is set using the next available *cardNumber* from the class/static variable *nextCardNumber* by the parent constructor when it is called **[1]**… In the case of a third argument being supplied, the parent constructor is not called and the *cardNumber* is set by the value of the parameter **[1]**.	3 marks	1 mark for each point (MAX. 3)

Question		Suggested Solution	Total Marks	Marking Guidance
8	(b)	A class or static variable has the same value for every object and is changed in them all when it changes in one **[1]**… An attribute may start off the same but has a different value in each object and if changed in one will not affect the others **[1]**.	2 marks	1 mark for each point
9	(a)	Inheritance is where a child gains the attributes and behaviours/methods of its parent.	1 mark	**A:** other words with similar meanings.
	(b)	Aggregation associate is where one class contains another class but their lifespans are not linked/they can function independently.	1 mark	**R:** answers that do not refer to lifespan in some way.
	(c)	A data structure for which the memory usage will shrink and grow over time according to the storage needs.	1 mark	

10	(a)	***Solution #1***
		Code **[1]:**
		`moveCard(hand, discard, hand.getCardNumberAt(cardChoice - 1));`
		… because CardCollection treats DifficultyCards and ToolCards **[1]** as Cards **[1]**… which is an example of polymorphism but when the statement resolves any methods will actually execute on the child's object class **[1]**.
		
		Solution #2
		Code **[1]:**
		`@Override`
		`public void process(CardCollection deck, CardCollection discard, CardCollection hand, CardCollection sequence, Lock currentLock, String choice, int cardChoice)`
		… because this overrides the process method **[1]**… in the parent Card class **[1]**… which will mean that DifficultyCards can be treated as cards but behave as themselves **[1]**
			4 marks	For any one possible answer: 1 mark for each point and 1 mark for the code
				A: any example of code related to inheritance for 1 mark provided the explanation gains at least 1 mark.
				R: code only with no explanation.

Question		Suggested Solution	Total Marks	Marking Guidance
11		Create a new *AdvancedLock* class **[1]**… that inherits from *Lock* **[1]**… and override the *getLockSolved* method **[1]**… so that when the basic challenges are solved it unlocks the final challenge and returns *false* instead of *true* **[1]**… this will then mean that it can refer to a pr… ttribute such as *secretUnlocked* **[1]**… when che… getLockSolved the next ti…	6 marks	1 mark for each point
12		The 'K a' card is in position 2 meaning that when *getCardFromDeck* is called, *cardChoice* will be … **[1]**… The card in position 2 will have already been m…d from the hand to the deck as *mo…rd* is called by *playCardToSequence* before *getC…FromDeck* **[1]**… The hand is redisp…d the player enters 3 to choose 'K c' as the key to c…rd because it is now in position 3 … When *process* is called *cardChoice* is 2 and *ch…e* is 3 **[1]**… *choice* is converted … a string to an integer, range checked (1–5) succes…ly and stored in *choiceAsInteger* **[1]**… as 3 >= 2, *choiceAsInteger* is decremented … 1 **[1]**… and as *choiceAsInteger* is still > 0 it is decremented by 1 again to give 1 **[1]**… The selection statement ch…ck against the card in the hand at index *choiceAsInteger* (which is 1) and that card is now … b' so the condition is *false* **[1]**… and 5 cards are di…ded from the deck to the discard p… **[1]**.	8 marks	1 mark for e…ch point (MAX. 8)
13	(a)	Using a constant **[1]** … which would be declared once at the top of prog…m and could be changed in that sin…place **[1]**.	2 marks	1 mark for each poi…
	(b)	It is possible to mis…t values **[1]** ,… update the wrong values **[1]**,… or m…e values **[1]** … constants have an …entifier, making a value easier to understand **[1]**.	2 marks	1 mark for each point (MAX. 2) **A:** opposite points.
14	(a)	File handling can always generate exceptions **[1]**… because files could be locked **[1]**… removed/unavailable/inaccessible **[1]**	2 marks	1 mark for each point (MAX. 2)
	(b)	Converting an inputted string to an integer **[1]**… because if it fails you want to catch the error and ask the user to input again **[1]**.	2 marks	1 mark for each point **A:** any example of validation.

BREAKTHROUGH

Programming Tasks (Mark Sch

Task 1

Coding

- Printing out the number of cards left in the deck correctly each turn [1 mark]

Example Solution

```
// code added
Console.writeLine("There are " + deck.getNumberOfCards() + " c
// end addition
Console.writeLine(hand.getCardDisplay());
```

Testing:

- Printing out the number of cards left correctly between SEQUENCE and HAND [1

```
SEQUENCE: empty

There are 33 cards left in the deck

HAND:
```

Task 2

Coding:

- Changing `getChoice` to show Peek (even if it doesn't check `getPeekUsed`) [1 ma
- Changing `playGame` to accept 'P' and printing out the three cards in the deck (reg [1 mark]
- Adding the `peekUsed` attribute with get's and sets methods to `Lock` [1 mark]

Example Solution

Changes to ge ce

```
private String getChoice() {
      Console.writeLine();
      // code change
      if (currentLock.getPeekUsed()) {
            Console.write("(D)iscard inspect, (U)se card:> ");
      } else {
            Console.write("(D)iscard inspect, (U)se card, (P)e
      }
      // end change
      String choice = Console.readLine().toUpperCase();
```

Changes to `playGame`

```
      Console.writeLine(discard.getCardDisplay());
      break;
// code added
case "P":
      if (!currentLock.getPeekUsed()) {
            Console.writeLine("The next three cards are: " + de
            "  deck.getCardDescriptionAt(1) + ", " + deck
            entLock.setPeekUsed();
      }
      break;
// end addition
case "U":
```

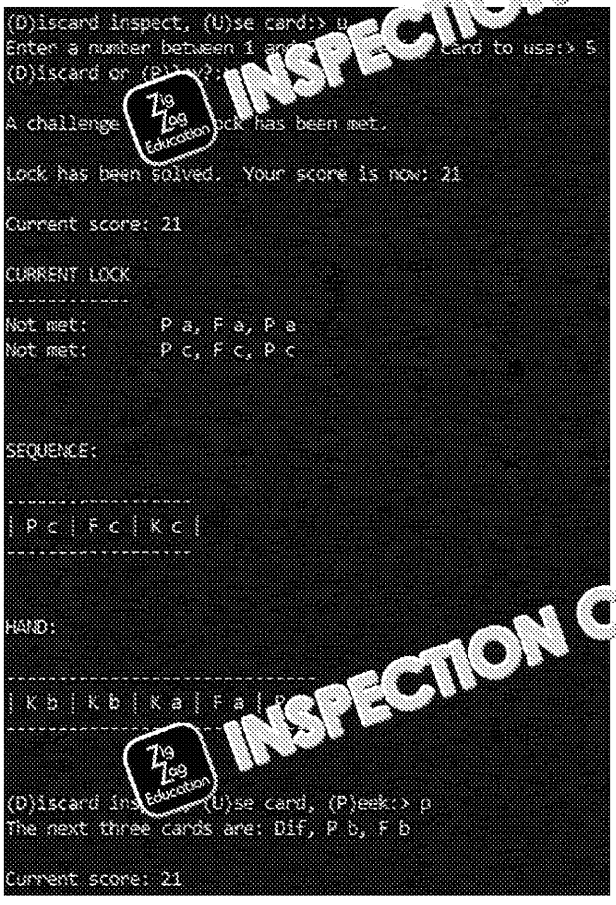
Changes to `Lock`

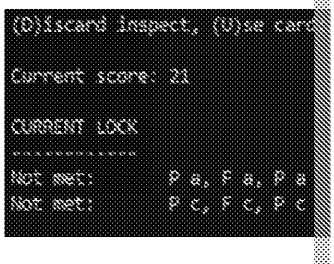
```
class Lock {
    protected List<Challenge> challenges = new ArrayList<>();
    //code added
    private boolean peekUsed = false;

    public boolean getPeekUsed() {
        return peekUsed;
    }

    public void setPeekUsed() {
        peekUsed = true;
    }
    //end addition
```

Testing:

- Peek is an option, works correctly and then disappears [1 mark] ↓

- Peek reappears for the next lock and works until then disappears and doesn't wo





Task 3

Coding:

- Checking for correct condition to print out the error [1 mark]
- Printing out a sensible error message with the card or type that is in error [1 m

Example Solution

Changes to playCardToSequenc

```
if (hand.getCardDescriptionAt(cardChoice - 1).charAt(0) !=
sequence.getCardDescriptionAt(sequence.getNumberOfCards() - 1)
      score = moveCard(hand, sequence, hand.getCardNumberAt(ca
      getCardFromDeck(cardChoice);
      // code added
} else {
      Console.writeLine("Error: The card you are trying to play(
      (cardChoice -1) + ") is the same type as the last card i
      // end addition
}
```

Testing:

- Showing the error message and the hand and sequence afterwards confirming tha discarded [1 mark] ↓

Task 4

Coding:

- Printing out the correct message only when a mulligan is available [1 mark]
- Adding the `mulliganUsed` attribute to `Breakthrough` and initialising it to False [1
- Implementing the mulligan to add all cards from the player's hand, the discard pile
- Shuffling up and dealing again (and discarding any difficulty cards drawn) [1 mark]

Example Solution

Changes to getChoice

```
Console.writeLine();
//code change
if (mulliganUsed) {
      Console.write("(D)iscard inspect, (U)se card:> ");
} else {
      Console.write("(D)iscard inspect, (U)se card, (M)ulligan
}
//end change
String choice = Console.readLine().toUpperCase();
```

Changes to Breakthrough

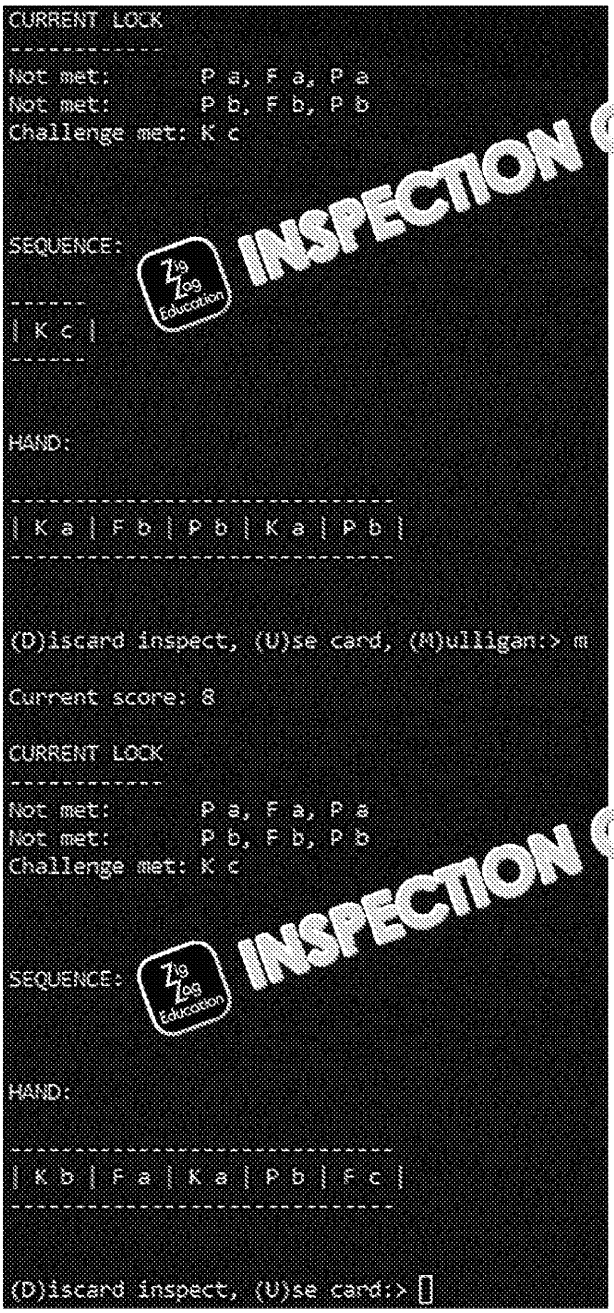
```
private boolean lockSolved;
//code added
private boolean mulliganUsed = false;
//end addition
```

Changes to playGame

```
            Console.writeLine(discard.getCardDisplay());
            break;
      // code added
      case "M":
            if (!mulliganUsed) {
                  int loopCount;
                  //move cards from sequence to deck
                  loopCount = sequence.getNumberOfCards();
                  for (int count=0; count<loopCount;count++) {
                        moveCard(sequence,deck,sequence.getCard
                  }
                  //move cards from discard pile to deck
                  loopCount = discard.getNumberOfCards();
                  for (int count=0; count<loopCount;count++) {
                        moveCard(discard,deck,discard.getCardNu
                  }
                  //move cards from hand to deck
                  loopCount = hand.getNumberOfCards();
                  for (int count=0; count<loopCount;count++) {
                        moveCard(hand,deck,hand.getCardNumberAt
                  }
                  //shuffle up and deal
                  deck.shuffle();
                  for (int count=0; count<5; count++) {
                        while (deck.getCardDescriptionAt(0)=="
                              moveCard(deck,discard,deck.getCar

                        moveCard(deck,hand,deck.getCardNumberAt

                  mulliganUsed = true;
            }
            break;
      //end addition
      case "U":
```

Testing:

- Showing a mulligan being used after solving a challenge [1 mark] ↓



- Showing an attempt to use the mulligan again failing [1 mark] →

Task 5

Coding:

- Printing out quit as a menu option and including it in the selection statement in pl
- Cleanly exiting the main game loop in playGame without using System.exit or a successfully ending the program [1 mark]

Example Solution

Changes to getChoice

```
private Str    etChoice() {
      Conso    teLine();
      //code change
      Console.write("(D)iscard inspect, (U)se card, (Q)uit:> ");
      //end change
      String choice = Console.readLine().toUpperCase();
      return choice;
}
```

Changes to playGame

```
public void playGame() {
      String menuChoice;
      //code added
      boolean hasQuit = false;
      //end addition
      if (locks.size() > 0) {
            gameOver = false;
            currentLock = new Lock();
            setupGame();
            while (!gameOver
                  loc      = false;
                  //code change
                  while (!lockSolved && !gameOver && !hasQuit)
                  //end change
                        Console.writeLine();
                        Console.writeLine("Current score: " +
                        Console.writeLine(currentLock.getLockD
                        Console.writeLine(sequence.getCardDisp
                        Console.writeLine(hand.getCardDisplay()
                        menuChoice = getChoice();
                        switch (menuChoice) {
                              case "D":
                                    Console.writeLine(discard.
                                    break;
                                    //code added
                              case "Q":
                                    hasQuit=true;
                                    break;
                                    //end addition
                              case "U":
                                    int cardChoice = getCardCho
                                    String discardOrPlay = getI
                                    if (discardOrPlay.equals("
                                          moveCard(hand, discar
                                          hand.getCardNumberAt(
                                          getCardFromDeck(card
                                    } else if (discardOrPlay.e
                                          playCardToSequence(c
                                    }
```

```
                                    break;
                              }
                              if (currentLock.getLockSolved()) {
                                    lockSolved = true;
                                    processLockSolved();
                              }
                        }
                        //added the if code to the existing code i
                        if (hasQuit) {
                              gameOver = true;
                              score += deck.getNumberOfCards();
                              Console.writeLine("Final score: " + sc
                        } else {
                              gameOver = checkIfPlayerHasLost();
                        }
                        //end change
                  }
            } else {
                  Console.writeLine("No locks in file.");
            }
}
```

Testing:

- Printing out a final score of 33 (if a Pick was played), 34 (if a File was played) or 3

```
HAND:

-----------------------------
| P b | P a | F c | K b | K c |
-----------------------------


(D)iscard inspect, (U)se card, (Q)uit:> u
Enter a number between 1 and 5 to specify card to use:> 2
(D)iscard inspect, (U)se card, (Q)uit:> d

Current score: 1

CURRENT LOCK
------------
Not met:      P a, F a, P a
Not met:      K b


SEQUENCE:

------
| P a |
------


HAND:

-----------------------------
| P b | F c | K b | F a |
-----------------------------


(D)iscard inspect, (U)se card, (Q)uit:> q
Final score: 33
```

Task 6

Coding:

- Adding the three attributes numPicks, numKeys and numFiles to the CardColle
  [1 mark]
- Ensuring that at least one attribute is updated corr[illegible] w[illegible]n a card is added [1 m
- Ensuring that all three attributes are updated [illegible]re th[illegible]when a card is removed [1 m
- Creating a displayStats method th[illegible]ll[illegible]t out the percentage of each type o
  if not to two decimal places). Note that 'correctly' means dividing the number of th
  in the deck. [1 mark]

Example Solu[illegible]

Changes to the CardCollection class

```
class CardCollection {
    protected List<Card> cards = new ArrayList<>();
    protected String name;
    //code added
    private int numPicks = 0;
    private int numFiles = 0;
    private int numKeys = 0;
    //end addition

    public CardCollection(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public int getCardNum[illegible]At(int x) {
        return cards.[illegible].getCardNumber();
    }

    public S[illegible]g getCardDescriptionAt(int x) {
        return cards.get(x).getDescription();
    }

    public void addCard(Card c) {
        //code added
        switch (c.getDescription().charAt(0)) {
            case 'F':
                numFiles++;
                break;
            case 'P':
                numPicks++;
                break;
            case 'K':
                numKeys++;
                break;
        }
        //end addition
        cards.add(c);
    }

    public [illegible]etNumberOfCards() {
        return cards.size();
    }

    public void shuffle() {
```

```
        Card tempCard;
        int rNo1, rNo2;
        Random rNoGen = new Random();
        for (int count = 1; count <= 10000; count++) {
            rNo1 = rNoGen.nextInt(cards.size());
            rNo2 = rNoGen.nextInt(cards.size());
            tempCard = cards.get(rNo1);
            cards.set(rNo1, cards.get(rNo2));
            cards.set(rNo2, tempCard);
        }
    }

    public Card removeCard(int cardNumber) {
        boolean cardFound = false;
        int pos = 0;
        Card cardToGet = null;
        while (pos < cards.size() && !cardFound) {
            if (cards.get(pos).getCardNumber() == cardNumber) {
                cardToGet = cards.get(pos);
                cardFound = true;
                cards.remove(pos);
            }
            pos++;
        }
        //code added
        switch (cardToGet.getDescription().charAt(0)) {
            case 'F':
                numFiles--;
                break;
            case 'P':
                numPicks--;
                break;
            case 'K':
                numKeys--;
                break;
        }
        //end addition
        return cardToGet;
    }

    //code added
    public void displayStats() {
        float keyChance, pickChance, fileChance;
        keyChance = (float)numKeys/getNumberOfCards() * 100;
        fileChance = (float)numFiles/getNumberOfCards() * 100;
        pickChance = (float)numPicks/getNumberOfCards() * 100;
        Console.writeLine("There is a " + String.format("%.2f",
the next card will be a key, a " + String.format("%.2f",fileCha
will be a file and a " + String.format("%.2f",pickChance) + "%
file.");
    }
    //end addition
```

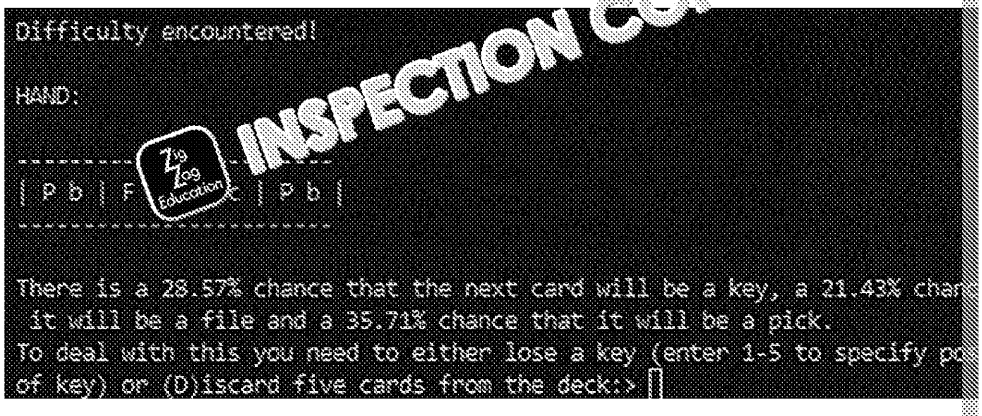
Changes to getCardFromDeck

```
Console.writeLine(deck.getCardDisplay());
//code added
deck.displayStats();
//end addition
Console.write("To deal with this you need to either lose a key
```

Testing:

- Showing the percentage of at least one tool (even if incorrect) to two decimal place which card the player would like to select or whether to discard from the deck.

  Note that the percentages are unlikely to match the ones below. [1 mark] ↓



Task 7

Coding:

- Adding one multi-tool of each kind to the deck at creation time [1 mark]
- Adding one multi-tool of each kind to the deck whenever a lock is solved [1 mark]
- Adding the three setCardToolkit methods that successfully allow a card's toolkit
- Changing playCardToSequence to ask which toolkit the player would like whenev
- Calling setCardToolkit for the correct card and toolkit from playCardToSequen

Example Solution

Changes to createStandardDeck

```
//code added
newCard = new ToolCard("P", "m");
deck.addCard(newCard);
newCard = new ToolCard("K", "m");
deck.addCard(newCard);
newCard = new ToolCard("F", "m");
deck.addCard(newCard);
//end addition
}
```

Changes to processLockSolved

```
while (discard.getNumberOfCards() > 0) {
      moveCard(discard, deck, discard.getCardNumberAt(0));
}
//code added
Card newCard;
newCard = new ToolCard("P", "m");
deck.addCard(newCard);
newCard = new ToolCard("K", "m");
deck.addCard(newCard);
newCard = new ToolCard("F", "m");
deck.addCard(newCard);
//end addition
deck.shuffle();
```

Changes to playCardToSequence

```
private void playCardToSequence(int cardChoice) {
      //code added
      if (hand.getCardDescriptionAt(cardChoice-1).charAt(2) ==
            String toolKit;
            Console.writeLine();
            Console.write("Which toolkit would you like to cho
            toolKit = Console.readLine();
            hand.setCardToolkit(cardChoice-1,toolKit);
      }
      //end addition
      if (sequence.getNumberOfCards() > 0) {
```

Creation of setCardToolkit in CardCollection

```
//code added
public void setCardToolkit(int position, String toolKit) {
      if (cards.get(position).getDescription().charAt(2) == 'm
            cards.get(position).setCardToolkit(toolKit);
      }
}
//end addition
```

Creation of `setCardToolkit` in `ToolCard`

```
//code added
@Override
public void setCardToolkit(String toolKit) {
      kit = toolKit;
}
//end addition
}
```

Creation of `setCardToolkit` in `ToolCard`

```
//code added
public void setCardToolkit(String toolKit) {
}
//end addition
```

Testing:

- Showing the sequence updated with the card played of the `toolkit` chosen [1 m

Task 8

- Changing `playGame` to pass in the argument for the sequence to `getLockDetai` to accept the new `parameter` [1 mark]
- Changing `getLockDetails` to match a single card on the ...uence to the first ca... message to partially met, also to not crash when t... ...n empty sequence [1 ...
- Changing `getLockDetails` to generate ...lly met message when only the la... first card of a `challenge` [1 mark]
- Changing `getLockDet...` ...enerate a partially met message when the last tw... first two car... c... ...ge. [1 mark]

Example Soluti...

Changes to `getLockDetails`

```
//code change
public String getLockDetails(CardCollection sequence) {
//end change
      String lockDetails = System.lineSeparator() + "CURRENT L
      System.lineSeparator() + "----------------" + System.lineSepa
      for (Challenge c : challenges) {
            if (c.getMet()) {
                  lockDetails += "Challenge met: ";
            } else {
                  //code change: current code moved to new els
                  List<String> condition = c.getCondition();
                  int sequenceLength = sequence.getNumberOfCard
                  switch (condition.size()
                        case 3:
                              if (sequenceLength > 0 &&
                              condition.get(0).equals(sequence
                              (sequenceLength)) && condition.g
                              (sequence.getCardDescriptionAt(s
                                    lockDetails += "Partially
                              }
                              else if (sequenceLength >= 0 &&
                              (sequence.getCardDescriptionAt(s
                                    lockDetails += "Partially
                              } else {
                                    lockDetails += "Not met:
                              }
                              break;
                        case 2:
                              if (sequenceLength >= 0 && condi
                              (sequence.getCardDescriptionAt(s
                                    lockDetails += "Partially
                              } else {
                                    lockDetails += "Not met:
                              }
                              bre
                        def
                              lockDetails += "Not met:
                              break;
                  }
                  //end change
            }
            lockDetails += convertConditionToString(c.getCondi
            System.lineSeparator();
```

Changes to playGame

```
while (!lockSolved && !gameOver) {
        Console.writeLine();
        Console.writeLine("Current score: " + score);
        //code change
        Console.writeLine(currentLock.getLockDetails(sequence));
        //end change
        Console.writeLine(sequence.getCardDisplay());
```

Testing:

- First card pla[illegible] sequence doesn't generate partially met (if it doesn't match) a[illegible]

```
CURRENT LOCK
-----------
Not met:      P a, F a, P a
Not met:      P c, F c, P c


SEQUENCE:

------
| F b |
------


HAND:

-----------------------------
| P a | P c | F b | P b | F b |
-----------------------------


(D)iscard inspect, (U)se card:> u
Enter a number between 1 and [illegible] to use:> 1
(D)iscard or (P)[illegible]:>

Current score:

CURRENT LOCK
-----------
Partially met: P a, F a, P a
Not met:       P c, F c, P c


SEQUENCE:

-----------
| F b | P a |
-----------


HAND:

-----------------------------
| P c | F b | P b | F b | P c |
-----------------------------
```

```
HAND:

-------------
| P c | K b | P
-------------


(D)iscard inspe
Enter a number
(D)iscard or (P

Current score:

CURRENT LOCK
-----------
Partially met:
Not met:


SEQUENCE:

-----------
| F b | P a |
-----------


HAND:

-------------
| P c | K b | P
-------------


(D)iscard inspe
```

- Last two cards [illegible] th[illegible]ce matching first two of a challenge g[illegible]s [illegible]rtially met [1 mark] →

Task 9

Coding:

- Adding a variable for bonusCounter and initialising it to 0 for each new lock [1 mark]
- Adding 1 to the variable each time a card is played or discarded [1 mark]
- Awarding the correct bonus once the lock is solved (including 0 if over 20 cards were played) and resetting to 0 [1 mark]

Example Solution

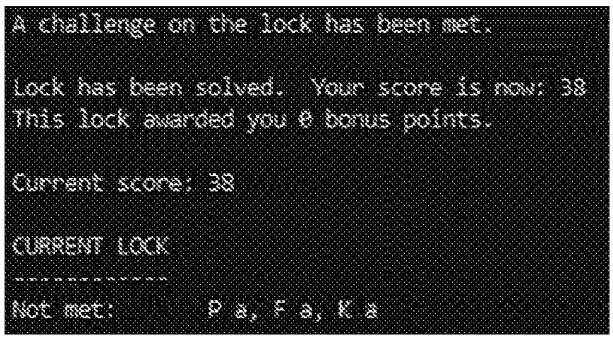
Changes to play:

```
while (!gameOver) {
	lockSolved = false;
	//code added
	int bonusCounter = 0;
	//end addition
	while (!lockSolved && !gameOver) {
		Console.writeLine();
		Console.writeLine("Current score: " + score);
		Console.writeLine(currentLock.getLockDetails());
		Console.writeLine(sequence.getCardDisplay());
		Console.writeLine(hand.getCardDisplay());
		menuChoice = getChoice();
		switch (menuChoice) {
			case "D":
				Console.writeLine(discard.getCardDispla
				break;
			case "U":
				int cardChoice = getCardChoice();
				String discardOrPlay = getDiscardOrPlay
				//code added
				bonusCounter += 1;
				//end addition
				if (discardOrPlay.equals("D")) {
					moveCard(hand, discard, hand.getCa
					getCardFromDeck(cardChoice);
				} else if (discardOrPlay.equals("P")) {
					playCardToSequence(cardChoice);
				}
				break;
		}
		if (currentLock.getLockSolved()) {
			lockSolved = true;
			processLockSolved();
			//code added
			int bonus = Math.max(0, 20 - bonusCounter);
			score += bonus;
			Console.writeLine("This lock awarded you " +
			bonusCounter = 0;
			//end addition
		}
	}
	gameOver = checkIfPlayerHasLost();
```

Testing:

- Solving a lock in under 20 cards and getting the correct bonus (which doesn't hav
  [1 mark] ↓

- Solving a lock in over 20 cards and getting 0 bonus. [1 mark] ↓



Task 10

Coding:

- Modifying `setupGame` to have a 25% chance of adding a `GeniusCard` [1 mark]
- Modifying `processLockSolved` to have a 25% chance of adding a `GeniusCard` [1 mark]
- Creating a `GeniusCard` class that inherits from `Card` with a constructor with `self.cardType` to `Gen` [1 mark]
- Asking the user to enter a challenge number or discard when a genius card is drawn [1 mark]
- Processing the `GeniusCard` correctly to solve the challenge chosen [1 mark]
- Processing the `GeniusCard` correctly to discard it [1 mark]
- Handling the drawing of a `GeniusCard` correctly if drawn while refilling the hand (e.g. print a message) [1 mark]

Example Solution

Creation of `addGeniusCardToDeck`

```
//code added
private void addGeniusCardToDeck() {
      deck.addCard(new GeniusCard());
}
//end addition
```

Creation of `GeniusCard`

```
//code added
class GeniusCard extends Card {
    protected String cardType;

    public GeniusCard() {
        super();
        cardType = "Gen";
    }

    public GeniusCard(int cardNo) {
        cardType = "Gen";
        cardNumber = cardNo;
    }

    @Override
    public String getDescription() {
        return cardType;
    }
}
//end addition
```

Changes to `setupGame`

```
addDifficultyCardsToDeck();
//code added
if (rNoGen.nextInt(4) == 1) {
      addGeniusCardToDeck();
}
//end addition
deck.shuffle();
```

Changes to processLockSolved

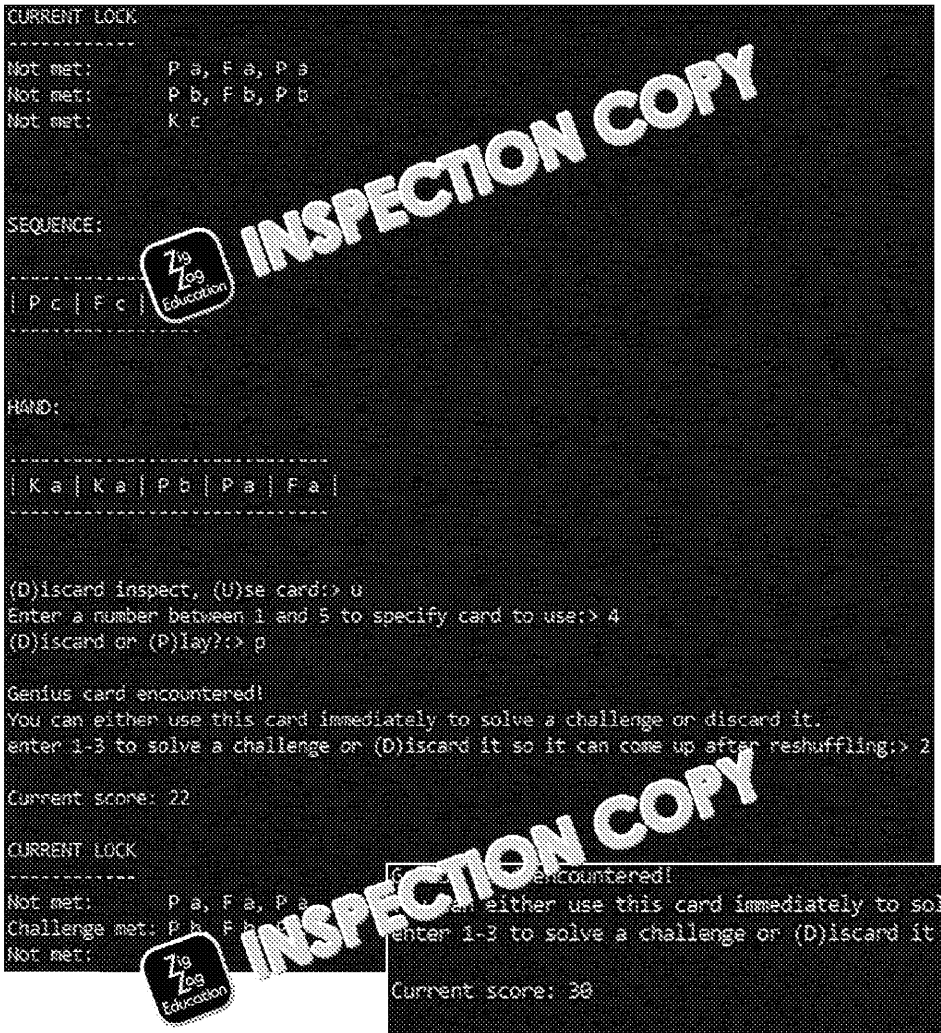
```
while (discard.getNumberOfCards() > 0) {
      moveCard(discard, deck, discard.getCardNumberAt(0));
}
//code added
if (rNoGen.nextInt(4) == 1) {
      addGeniusCardToDeck();
}
//end addition
deck.shuffle();
currentLock = getRandomLock();
}
```

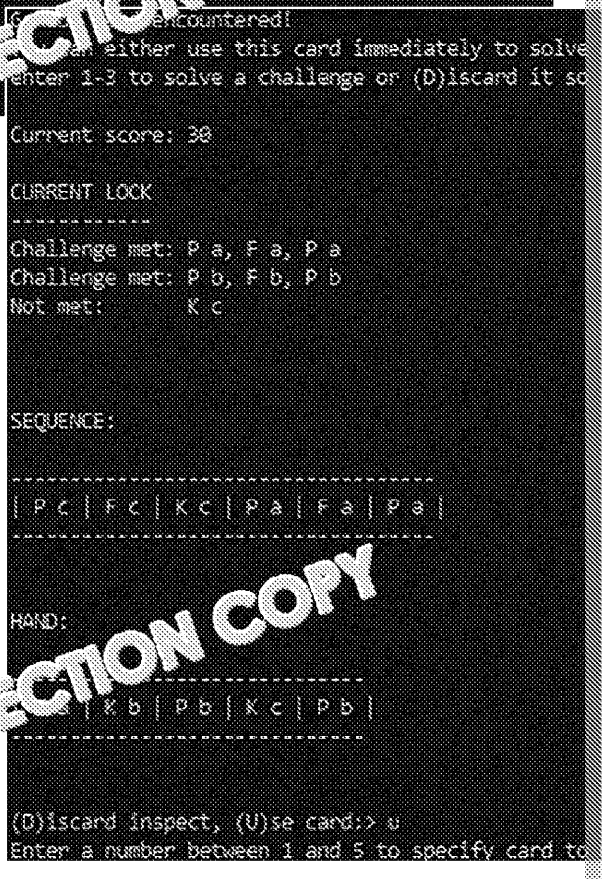
Changes to getCardFromDeck

```
      discard.addCard(currentCard);
      currentCard.process(deck, discard, hand, sequence, curren
      //code added
} else if (deck.getCardDescriptionAt(0).equals("Gen")) {
      Card currentCard = deck.removeCard(deck.getCardNumberAt(
      Console.writeLine();
      Console.writeLine("Genius card encountered!");
      Console.writeLine(hand.getCardDisplay());
      Console.write("You can either use this card immediately
      discard it.");
      Console.write("Enter 1-"+currentLock.getNumberOfChallenge
      or (D)iscard it so it can come up again after reshuffling:> ");
      String choice = Console.readLine().toUpperCase();
      Console.writeLine();
      if (choice.equals("D")) {
            discard.addCard(currentCard);
      } else {
            currentLock.setChallengeMet(Integer.parseInt(choice
      }
      //end addition
}
}
while (hand.getNumberOfCards() < 5 && deck.getNumberOfCards() >
if (deck.getCardDescriptionAt(0).equals("Dif")) {
      moveCard(deck, discard, deck.getCardNumberAt(0));
      Console.writeLine("A difficulty card was discarded from
      hand.");
      //code added
} else if (deck.getCardDescriptionAt(0).equals("Gen")) {
      moveCard(deck, discard, deck.getCardNumberAt(0));
      Console.writeLine("A genius card was discarded from the dec
      //end addition
} else {
      moveCard(deck, hand, deck.getCardNumberAt(0));
}
```

Testing:

- Using a GeniusCard successfully [1 mark] ↓





- Discarding a GeniusCard successfully [1 mark] →

Task 11

Coding:

- Adding the `credits` attribute and initialising it to 10 [1 mark]
- Asking whether the player would like to buy a tool only wh[en] they have played or [if] credits left [1 mark]
- Ensuring that keys are not listed if they ha[ve] credits remaining (even if they di[d]
- Adding a `ToolCard` of the correct [ty]pe to the toolkit to the player's hand in the fifth p[lace]
- Removing the `ToolCard` fr[om] the deck at the correct position [1 mark]
- Deducting the correct number of credits for buying a card [1 mark]
- Printing a li[st] number of each tool available and not printing tools where the[re]
- Printing optio[n] correctly at the end of the menu, once and once only [1 mark]
- Having an iteration statement to correctly calculate the number of tools of each ty[pe]
- Returning a list from `printToolsAvailable` that contains the index of a card wit[h]

Example Solution

Adding the `credits` attribute

```
private boolean lockSolved;
//code added
private int credits = 10;
//end addition
```

Changes to `getCardFromDeck`

```
if (deck.getNumberOfCards() > 0) {
      //code added
      if (credits>=2) {
            Console.writeLine();
            Console.write("Would you like to buy a tool (y/n)?
            String choice = Console.readLine().toLowerCase();
            if (choice.equals("y")) {
                  boolean toolsAvailable = credits>2;
                  List<Integer> toolList = deck.printToolsAvai
                  Console.write("Which tool would you like to
                  int cardChosen = Integer.parseInt(Console.re
                  if (cardChosen != 10 && toolList.get(cardCho
                        moveCard(deck,hand,deck.getCardNumberAt(
                        if (cardChosen > 6) {
                              credits -=3;
                        } else {
                              credits -=2;
                        }
                  }
            }
      }
      //end addition
```

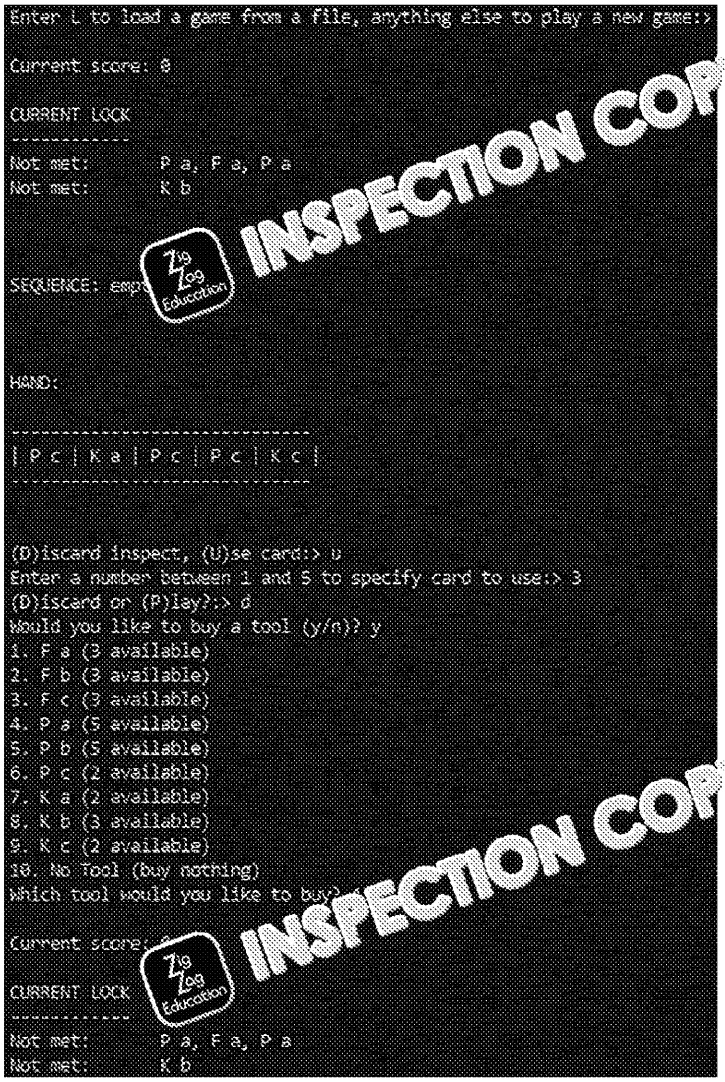
New `printToolsAvailable` method in `CardCollection`

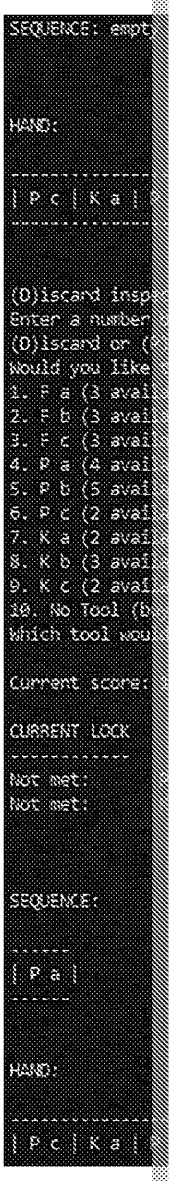
```
//code added
public List<Integer> printToolsAvailable(bool[illegible]keysAvailable)
      List<String> tools = new ArrayList<>(List.of("F a", "F b
      c", "K a", "K b", "K c"));
      List<Integer> toolList = new ArrayList<>(List.of(-1,-1,-
      List<Integer> toolsAvailable = new ArrayList<>(List.of(0
      for (int i=0;i<[illegible]berOfCards()-1;i++) {
            [illegible]s.get(i).getDescription()) {
                  case "F a":
                        toolsAvailable.set(0,toolsAvailable.ge
                        if (toolList.get(0) == -1) {
                              toolList.set(0,i);
                        }
                        break;
```

```
                case "P b":
                      toolsAvailable.set(1,toolsAvailable.ge
                      if (toolList.get(1) == -1) {
                            toolList.set(1,i);
                      }
                      break;
                case "P c":
                      toolsAvailable.set(2,toolsAvailable.ge
                      if (toolList.get(2) == -1) {
                            toolList.set(2,i);
                      }
                      break;
                case "P a":
                      toolsAvailable.set(3,toolsAvailable.ge
                      if (toolList.get(3) == -1) {
                            toolList.set(3,i);
                      }
                      break;
                case "P b":
                      toolsAvailable.set(4,toolsAvailable.ge
                      if (toolList.get(4) == -1) {
                            toolList.set(4,i);
                      }
                      break;
                case "P c":
                      toolsAvailable.set(5,toolsAvailable.ge
                      if (toolList.get(5) == -1) {
                            toolList.set(5,i);
                      }
                      break;
                case "K a":
                      if (keysAvailable) {
                            toolsAvailable.set(6,toolsAvaila
                            if (toolList.get(6) == -1) {
                                  toolList.set(6,i);
                            }
                      }
                      break;
                case "K b":
                      if (keysAvailable) {
                            toolsAvailable.set(7,toolsAvaila
                            if (toolList.get(7) == -1) {
                                  toolList.set(7,i);
                            }
                      }
                      break;
                case "K c":
                      if (keysAvailable) {
                            toolsAvailable.set(8,toolsAvaila
                            if (toolList.get(8) == -1) {
                                  toolList.set(8,i);
                            }
                      }
                      break;
          }
      }
      for (int i=0;i<tools.size();i++) {
            if (toolList.get(i) == -1) {
                  Console.WriteLine("" + (i+1) + ". " + tools.g
                  tools available.get(i) + " available);
            }
      }
      Console.WriteLine("10. No tool (buy nothing)");
      return toolList;
}
//end addition
```

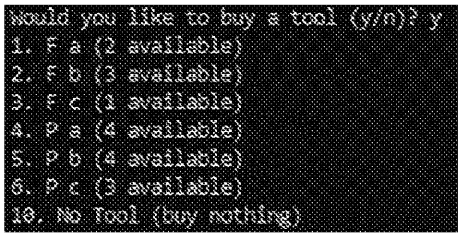
Testing:

- Buying two tools [1 mark] ↓





- Trying to buy a tool with 2 credits left [1 mark] ↓



Task 12

Coding:

- Selection statement to check whether the challenge just solved was at least three
- Iteration statement to run once for each tool card in the challenge just solved [1 m
- Call to moveCard to move one tool card from the sequence to the discard pile insi

Example Solution

```
if (currentLock.challengeConditionMet(sequenceAsString)) {
      //code added
      if (sequenceAsString.length() >= 10) {
            for (int i=0; i<sequenceAsString.length()/4;i++) {

      moveCard(sequence,discard,sequence.getCardNumberAt(sequen
            }
      }
      //end addition
      return true;
```

Testing:

- Solve a challenge with one card, then with three cards [1 mark] ↓

Task 13

Coding:

- Changing `getChoice` to correctly prompt you to (S)ave the game and `playGame` to call `saveGame` if 'S' was chosen [1 mark]
- Returning a string of the correct format for the save file from `getChallengesAsString` [1 mark]
- Returning a string of the correct format for the save file from `getChallengesMetAsString` [1 mark]
- Saving the current score to the save game file [1 mark]
- Saving the current lock to the save game file [1 mark]
- Saving the player's sequence, deck and discard pile to the save game file [1 mark]
- Having a method or a loop that creates the string for a `CardCollection` in the correct format [1 mark]

Example Solution

Changes to `getChoice`

```
private String getChoice() {
	Console.writeLine();
	//code change
	Console.write("(D)iscard inspect, (U)se card, (S)ave:> ");
	//end addition
	String choice = Console.readLine().toUpperCase();
```

Changes to `playGame`

```
switch (menuChoice) {
	case "D":
		Console.writeLine(discard.getCardDisplay());
		break;
		//code added
	case "S":
		saveGame();
		break;
		//end addition
```

Code for `saveGame` (added imports were of course at the top of the program)

```
//code added
import java.io.FileWriter;
import java.io.IOException;
//end addition
```

...

```
//code added
private void saveGame() {
	try {
		FileWriter myWriter = new FileWriter("game2.txt");
		myWriter.write(score + ",");
		myWriter.write(currentLock.getChallengesAsString()+
		myWriter.write(currentLock.getChallengesMetAsString
		myWriter.write(hand.getCardDescriptionAt(0) + " " +
		for(int i=1;i<hand.getNumberOfCards();i++) {
			myWriter.write("," + hand.getCardDescriptionA
			hand.getCardNumberAt(i));
		}
		myWriter.write("\n");
```

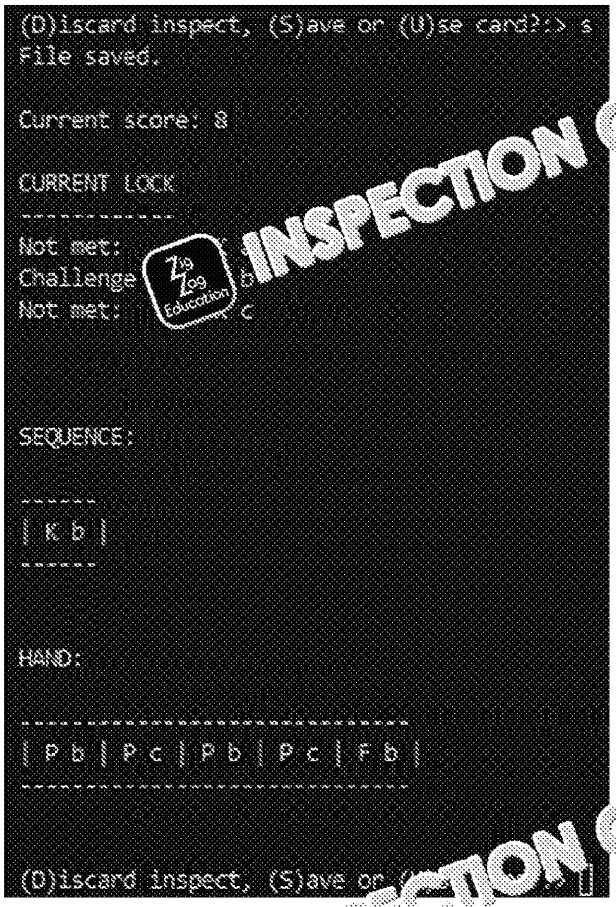
```
            myWriter.write(sequence.getCardDescriptionAt(0) +
            sequence.getCardNumberAt(0));
            for (int i=1;i<sequence.getNumberOfCards();i++) {
                  myWriter.write("," + sequence.getCardDescript
                  sequence.getCardNumberAt(i));
            }
            myWriter.write("\n");
            myWriter.write(discard.getCardDescriptionAt(0) + "
            discard.getCardNumberAt(0));
            for (int i=1;i<discard.getNumberOfCards();i++) {
                  myWriter.write("," + discard.getCardDescripti
                  discard.getCardNumberAt(i));
            }
            myWriter.write("\n");
            myWriter.write(deck.getCardDescriptionAt(0) + " " +
            for (int i=1;i<deck.getNumberOfCards();i++) {
                  myWriter.write("," + deck.getCardDescriptionA
                  deck.getCardNumberAt(i));
            }
            myWriter.close();
            Console.writeLine("File saved.");
      } catch (IOException e) {
            Console.writeLine("File not saved");
      }
}
//end addition
```

Code for getChallengesAsString and getChallengesMetAsString

```
//code added
public String getChallengesAsString() {
      String challengeStr = "";
      for (Challenge c : challenges) {
            if (challengeStr.length()>0) {
                  challengeStr += ";";
            }
            challengeStr += convertConditionToString(c.getCond
      }
      return challengeStr;
}

public String getChallengesMetAsString() {
      String challengeStr = "";
      for (Challenge c : challenges) {
            if (challengeStr.length()>0) {
                  challengeStr += ";";
            }
            if (c.getMet()) {
                  challengeStr += "Y";
            } else {
                  challengeStr += "N";
            }
      }
      return challengeStr;
}
//end addition
```

Testing:

- Saving game then loading game [1 mark] ↓



Task 14

Coding:

- Adding 5 to the bonusPool after adding 5 to the score when completing a challen
- Adding 5 to the bonusPool when playing a card to the sequence that is a partial s
- Resetting the bonusPool to 0 under all circumstances where a card is not played
- Creating the new attribute bonusPool and initialising it to 0 [1 mark]
- Writing the code for isPartial such that it returns True if the card just played add
  add to an existing challenge [1 mark]

Example Solution

Changes to playCardToSequence

```
if (checkIfLockChallengeMet()) {
      Console.writeLine();
      Console.writeLine("A challenge on the lock has been met."
      Console.writeLine();
      //code change
      score += 5 + bonusPool;
      //end change
      bonusPool += 5;
} else {
      //code added
      if (currentLock.isPartial(sequence)) {
            bonusPool += 5;
      } else {
            bonusPool = 0;
      }
      //end addition
}
```

Changes to playGame

```
if (discardOrPlay.equals("D")) {
      moveCard(hand, discard, hand.getCardNumberAt(cardChoice -
      getCardFromDeck(cardChoice);
      //code added
      bonusPool = 0;
      //end addition
```

Code for isPartial

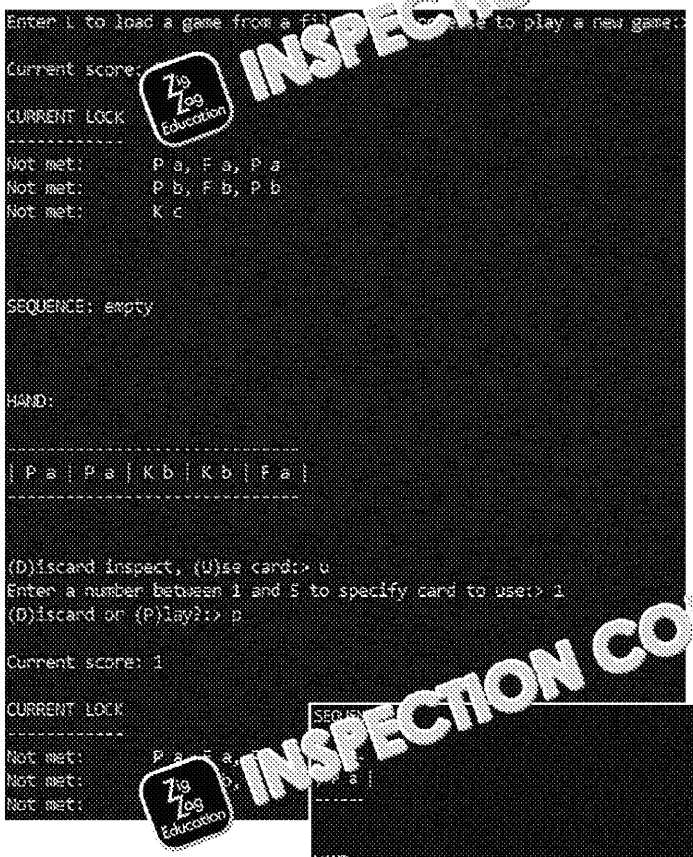
```
//code added
public boolean isPartial(CardCollection seq) {
      boolean part = false;
      for (Challenge c : challenges) {
            List<String> condition = c.getCondition();
            if (condition.size() == 3) {
                  int seqLen = seq.getNumberOfCards() - 1;
                  if (seqLen > 0 && condition.get(1).equals(seq.g
                  && condition.get(0).equals(seq.getCardDescrip
                        part = true;
                  } else if (seqLen == 0 &&
                  condition.get(0).equals(seq.getCardDescriptio
                        part = true;
                  }
            }
      }
      return part;
}
//end addition
```

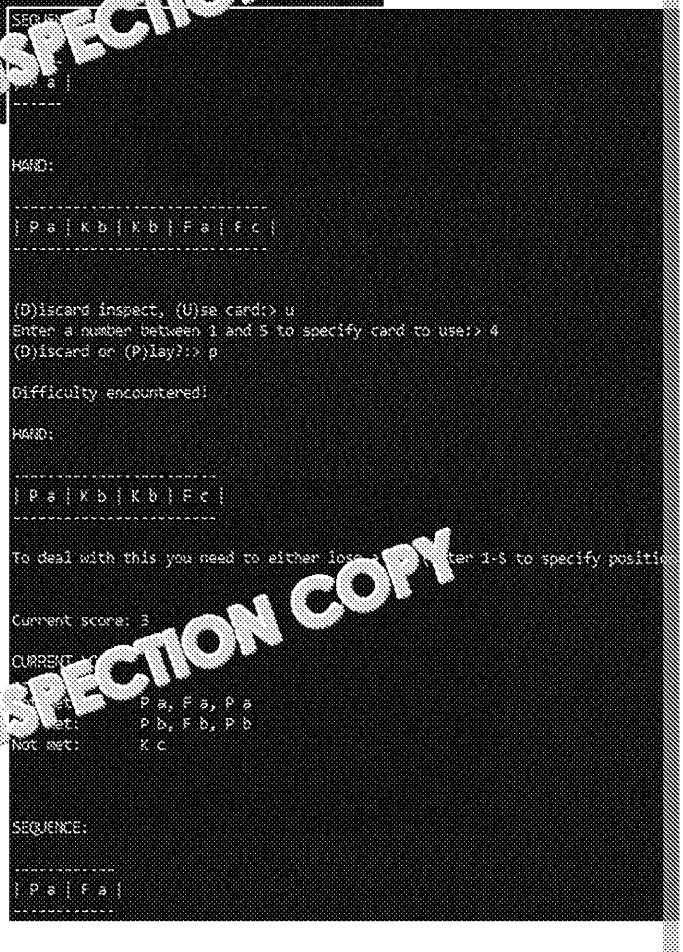
Code for new bonusPool attribute

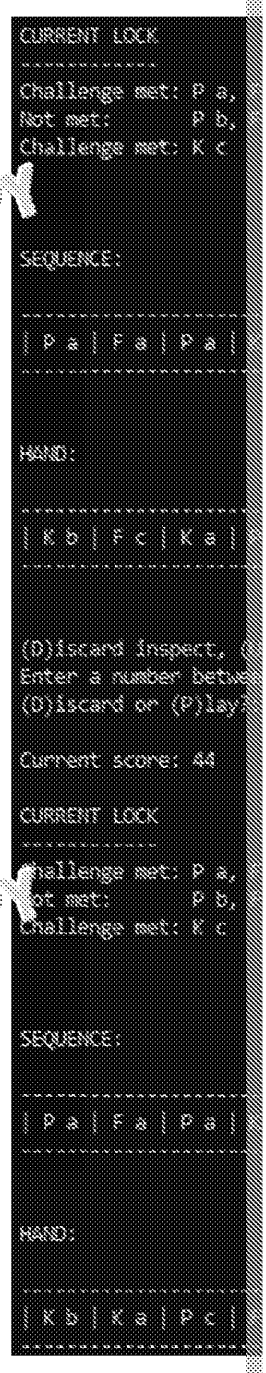
```
//code added
private int bonusPool = 0;
//end addition
```

Testing:

- Playing three cards to solve a challenge (give it one card after another [1 ma







Task 15

Coding:

- Adding `finalLock` as a private attribute and initialising it to 0 [1 mark]
- Adding the selection conditions for `finalLock == 1` and `finalLock == 2` to ch... don't have the correct contents) [1 mark]
- Changing the condition of the selection statement in `getCardFromDeck` correctly [...]
- Changing `processLockSo...` ...e 10 attempts to find a soluble lock [1 mark]
- Changing pro...sL...Solved to call `generateSolubleLock` once 10 attempts [...]
- Changing p...LockSolved to set `finalLock` to 1 and skip the main body of [...]
- Writing gene...teSolubleLock such that it always generates a soluble lock (rule... the same type excluded) [1 mark]
- Writing `generateChallenge` such that it generates a possible challenge from the... have two tools of the same type consecutively) [1 mark]
- Returning True and False correctly from `isSoluble` [1 mark]

Example Solution

Addition of `finalLock` attribute

```
private boolean lockSolved;
//code added
private int finalLock = 0;
//end addition
```

Changes to `checkIfPlayerHasLost`

```
private boolean checkIfPlayerHasLost(
      //code added
      if (finalLock == 1) {
            finalLock = 2;
      } else if (finalLock == 2) {
            Console.writeLine("You have solved the final lock.
            );
            return true;
            //end addition
            //code change
      } else if (deck.getNumberOfCards() == 0) {
            Console.writeLine("You have run out of cards in yo
            is: " + score);
            return true;
      }
      return false;
      //end change
}
```

Changes to `getCardFromDeck`

```
//code change
if ((deck.getNumberOfCards() == 0) && (hand.getNumberOfCards() <
(hand.getNumberOfCards() ==
//end change
      gameOver = true;
}
```

Changes to processLockSolved

```
private void processLockSolved() {
	score += 10;
	Console.writeLine("Lock has been solved.  Your score is
	//code change
	if (finalLock < 2) {
		while (discard.getNumberOfCards() > 0) {
			moveCard(discard, deck, discard.getCardNumber
		}
		deck.shuffle();
		int attempts = 0;
		while (attempts < 10) {
			currentLock = getRandomLock();
			if (currentLock.isSoluble(deck, hand)) {
				break;
			} else {
				attempts++;
			}
		}
		if (attempts == 10) {
			Console.writeLine("Final Lock");
			while (hand.getNumberOfCards() < 5) {
				moveCard(deck,hand,deck.getCardNumberA
			}
			currentLock = generateSolubleLock();
			finalLock = 1;
			gameOver = true;
		}
	}
	//end change
}
```

Code for generateSolubleLock

```
//code added

private Lock generateSolubleLock() {
	List<String> cardsLeft = new ArrayList<>();
	Lock newLock = new Lock();

	for (int i=0;i<deck.getNumberOfCards();i++) {
		cardsLeft.add(deck.getCardDescriptionAt(i));
	}
	for (int i=0;i<hand.getNumberOfCards();i++) {
		cardsLeft.add(hand.getCardDescriptionAt(i));
	}

	newLock.addChallenge(generateChallenge(cardsLeft));
	newLock.addChallenge(generateChallenge(cardsLeft));

	return newLock;
}
```

Code for generateChallenge

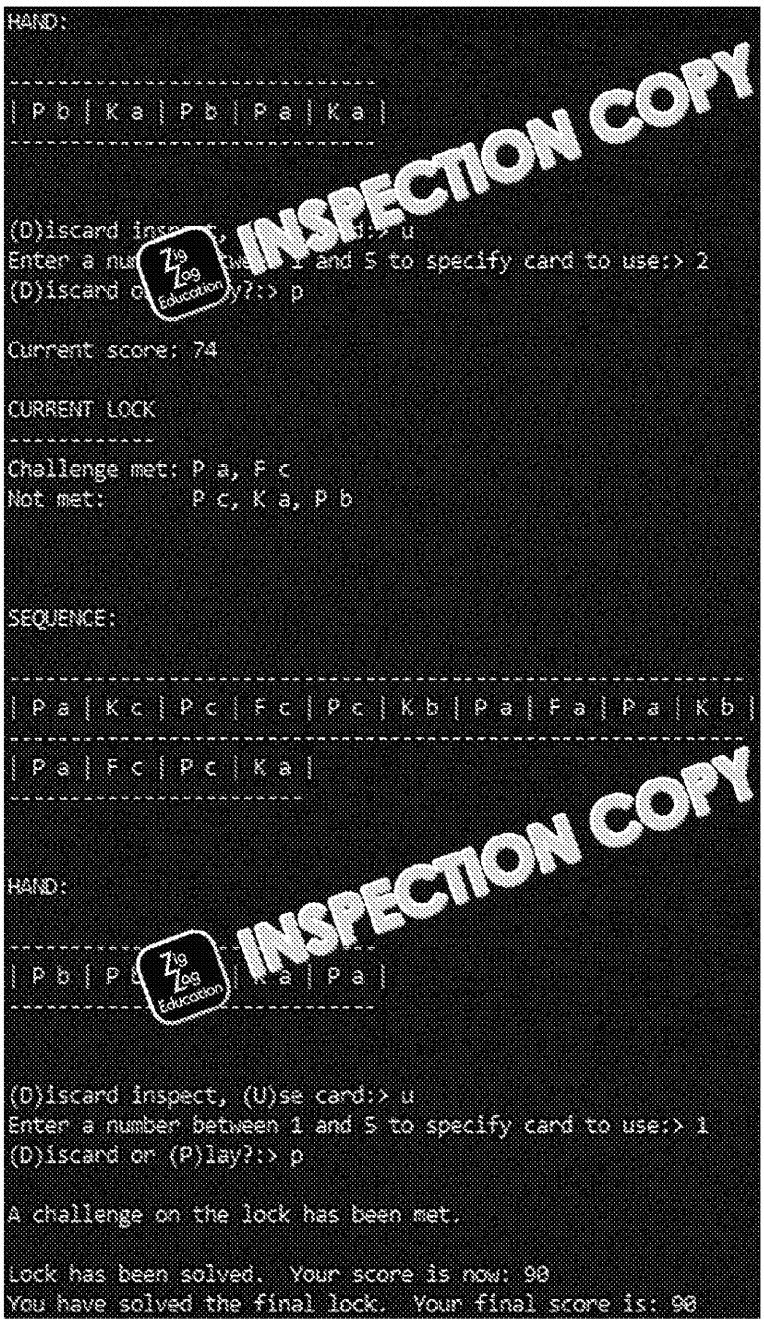
```
private List<String> generateChallenge(List<String> cards) {
	List<String> challng = new ArrayList<>();
	int cardToRemove;
	int tmpRand;

	try {
		cardToRemove = rNoGen.nextInt(cards.size() -1);
		challng.add(cards.get(cardToRemove));
		cards.remove(cardToRemove);
		tmpRand = rNoGen.nextInt(3);

		for (int i=0;i<tmpRand;i++) {
			cardToRemove = rNoGen.nextInt(cards.size() -
			while (cards.get(cardToRemove).charAt(0) ==
			-1).charAt(0)) {
				cardToRemove = rNoGen.nextInt(cards.si
			}
			challng.add(cards.get(cardToRemove));
			cards.remove(cardToRemove);
		}
	} catch (Exception e) {
		//ran out of cards so go with what we have so far
	}

	return challng;
}
```

Code for isSoluble

```
//code added
public boolean isSoluble(CardCollection deck, CardCollection ha
	List<String> cardsLeft = new ArrayList<>();
	List<String> challengesLeft = new ArrayList<>();

	for (int i=0;i<deck.getNumberOfCards();i++) {
		cardsLeft.add(deck.getCardDescriptionAt(i));
	}
	for (int i=0;i<hand.getNumberOfCards();i++) {
		cardsLeft.add(hand.getCardDescriptionAt(i));
	}
	for (Challenge c : challenges) {
		for (String s : c.getCondition()) {
			challengesLeft.add(s);
		}
	}
	for (String s : challengesLeft) {
		if (cardsLeft.contains(s)) {
			cardsLeft.remove(s);
		} else {
			return false;
		}
	}
	return true;
}
//end additio
```

Testing:

- Printing out the final lock [1 mark] ⬇



Name	

ZigZag Education supporting

A Level AQA Computer Science Pap

Summer 2022

BREAKTHROUGH!

Electronic Answer Document (EAD)

Instructions

- Enter your name in the box at the top of this page
- Answer **all** questions by entering your answers into this document
- Remember to **save** this document regularly
- Save and print this document and ar[illegible] additional pages
- Answer **all** qu[illegible]
- The ma[illegible]ailable for each question are shown in brackets
- You will need:
  - access to a computer
  - access to a printer
  - access to appropriate software
  - electronic copies of the required skeleton code
  - EAD (Electronic Answer Document)

Total marks:

Programming Theory Question

Answer all questions. Remember to save this document

Q		Answer
1	(a)	
	(b)	
2	(a)	
	(b)	
	(c)	
3	(a)	
	(b)	
4	(a)	
	(b)	
5	(a)	
	(b)	
6	(a)	
	(b)	

Count	SequenceAsString	Return value
	""	
5		

Count	SequenceAsString	Return value
	""	
5		

Q		Answer
7	(a)	
	(b)	
8	(a)	
	(b)	
9	(a)	
	(b)	
	(c)	
10	(a)	
	(b)	
11		
12		
13	(a)	
	(b)	
14	(a)	
	(b)	
	(c)	
15		

Programming Tasks

Answer all questions. Remember to save this document

Q	Answe
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	