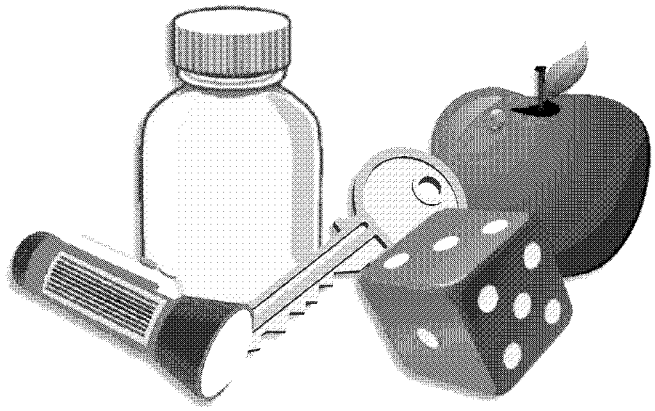
Zig Zag Education

Computer Science

A Level | AQA | 7517

2015 specification for the 2019 exam

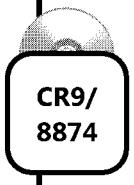


VB.NET

PAPER 1 EXAM RESOURCE PACK 2019

AQA Text Adventures

for A Level AQA Computer Science



CR9/ 8874

POD 8874

zigzageducation.co.uk

Contents

Teacher's Introduction

This resource pack is designed to help you support your students taking the **A Level Computer Science Paper 1** examination. It is based on the **'AQA Text Adventures'** preliminary material (VB.NET) – for examination June 2019.

On the CD, you will find the following files.

`AQATextAdventures`	for student use – this folder contains all of the content, accessible via a HTML interface
`editable`	for teacher use – this folder contains ALL of the documents in editable (docx/pptx) formats
`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 `AQATextAdventures` 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 in the 'Answers & Solutions' HTML page (`answers.html`) are password-protected, so that students can only access them with your permission. Each password is a four-digit code, as follows:

`Commentary.pdf`	1158
`Diagram1Complete.pdf`	4773
`Diagram2Complete.pdf`	5382
`Diagram3Complete.pdf`	3091
`QuestionsMarkScheme.pdf`	7642
`TaskMarkScheme.pdf`	2966

Should you wish to give students access to ALL protected-PDFs, the master password for all files is:

`zz2ghc4`

The resource pack consists of the following:

① **Pre-release Commentary**, consisting of two parts:

- A general walkthrough of the skeleton program; a written description, flowchart and an animated PowerPoint giving a visual demonstration of the game. It is non-technical in the sense that it doesn't reference or explain any actual code elements – only how the program works when it is run.
- A detailed, technical overview of the skeleton program, describing how all VB.NET subroutines, structures and variables work, including the relationship between them (also shown visually as a structure diagram).

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. For this reason, this content has been placed on the 'Answers & Solutions' HTML page as a password-protected file, to allow you to control if/when students access it.

② **Structure Diagram Activities**

Three partially complete structure diagram activities for students to complete while getting to grips with the skeleton program. Any missing identifiers, data types, return values, directional arrows, etc. must be added to the diagram. Solutions are provided on the *Answers & Solutions* page as a protected PDF.

③ **Written Questions**

Theory questions testing students' understanding of the skeleton program, like Section C in the exam. 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 version are available format. Solutions are provided on the *Answers & Solutions* page as a protected PDF.

④ **Programming Tasks**

Fifteen modification exercises put students' programming skills to the test, like Section D in the exam. Solutions are provided on the *Answers & Solutions* page as a protected PDF. Note that these are example solutions and you must use your discretion to award marks accordingly where there are valid alternative solutions.

Free Updates

Register your email address to receive any future free minor updates made to this resource or other Computing resources your school has purchased, and details of any promotions for your subject.

** resulting from minor specification changes, suggestions from teachers and peer reviews, or occasional errors reported by customers*

zzed.uk/freeupdates

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

AQA Text Adventu

Introduction

AQA Text Adventures is a text adventure game, w[illegible] the player enters text comm locations, use various items and interact [illegible] ffe[illegible]nt characters. The program d show the user the result of their [illegible] the current state of the game.

When the pr[illegible]m [illegible] there are no items, characters or places in the game. T game file (w[illegible] ill have the .gme file extension) they would like to use. If the pr a message w[illegible] be displayed to the player telling them that the game could not be terminate. If the file is found, then all of the items, characters and places in that fi and the game runs as below.

1. The game checks whether the player has moved to a new place. If they have, then the game displays the description of that place and the items in that place which the player can take.
2. The game asks the player to input an action.
3. The game identifies the command by splitting up the input into the command and the argument (e.g. if the player enters "get torch", the game will split this into the command "get" and the argument "torch").
4. If the command is recognised, the game will perf[illegible]e corresponding action and display the result.

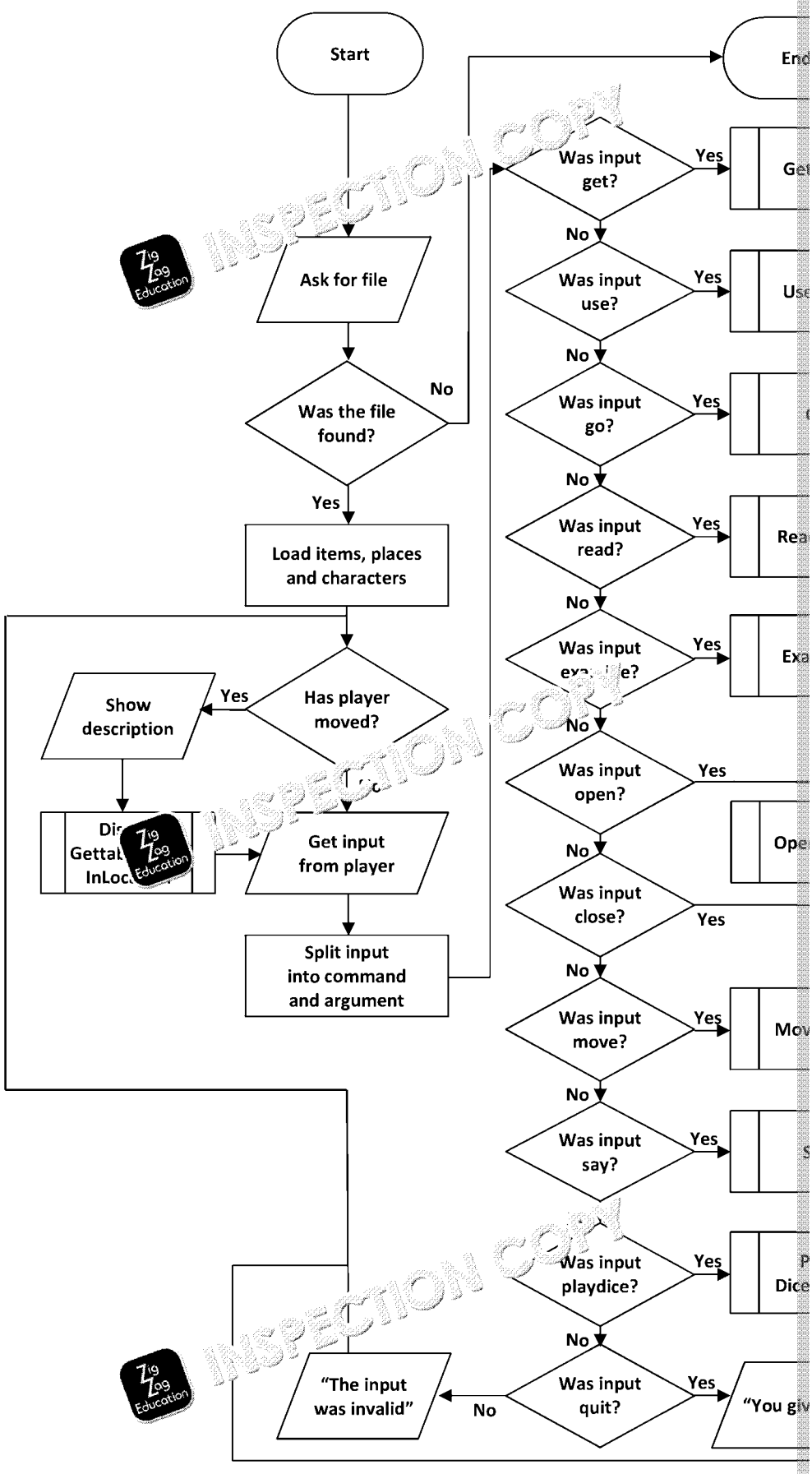
   The recognised commands a[illegible]llows:

`get`	[illegible] the specified item to the player inventory
`use`	Uses an item in the current place or the player's inventory
`go`	Moves the player from their current place to the place in th
`read`	Displays the text of a readable item in the current place or t
`examine`	Provides a description of the specified item, place or charac player's inventory
`open`	Opens the specified item in the current place
`close`	Closes the specified item in the current place
`move`	Moves the specified item in the current place
`say`	Prints the specified input to the screen
`playdice`	Plays a dice game with the [illegible] fi[illegible] character in the curren
`quit`	Ends the game

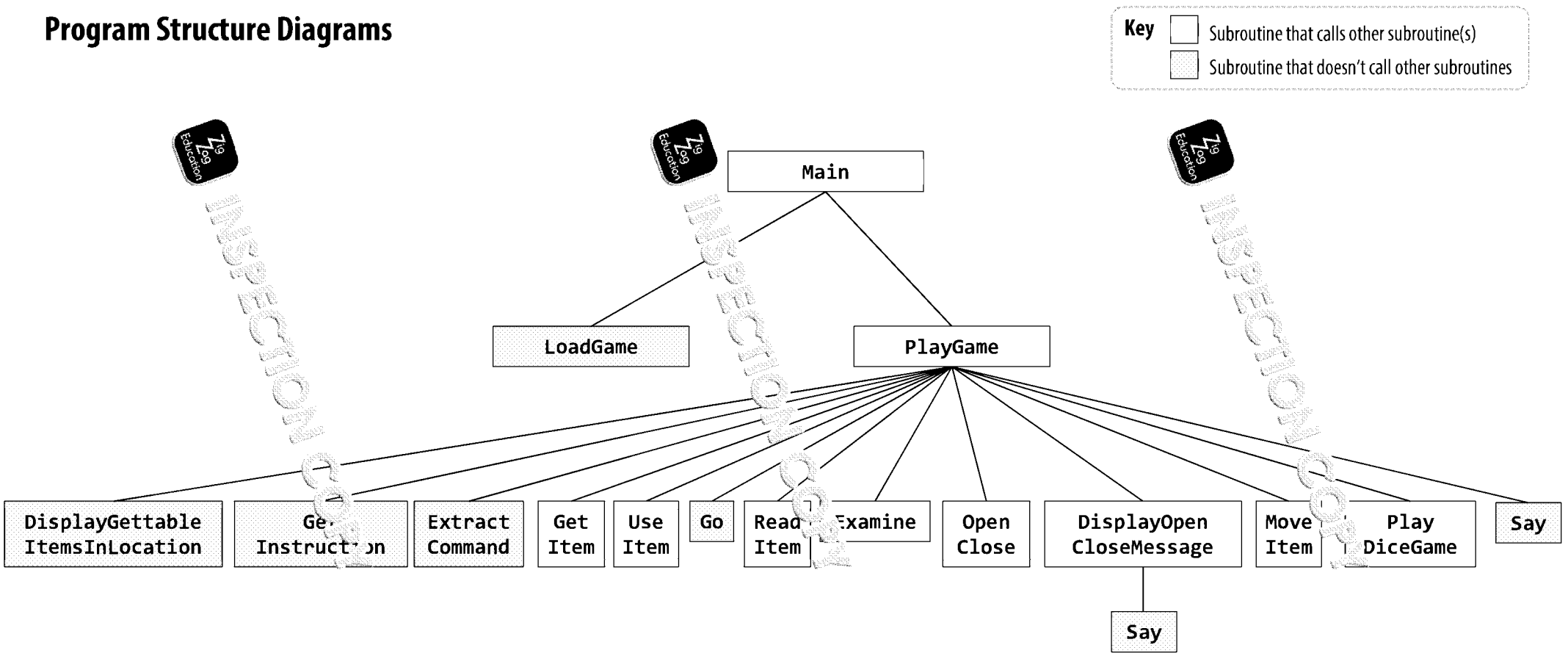
5. If the ga[illegible]do[illegible]r[illegible]ognise the command, a message is displayed to say th that au[illegible]

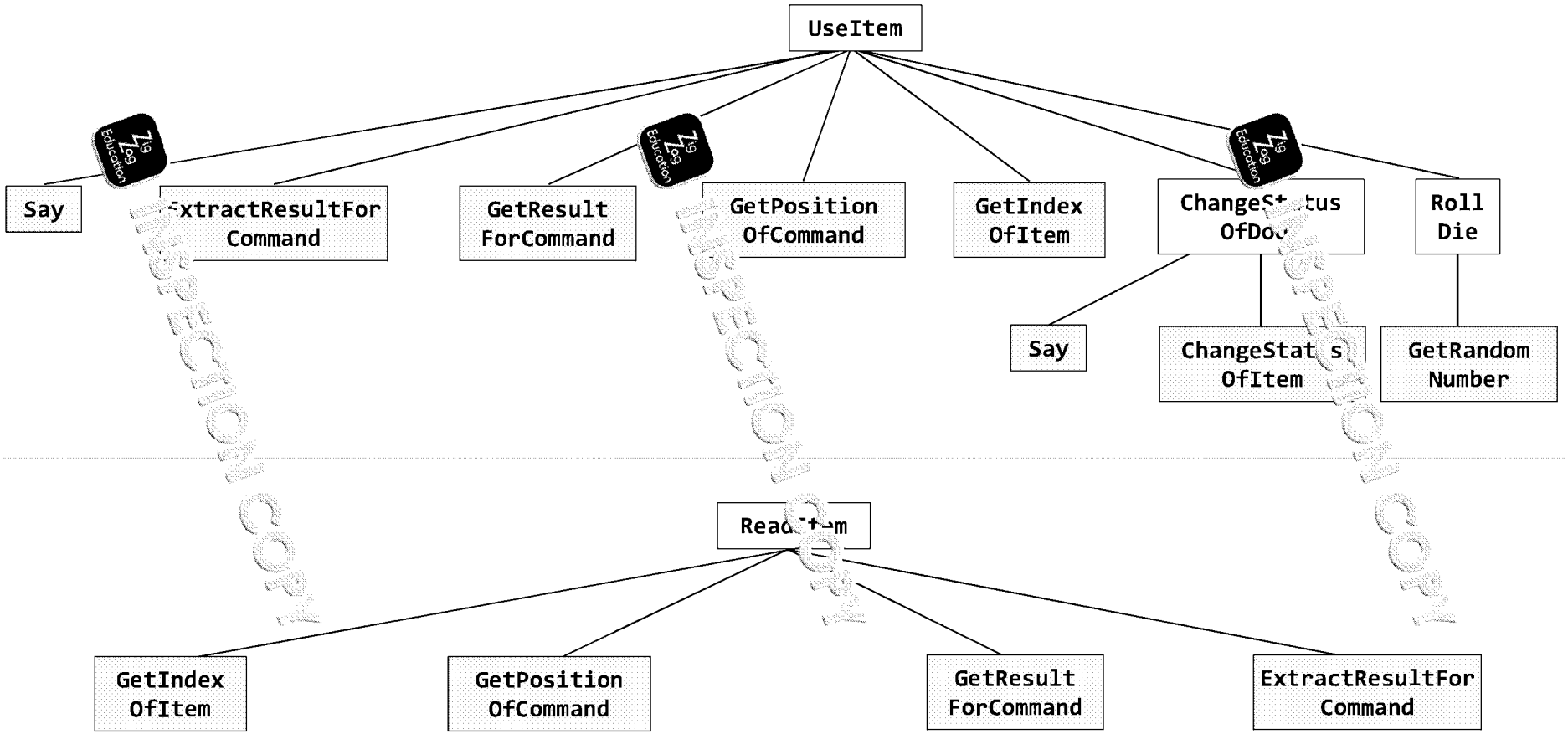
This loop continues until the player quits the game, or gets a particular item that t item is a flag in the provided .gme files). If the player quits the game, a message is given up, and if the player gets the flag (or other winning item), a message is displ

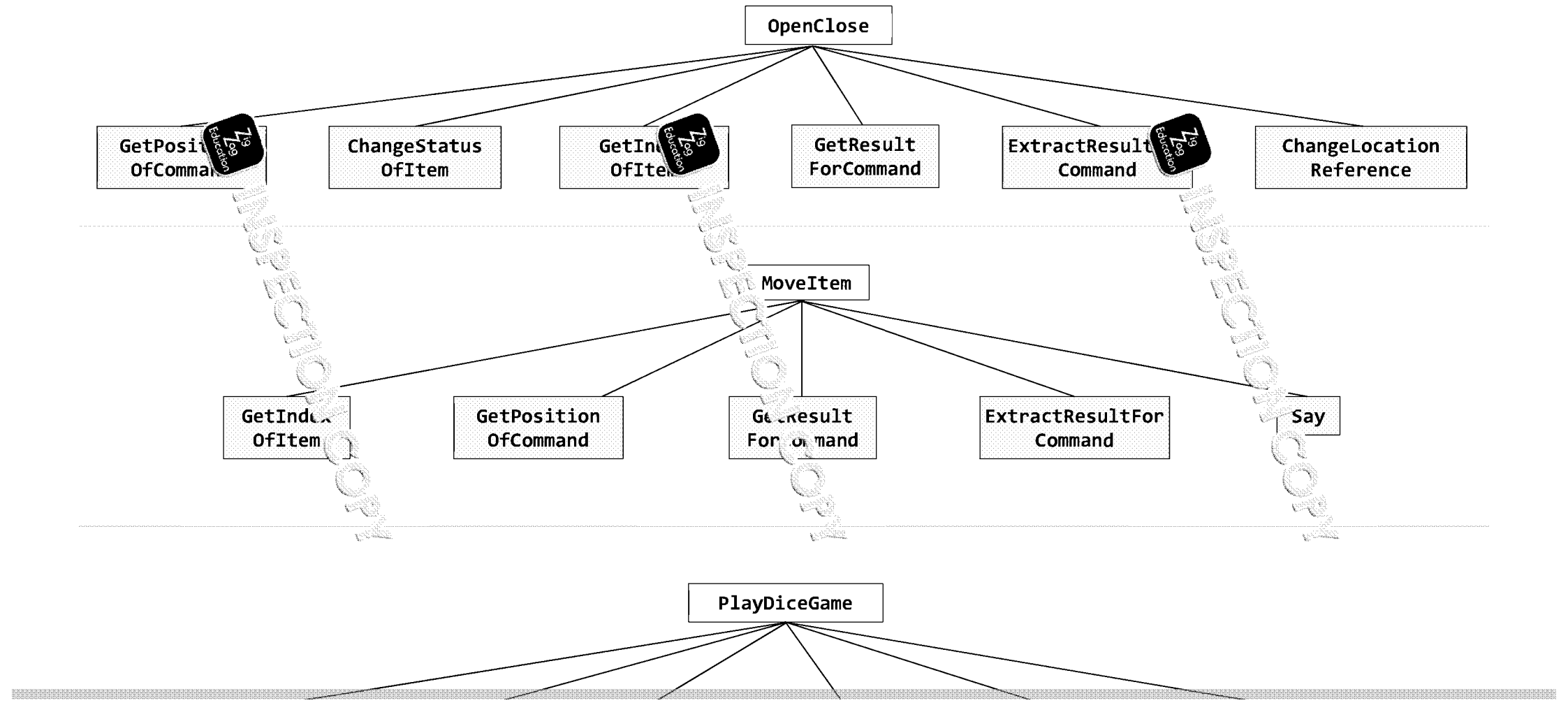
Program Flow Chart



Program Structure Diagrams







Program Subroutines

The program's functions Ⓕ and procedures Ⓟ are described below.

Subroutine	Data	Description
ChangeLocaOfItem Ⓟ	Parameters: Items([Item]) IndexOfItem(int) NewLocation(int) Returns: – Called From: GetItem TakeItemFromOtherCharacter TakeRandomItemFromPlayer Calls: –	1. Creates the ThisItem object of type Ite 2. Sets ThisItem to be equal to the given ite 3. Sets the location of ThisItem to be equal to Location 4. Sets the given item to be equal to ThisItem
ChangeLocation Reference Ⓟ	Parameters: Direction(String) NewLocationReference(int) Places([Place]) IndexOfCurrentLocation(int) Opposite(Boolean) Returns – Called From: OpenClose Calls: –	1. Creates the ThisPlace object of type Place 2. Sets ThisPlace to be equal to the location given by IndexOfCurrentLocation 3. The values of Direction and Opposite are checked to find the direction reference that needs to be changed. If Opposite is *False*, then the direction reference of ThisPlace for the direction Direction is set to NewLocationReference, but if Opposite is *True*, then the direction reference of ThisPlace for the opposite direction to Direction is set to NewLocationReference 4. The location given by IndexOfCurrentLocation is set to be equal to ThisPlace
ChangeStatus OfDoor Ⓟ	Parameters: Items([Item]) CurrentLocation(int) IndexOfItemToLockUnlock(int) IndexOfOtherSideItemToLock… Unlock(int) Returns: –	1. Checks whether the given door or the other side of the given door is in CurrentLocation 2. If either is *True*, the status of the door is "locked", then the statuses of the door and the other side of door are set to "close", and displays a message to say that the item has been unlocked 3. If the status of the door is "close", then the statuses of the door and the other

Subroutine	Data	Description
ChangeStatusOfItem Ⓟ	Parameters: Items([Item]) IndexOfItem(int) NewStatus(String) Returns: – Called From: ChangeStatusOfDo OpenClose Calls: –	1. Creates ThisItem object of type Item 2. Sets ThisItem to be equal to the given item 3. Sets the status of ThisItem to be equal to NewStatus 4. Sets the given item to be equal to Thi
CheckIfDiceGam Possible Ⓕ	Parameters: Items([Item]) Characters([Characte IndexOfPlayerDie(int) IndexOfOtherCharacter(n IndexOfOtherCharacter… Die(int) OtherCharacterName(Strin Returns: (boolean) Called From: PlayDiceGame Calls: GetIndexOfItem	1. Loops through each item in Items and if an t n is in Inventory and has "die" in its name, gets the index of that die a… ets PlayerHasDie to *True* 2. Loops through each character in Characters until a character is found in the same location as the player and has the name O herCharacterName or all of the characters have been checked 3. If a character is found that has the name OtherC aracterName and is in the same location as the player, sets PlayersInS Room to *True* and loops through each item in Items 4. If an item is held by the other character and has "die" in ts name, gets the index of that die and sets OtherCharacterHasDie t ue 5. Returns *True* if all three variables (PlayerHasDie, Pla sInSameRoom and therCharacterHasDie) are *True*, *False* otherwise
DisplayContentsOf ContainerItem Ⓟ	Parameters: Items([Item]) ContainerID(int) Returns: – Called From: Examine Calls: –	1. Displays the message "It contains: " 2. Creates the variable ContainsItem and sets it to *False* 3. Loops through all of the items in Items, and checks whether each item is in the location ContainerID 4. If the item is in the location ContainerID and ContainsItem is *True*, prints " " sets ContainsItem to *True* and prints the name of the item

Subroutine	Data	Description
DisplayGettable ItemsInLocation Ⓟ	Parameters: Items([Item]) CurrentLocation(int) Returns: – Called From: PlayGame Calls: –	1. Loops through each item in Items and checks whether the item is in CurrentLocation and whether its status contains "gettable" 2. If both of these conditions are *True,* appends the name of the item to a string starting "On the floor there is:" 3. If the name of another item has already been added to the string, prints a "," before the name of the next item 4. If the name of any item was appended to the string, a "." is appended to the end
DisplayInventory Ⓟ	Parameters: Items([Item]) Returns: – Called From: Examine Calls: –	1. Prints empty line 2. Prints "You are currently carrying the following items" 3. Loops through every item in Items and if the location of that item is Inventory, prints the name of that item
DisplayOpenCloseMessage Ⓟ	Parameters: ResultOfOpenClose(int) OpenCommand(Boolean) Returns: – Called From: PlayGame Calls: Say	1. If ResultOfOpenClose is greater than *0*, then checks OpenCommand 2. If OpenCommand is *True*, then displays a message to say that the door has been opened, otherwise displays a message to say that the door has been closed 3. If ResultOfOpenClose is *-3*, displays a message to say that the door is locked 4. If ResultOfOpenClose is *-2*, displays a message to say that the door is already in that state 5. If ResultOfOpenClose is *-1*, displays a message to say that the item can't be opened
Examine Ⓟ	Parameters: Items([Item]) Characters([Character]) ItemToExamine(String) CurrentLocation(int)	1. Creates the variable Count and sets it to *0* 2. If ItemToExamine is "Inventory", displays Inventory 3. Otherwise, gets the index of the item

Subroutine	Data	Description
ExtractCommand Ⓕ	Parameters: Instruction(String) Returns: Command(String) Called From: PlayGame Calls: –	1. If Instruction doesn't contain a space, then returns Instruction as both the Command and Instruction variables 2. Otherwise the characters in Instruction are added to the Command variable until a space is found in Instruction 3. Instruction is set to all of the characters in the Instruction that come after the first space 4. Returns Command (Instruction updated having been declared ByRef)
ExtractResultForCommand Ⓟ	Parameters: SubCommand(String) SubCommandParameter(String) ResultForCommand(String) Returns: – Called From: GetItem UseItem ReadItem MoveItem OpenClose Calls: –	1. Loads each character from ResultForCommand into SubCommand in sequence until a "," is read or the end of the string is reached 2. Any remaining characters, up until the next "," or the end of the string, are loaded into SubCommandParameter 3. Does not returns SubCommand and SubCommandParameter, but these parameters were declared ByRef, so they are altered at source (i.e. in the subroutine from which this subroutine was called)
GetIndexOfItem Ⓕ	Parameters: ItemNameToGet(String) ItemIDToGet(int) Items([Item]) Returns: Count(int) Called From: GetItem UseItem ReadItem Examine MoveItem	1. Creates the variable Count and sets it to *0*, and creates the variable StopLoop and sets it to *False* 2. Loops through each item in Items by increasing the Count variable for as long as Count is less than the length of Items and StopLoop is *False* 3. For each item, if ItemNameToGet is equal to the name of that item or itemIDToGet is equal to the ID of that item, sets StopLoop to *True* 4. If ItemNameToGet is not equal to the name of any item in Items and ItemIDToGet is not equal to the ID of any number in Items, returns -1

Subroutine	Data	Description
GetItem Ⓟ	Parameters: Items([Item]) ItemToGet(String) CurrentLocation(int) Returns: – Called From: PlayGame Calls: Say ExtractResultForComma GetResultForCommand GetPositionOfCommand GetIndexOfItem ChangeLocationOfItem	1. Gets the index of the given item 2. If the index of the item couldn't be found, then displays a message to say that the item cannot be found 3. If the item is in Inventory, displays a m to say that the item is already in the Inventory 4. If the item doesn't have the command "get" plays a message to say that the player can't get item 5. If the item is stored in another item and the lo n of the item that contains it is not CurrentLocation, displays a message t y that the item cannot be found 6. If the item is not stored in another item and its loc n is not CurrentLocation, displays a message to say that item cannot be found 7. If steps 2 to 7 above are all *False*, CanGet is set to *Tr* 8. If CanGet is *True*, gets the position of "get" in the item's list of commands and gets the result of using the "get" command on that ite 9. SubCommand of the result is "say", then displays the SubCommandParameter the result 10. SubCommand of the result is "win", displays a message t that the player has won the game and returns *True* 11. If the item's status contains "gettable", then displays the SubCommandParameter of the result, changes the location of the item to Inventory and displays a message to say that the player now has that item
GetPosition OfCommand(S	Parameters: CommandList(String) Command(String)	1. Creates the variable Position and sets it to *0*

Subroutine	Data	Description
`GetRandomNumber` Ⓕ	Parameters: `LowerLimitValue(int)` `UpperLimitValue(int)` Returns: `(int)` Called From: `RollDie` `TakeRandomItemFro…er` Calls: –	1. Returns random integer number between `LowerLimitValue` and `UpperLimitValue`
`GetResultFor Command` Ⓕ	Parameters: `Results(String)` `Position(int)` Returns: `ResultForCommand(Strin…` Called From: `GetItem` `UseItem` `ReadItem` `MoveItem` `OpenClose` `PlayDiceGame` Calls: –	1. Creates the variable `ResultForCommand` a… sets it to "" 2. Loops through each character in `Results` un… as many ";"s have been read as the value of `Position` or the end of the string n… been reached 3. Any remaining characters, up until the next ";" o… end of the string, are loaded into `ResultForCommand` 4. `ResultForCommand` is returned
`Go` Ⓕ	Pa…eters: `You(Character)` `Direction(String)` `CurrentPlace(Place)` Retur… `Moved(Boolean)` Called From: `PlayGame` Calls: –	1. Creates the variable `Moved` and sets it to *True* 2. `…irection` is checked to see if it is a valid direction: "no…", "east", "south", …est", "up" or "down" 3. If a valid direction is given and `CurrentPlace` is adjacent to another place in that direction, the location of `You` is set to point to that other location 4. If a valid direction is given and `CurrentPlace` is not adjacent to another place in that direction, the value of `Moved` is set to *False* 5. If an invalid direction is given, the value of `Moved` is set to *False* 6. If `Moved` is *False*, display the message "You are not able to go in that

Subroutine	Data		Description
LoadGame Ⓕ	Parameters: Returns: Called From: Calls:	Filename(String) Characters([Character]) Items([Item]) Places([Place]) (boolean) Main –	1. If an exception is thrown, returns *False*, otherwise the subroutine runs as follows 2. Creates a DataInputStream object Reader to connect to a binary file 3. Reads the number of characters from the file 4. Creates TempCharacter object of type Character, sets its properties to the values read in by Reader and adds this character to Characters. This is repeated for each character. 5. Reads the number of places from the file 6. Creates TempPlace object of type Place, sets its properties to the values read in by Reader and adds this place to Places. This is repeated for each place. 7. Reads the number of items from the file 8. Creates TempItem object of type Item, sets its properties to the values read in by Reader and adds this item to Items. This is repeated for each item. 9. Returns *True*, assuming no exceptions were thrown
Main Ⓟ	Parameters: Returns: Called From: Calls:	– – – LoadGame PlayGame	1. Creates Items, Characters and Places variables as empty ArrayLists 2. Gets Filename from user input (the program adds ".gme" to the user input, so the file will not be recognised if the user enters the .gme to the end of the filename themselves) 3. Outputs an empty line 4. If call to LoadGame returns *True*, calls PlayGame 5. If call to LoadGame returns *False*, displays a message to say that the game could not be loaded and terminates the program after the user enters input
MoveItem Ⓟ	Parameters: Returns:	Items([Item]) ItemToMove(String) CurrentLocation(int) –	1. Gets the index of the given item 2. If the index of the item is found and is in CurrentLocation, then checks the length of the item's list of commands

Subroutine	Data		Description

`OpenClose` Ⓕ	Parameters:	`Open(boolean)`
		`Items([Item])`
		`Places([Place])`
		`ItemToOpenClose(String)`
		`CurrentLocation(int)`
			1. If `Open` is *True*, creates the variable `Command` and sets it to "open"
			2. If `Open` is *False*, creates the variable `Command` and sets it to "close"
			3. Loops through each item in `Items` and if the name of the item is equal to `ItemToOpenClose`, the item is in `CurrentLocation` and the string of that item's commands is at least 4 characters long, checks the status of the item
			4. If the status of the item is equal to `Command`, returns *-2*
			5. If the status of the item is "locked", returns *-3*
			6. Otherwise, creates the variable `Position` and sets it to be equal to the position of `Command` in the door's list of commands, creates the variable `ResultForCommand` and sets it to be equal to the result of using `Command` on the door
			7. Changes the status of the door
			8. Sets `ActionWorked` to *True*
			9. Loops through each place to change the appropriate location references in `CurrentLocation` and the location that is linked to `CurrentLocation` by the door
			10. Changes the status of the other side of the door
			11. If `ActionWorked` is *False*, returns *-1*
			12. If `ActionWorked` is *True*, returns `DirectionChange` cast as an integer

	Returns:	`DirectionChange(int)`	
	Called From:	`PlayGame`	

	Calls:	`GetPositionOfCommand`
		`GetResultForCommand`
		`ExtractResultForCommand`
		`ChangeStatusOfItem`
		`GetIndexOfItem`
		`ChangeLocationReference`	

`PlayDiceGame` Ⓟ	Parameters:	`Characters([Character])`
		`Items([Item])`
		`OtherCharacterName(String)`
			1. Calls `CheckIfDiceGamePossible` to see if a dice game can be played
			2. If a dice game can't be played, displays the message "You can't play a dice game."
			3. If a dice game can be played, gets the position of "use" in the player's die's list of commands and gets the result of using the "use" command on that die to get the result of the dice roll
			4. Displays a message to show what the player rolled
			5. Gets the position of "use" in the other character's die's list of commands and gets the

	Returns:	–	
	Called From:	`PlayGame`	

	Calls:	`CheckIfDiceGamePossible`
		`GetPositionOfCommand`
		`GetResultForCommand`	

Subroutine	Data	Description
PlayGame Ⓟ	Parameters: Characters([Character]) Items([Item]) Places([Place]) Returns: – Called From: Main Calls: DisplayGettableItemsInLocation GetInstruction ExtractCommand GetItem UseItem Go ReadItem Examine OpenClose DisplayOpenCloseMessage MoveItem PlayDiceGame Say	1. If the game has ended, waits for user input and then terminates, otherwise the subroutine runs the following loop 2. If the player has moved to a new location, prints two empty lines, the description of the location and the gettable items at the location 3. Gets input from the user, converts it to lowercase and splits it into a Command and an Instruction 4. If Command is "get", tries to get the given item 5. If Command is "use", tries to use the given item 6. If Command is "go", tries to move the character in the given direction 7. If Command is "read", tries to read the given item 8. If Command is "examine", tries to examine the given item, character or Inventory 9. If Command is "open", tries to open the given item 10. If Command is "close", tries to close the given item If Command is "move", tries to move the given item 12. If Command is "say", calls displays the Instruction of the input 13. If Command is "playdice", tries to play a dice game with the given character 14. If Command is "quit", displays a message to say that the player has given up and ends the game 15. If Command is anything else, displays a message to say that the command isn't recognised
ReadItem Ⓟ	Parameters: Items([Item]) ItemToRead(String) CurrentLocation(int) Returns: – Called From: PlayGame	1. Gets the index of the given item 2. If the index of the item couldn't be found, then displays a message to say that the item cannot be found 3. If the item doesn't have "read" in its list of commands, displays a message to say that the item cannot be read

Subroutine	Data		Description

`RollDie` Ⓕ	Parameters:
	
	Returns:
	Called From:
	
	Calls:
		`Lower(String)`
		`Upper(String)`
		`(int)`
		`UseItem`
		`PlayDiceGame`
		`GetRandomNumber`
			1. Creates `LowerLimitValue` variable, and sets it to *0*
			2. If the `Lower` string is made up entirely of numeric digits, as determined by a call to `IsNumeric`, sets `LowerLimitValue` to `Lower`(cast as an integer)
			3. Otherwise, sets `LowerLimitValue` to be equ[illegible]e `int` equivalent of user input until `LowerLimitValue` is an integer from *1* t[illegible]
			4. Creates `UpperLimitValue` variable, and sets it to *0*
			5. If the `Upper` string is made up entirely of numeric d[illegible]ts, as determined by a call to `IsNumeric`, sets `UpperLimitValue` to `Upper` (ca[illegible] an integer)
			6. Otherwise, sets `UpperLimitValue` to be equal to th[illegible]nt equivalent of user input until `UpperLimitValue` is an integer from *1* to *6*
			7. Returns a random integer number between `LowerLim[illegible]tValue` and `[illegible]pperLimitValue`

`Say` Ⓟ	Parameters:
	Returns:
	Called From:
	
	Calls:
		`Speech(String)`
		–
		`PlayGame`
		`GetItem`
		`UseItem`
		`MoveItem`
		–
			1. Prints empty line
			2. Prints `Speech`
			3. Prints empty line

`TakeItemFrom OtherCharacter` Ⓟ	Parameters:
	
	Returns:
	Called From:
	Calls:
		`Items([Item])`
		`OtherCharacterID(int)`
		–
		`PlayDiceGame`
		`ChangeLocationOfItem`
			1. Loops through all of the items in `Items`, and checks whether each item is held by the given character
			2. If the item is held by that character, the name of that item is added to a list
			3. The message "Which item do you want to take? They have:" is displayed, followed by each item in the list, with each item separated by "," and the list ending with ".", although the last item in the list is not displayed unless it is the only item in the list

Subroutine	Data	Description

`TakeRandomItemFromPlayer` Ⓟ	Parameters: `Items([Item])`
	`OtherCharacterID(int)`
	Returns: –
	Called From: `PlayDiceGame`
	Calls: `GetRandomItem`
	`ChangeLocationOfItem`
		1. Loops through each item in `Items` and checks whether each item is in `Inventory`
		2. If the item is in `Inventory`, the position of that item in `Items` is added to a list
		3. Gets a random number from *0* to the highest position in the list
		4. Displays a message to say which item has been [illegible] (the item who's position in the list matches the random number)
		5. Moves the item from `Inventory` to the other character

`UseItem` Ⓟ	Parameters: `Items([Item])`
	`ItemToUse(String)`
	`CurrentLocation(int)`
	`Places([Place])`
	Returns: –
	Called From: `PlayGame`
	Calls: `Say`
	`ExtractResultForCommand`
	`GetResultForCommand`
	`GetPositionOfCommand`
	`GetIndexOfItem`
	`ChangeStatusOfDoor`
	`RollDie`
		1. Gets the index of the given item
		2. If the index of the item can be found, then checks whether the item is in `Inventory` or is in `CurrentLocation` and has the status "usable"
		3. If either is *True*, gets the position of "use" in the item's list of commands and gets the result of using the "use" command on that item
		4. If `SubCommand` of the result is "say", then displays the `SubCommandParameter` of the result
		5. If `SubCommand` of the result is "lockunlock", then changes the status of the relevant doors
		6. If `SubCommand` of the result is "roll", then displays a message to say what number was rolled
		7. If the index of item couldn't be found, then displays a message to say that the item can't be used

Program Structures

The structures defined in the program, and their attributes, are described below.

Structure	Description
Place	Structure that defines the properties of each of the game's locations
Character	Structure that defines the properties of each of the game's characters
Item	Structure that defines the properties of each of the game's items

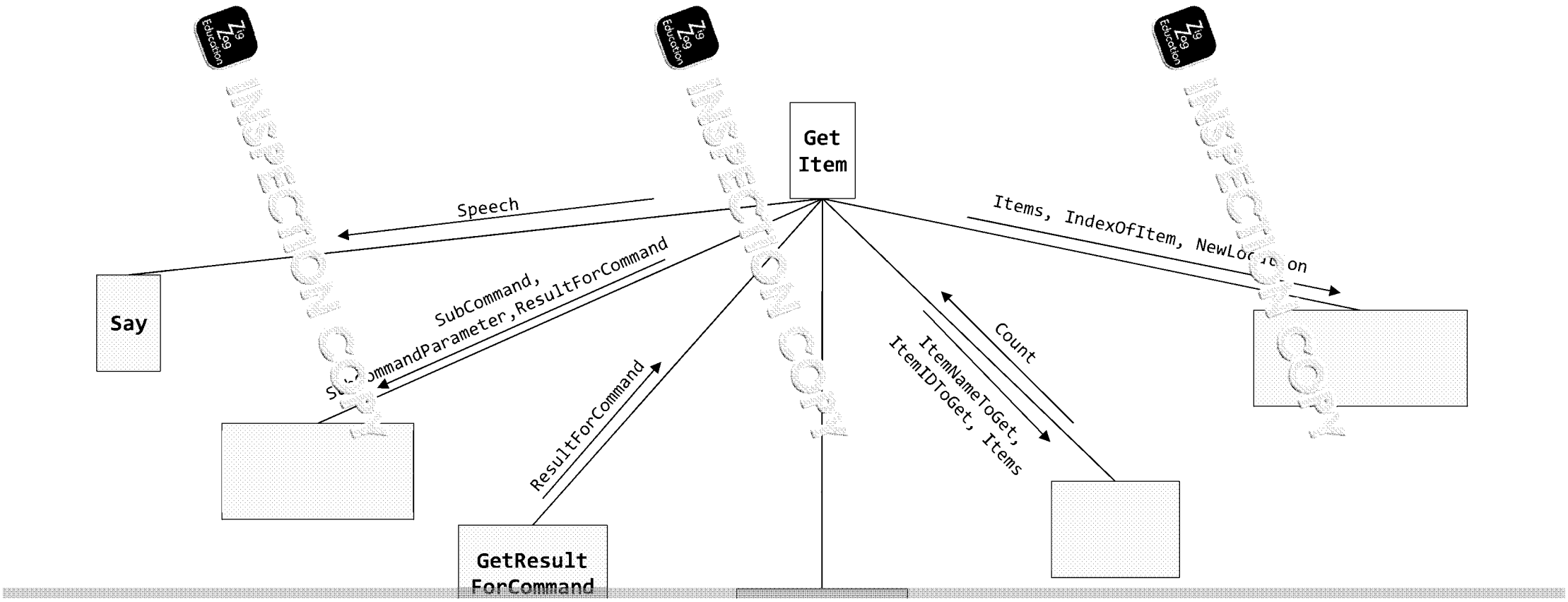
Place Attributes

Attribute	Type	Default Value	Description
Description	String	""	The text description of the place, which is shown to the player whenever they move to this place
ID	Integer	0	The ID of this place, which allows it to be referenced by the program or other objects
North	Integer	0	The ID of the location to the north of this place; if this is 0, then there is no location to the north of the place
East	Integer	0	The ID of the location to the east of this place; if this is 0, then there is no location to the east of the place
South	Integer	0	The ID of the location to the south of this place; if this is 0, then there is no location to the south of the place
West	Integer	0	The ID of the location to the west of this place; if this is 0, then there is no location to the west of the place
Up	Integer	0	The ID of the location above this place; if this is 0, then there is no location above the place
Down	Integer	0	The ID of the location below this place; if this is 0, then there is no location below the place

Character Attributes

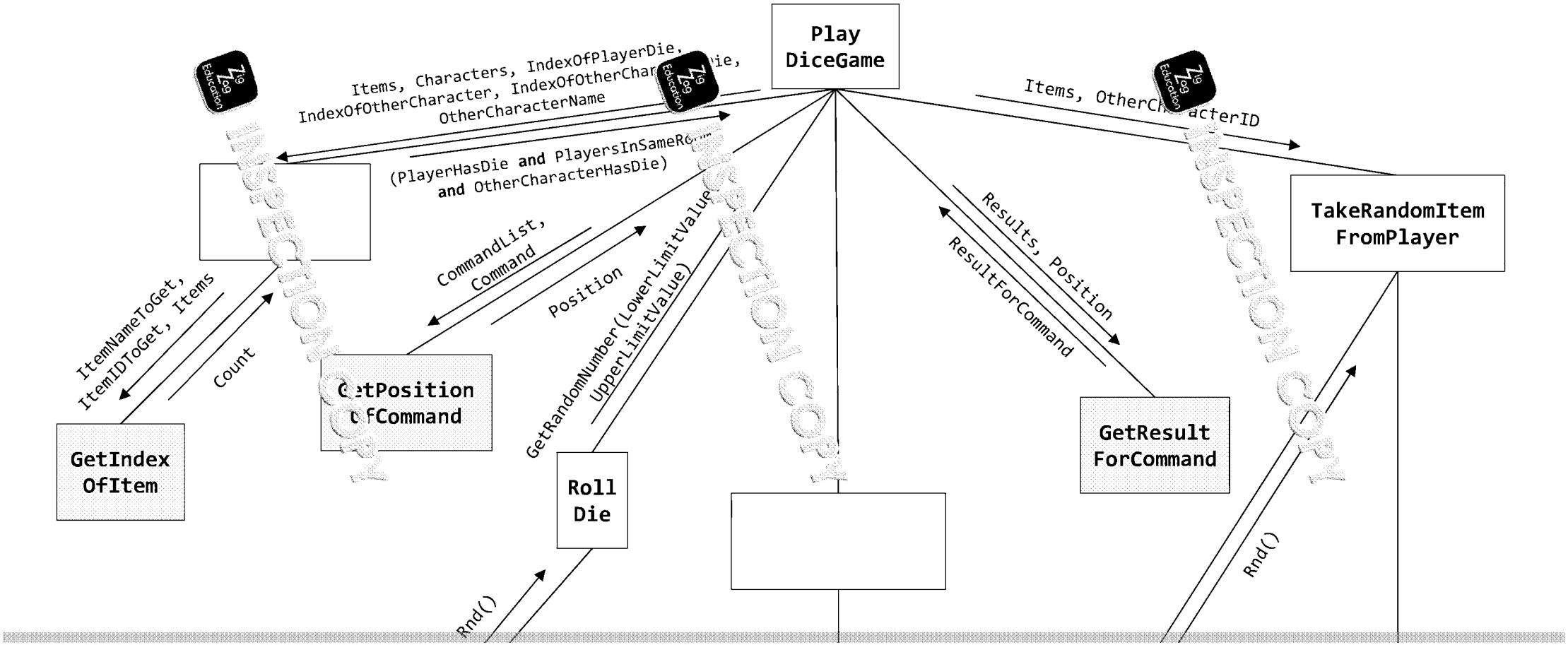
Attribute	Type	Default Value	Description
Name	String	""	The name of the character, which is displayed to the player and used by the player to interact with this character
Description	String	""	The text description of the character, which is shown to the player whenever they examine this character
ID	Integer	0	The ID of this character, which allows it to be referenced by the program or other objects
CurrentLocation	Integer	0	The ID of the place where the character is; if this is 0, then the character is not in a place

Structure Diagram Activity 1 (GetItem)



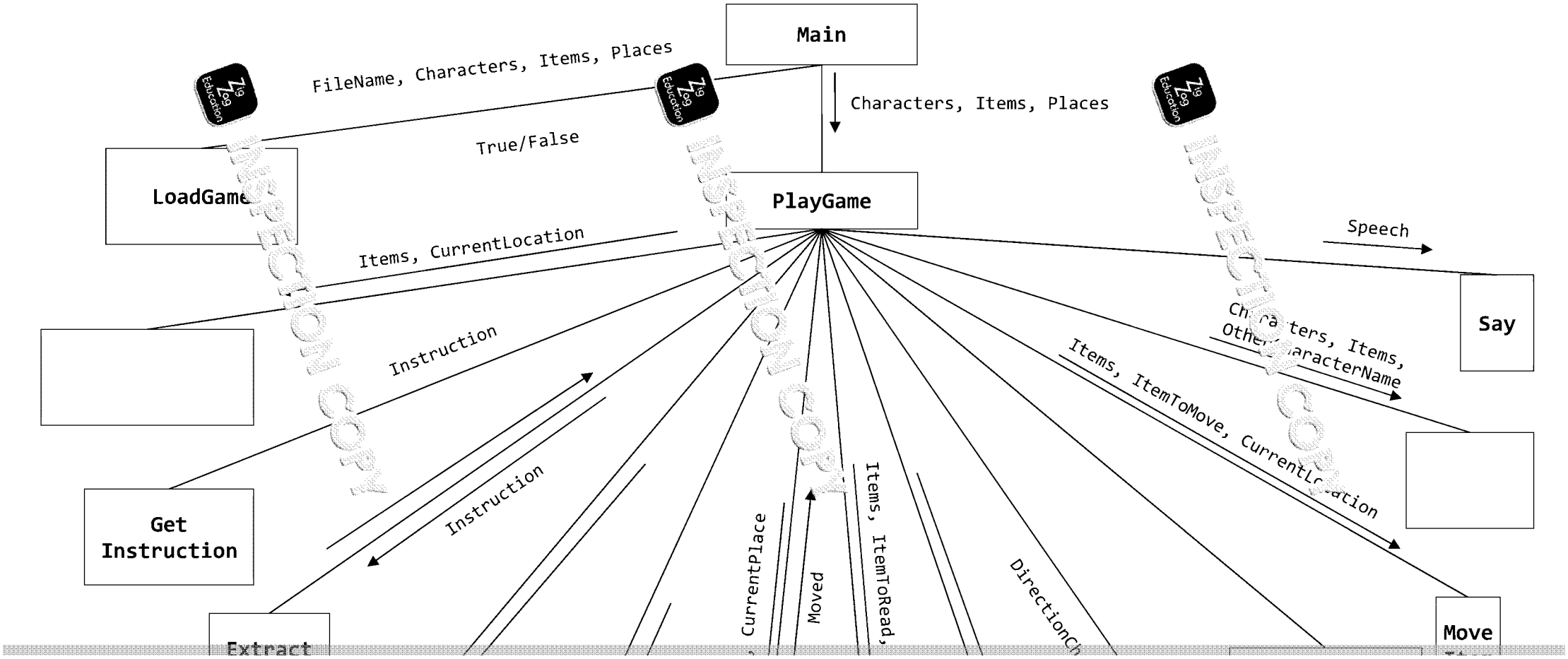
Structure Diagram Activity 2 (PlayDiceGame)

AQA Text Adventures



Structure Diagram Activity 3 (Main)

AQA Text Adventures



AQA Text Adventures

Written Questions (VB.NET)

These questions refer to the preliminary material and require you to load the skeleton program and make any additional programming.

1. State the name of an identifier for:
   a) Two integer variables in the function `LoadGame` [2]
   b) Three subroutines that each return a string [3]
   c) Two built-in functions used in the function `ExtractCommand` [2]
   d) A structure that has exactly four attributes [1]
   e) Two constants [2]
2. Explain the purpose of the following line in the `GetInstruction` subroutine:

```
Instruction = Console.ReadLine.ToLower
```

3. Explain the role of the variable `Moved` in the function `Go`. [3]
4. Describe the purpose of the `Select Case` structure used throughout the program.
5. Using any example from the Skeleton Program, describe what it meant by the term *parameter*.
6. Describe in detail the role of the built-in function `IsNumeric` as used in the function `LoadGame`.
7. Describe the role of the variable `TempCharacter` in the function `LoadGame`.
8. The procedure `DisplayInventory` accepts an `ArrayList` as a parameter. Explain why it is better to use an `ArrayList` in `DisplayInventory` instead of an array. [4]
9. Explain why the `LowerLimitValue` in the function `RollDie` is initialised to 1. [2]
10. Explain the purpose of each of the two while loops in the function `ExtractCommand`. [4]
11. Describe the purpose of the `for` loop in the function `DisplayInventory`. [3]
12. Describe the changes that would need to be made so that the game would output 'You are carrying nothing' if the player attempts to examine an empty inventory. [4]
13. Describe the changes that would need to be made to the function `GetInstruction` so that it ignores if two or more characters has been entered before `GetInstruction` returns. [4]
14. Describe the effect of a file not being found during the function `LoadGame`. [3]

- END OF TEST -

AQA Text Adventures

Written Questions (VB.NET)

These questions refer to the preliminary material and require you to load the skeleton program. You do not require any additional programming.

1. State the name of an identifier:

   a) Two integer variables in the function `LoadGame` [2]

   .....

   b) Three subroutines that each return a string [3]

   .....

   c) Two built-in functions used in the function `ExtractCommand` [2]

   .....

   d) A structure that has exactly four attributes [1]

   .....

   e) Two constants [2]

   .....

2. Explain the purpose of the following line in the `GetInstruction` subroutine:

```
Instruction = Console.ReadLine.ToLower
```

.....

.....

.....

.....

3. Explain the role of the variable `Moved` in the function `Go`. [3]

.....

.....

.....

.....

.....

4. Describe the purpose of the `Select Case` structure used throughout the p

5. Using any example from the Skeleton Program, describe what it meant by the

6. Describe in detail the role of the library function `IsNumeric` as used in the f

7. Describe the role of the variable `TempCharacter` in the function `LoadGam`

8. The procedure `DisplayInventory` accepts an `ArrayList` as a paramet using an `ArrayList` in `DisplayInventory` instead of an array. [4]

9. Explain why the `LowerLimitValue` in the function `RollDie` is initialised

.....

.....

.....

.....

.....

10. Explain the purpose of each of the two while loops in the function `ExtractC`

While loop 1

.....

.....

.....

.....

While loop 2

.....

.....

.....

11. Describe the purpose of the `for` loop in the function `DisplayInventory`. [2]

.....

.....

.....

.....

.....

.....

12. Describe the changes that would need to be made so that the game would output 'nothing' if the player attempts to examine an empty inventory. [4]

.....

.....

.....

.....

.....

.....

13. Describe the changes that would need to be made to the function `GetInst` of two or more characters has been entered before `GetInstruction` retu

.....

.....

.....

.....

.....

.....

14. Describe the effect of a file not being found during the function `LoadGame`.

.....

.....

.....

.....

- END OF TEST -

AQA Text Adventures

Programming Tasks (VB.NET)

The following require you to open the skeleton program and make modifications

Task 1

This question refers to `PlayGame` as well as a new subroutine `ShowHelp`.

Currently, users are not offered any help regarding valid commands. If a user ent
told '**Sorry, you don't know how to…**'

Add a new subroutine called `ShowHelp`, which should display the following text

You can enter one of the following commands:
go, get, use, examine, say, quit, read, move, open, close and playdice.

Modify the subroutine `PlayGame` so that if a user enters the command 'help', th

Evidence you need to provide:

- Your amended SOURCE CODE for `PlayGame`
- Your SOURCE CODE for the new `ShowHelp` subroutine
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'
  - Type 'help' at the first prompt

Task 2

This question refers to `Examine` and `PlayGame`.

Currently, users are able to examine their inventory, as well as objects within a ro

Modify `Examine` so that if the user has entered the text 'examine room', the des
be re-displayed. A call to `DisplayGettableItemsInLocation` should also

You should pass `Places ArrayList` from `PlayGame` to `Examine`.

Evidence you need to provide:

- Your amended SOURCE CODE for `Examine`
- Your amended SOURCE CODE for `PlayGame`
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'
  - Type 'examine room' at the first prompt

Task 3

This question refers to `PlayGame`.

Currently, if the user types 'quit' at the prompt, the game ends without any furth

Modify `PlayGame` so that an entry of 'quit' triggers an output of '**Are you sure?**
Entering either 'y' or 'yes' in any combination of upp rcase should termi
'**You decide to give up, try again another ti st.**' being displayed.

Any other input should ret yer to the game and the main prompt.

Evidence y d to provide:

- Your amended SOURCE CODE for `PlayGame`
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'
  - Type 'quit' at the first prompt
  - Type 'no' in response to 'Are you sure'
  - Type 'quit' at the next prompt
  - Type 'yes' in response to 'Are you sure'
  - Press enter after resulting output

Task 4

This question refers to `Say` and `Main`.

Currently, output that is displayed as a resu a to `Say` is displayed in a sing
displayed, multiple calls to `Say` h e nade.

Create an ad na entation of `Say` that receives a string array instead o
content of ement of the array on separate lines, maintaining the line break

In order to test your modified version of `Say`, you will need to add code to the `M`
Immediately before the call to `PlayGame`, create an array consisting of the strin
"AQATEXTADVENTURES" and pass this array to `Say`.

Evidence you need to provide:

- Your amended SOURCE CODE for `Say`
- Your amended SOURCE CODE for `Main`
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'

Task 5

This question refers to `DisplayInventory`.

Currently, when the inventory is examined items are displayed in the order in wh
binary file.

Modify the `DisplayInventory` subroutine, so th[illegible] a bubble sort to so
alphabetical order based on their `Name` att[illegible] before displaying the list of ite
algorithm yourself, and should n[illegible] library sorting functions.

Evidence y[illegible]d [illegible]ide:

- You[illegible]ded SOURCE CODE for `DisplayInventory`
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'
  - Type 'examine inventory' at the first prompt
  - Type 'get red die' at the second prompt
  - Type 'examine inventory' at the third prompt

Task 6

This question refers to `GetItem` and `TakeItemFromOtherCharacter` as w

Currently, the player's inventory can contain as many items as can be found.

Create an integer constant called `MaxInventor[illegible]` and [illegible] it to 4. When an atte
player with four or more items already in [illegible]ventory, the message '**Inventory f**
The check for a full inventory should take place before any existing selection state
messages should be [illegible] within `GetItem` other than 'Inventory full'.

This check s[illegible]also be applied to `TakeItemFromOtherCharacter` so tha
after winning a dice game, the message 'Inventory full' will be displayed if the pla

Evidence you need to provide:

- Your amended SOURCE CODE for `GetItem`
- Your amended SOURCE CODE for `TakeItemFromOtherCharacter`
- Your SOURCE CODE for the new `MaxInventory` constant
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'
  - Type 'get torch' at the first subsequent prompt
  - Type 'get red die' at the second prompt
  - Type 'examine inventory' at the third prom[illegible]
- One screenshot showing the output that [illegible]ts [illegible]m the following:
  - Run the program and loa[illegible]ile 'flag2'
  - Type 'playdice g[illegible]a[illegible]first subsequent prompt
  - At each [illegible]e two prompts, type '6' (repeat the previous ste
  - Typ[illegible] the next prompt

Task 7

This question refers to `PlayGame` as well as a new subroutine `DropItem`.

Currently, items can be picked up but not dropped.

Create a new subroutine called `DropItem` which accepts three parameters:

- An `ArrayList` called 'items', which will contain all items in the game
- A string called 'itemToDrop' which will be the name of the item leaving th
- An integer called 'currentLocation', which will be the ID of the player's loc

`DropItem` should check that the item exists and the player is carrying the item. If not, the text '**You are not carrying a _____**' should be displayed (with the item name dis Otherwise, the item should be placed into the room with the text '**_____ dropped** dropped.

You should modify `PlayGame` to call `DropItem` if a command of 'drop' is enter

Evidence you need to provide:

- Your amended SOURCE CODE for `PlayGame`
- Your SOURCE CODE for the new `DropItem` subroutine
- One screenshot showing the output that results from the following:
  - Run the program and load the file 'flag1'
  - Type 'drop flask' at the first prompt
  - Type 'drop flask' again at the second prompt
  - Type 'examine inventory' at the third prompt

Task 8

This question refers to `Main`.

Currently, if the user enters the name of a non-existent file, the program ends wit different file name. Additionally, the file extension '.gme' is always appended to t included the extension themselves. This would result in a file not being found.

Modify the user prompt in `Main` to read '**Enter filename or enter Q to quit**'. If th should end. If the user enters a valid file name, the file should load as normal. Ot continually re-display until the user has entered 'Q' or a file name that successfull

If the user enters a file name that contains the string '.gme', the program should n

You should only make changes to `Main`.

Evidence you need to provide:

- Your amended SOURCE CODE for `Main`
- One screenshot showing the output that results from the following:
  - Type 'flag3' when prompted for a file name
  - At the next prompt, type 'flag2.gme'

Task 9

This question relates to `PlayGame` as well as a new subroutine `DisplayChara`

In order for a developer to work effectively on existing code, it is helpful for them content of a data structure. Create a new subroutine called `DisplayCharact` parameter in the form of `Characters ArrayList`.

When `DisplayCharacters` is called it s ild display four column headers in

- ID (followed by th)
- NA lc , eight spaces)
- DES ON (followed by 53 spaces)
- CURRENTLOCATION

Beneath these columns, all characters within the game should be displayed, with the attributes `ID`, `Name`, `Description` and `CurrentLocation` should each column header.

Modify `PlayGame` so that if 'debugchar' is entered during gameplay, a call to `Di` made.

Evidence you need to provide:

- Your amended SOURCE CODE for `PlayGame`
- Your SOURCE CODE for the new `DisplayCharact rs` subroutine
- One screenshot showing the output that res m the following:
  - Type 'flag1' when prompted f a ile ne
  - At the next prompt ' ugchar'

Task 10

This question relates to `PlayGame` as well as a new subroutine `FillVessel`.

In the flag1 game, there is a room below the starting location that contains a barr
subroutine called `FillVessel` that requires the following parameters

- The `Characters ArrayList`
- The `Items ArrayList`
- The `Places ArrayList`
- A string called

`FillVess` uld check whether the player and the barrel are in the same lo
whose `Status` attribute contains the text 'container' and does not contain the t
be in the player's inventory.

If any of these conditions are not met, the message '**You cannot do that**' should be
item's `Name` attribute should have the text ' **of water**' appended to it, and the text
displayed, with the vessel appended to the end.

If the vessel's name attribute already contains the text 'of water', the text '**Already**

Add a call to `FillVessel` in `PlayGame` that will occur if the player enters the

Evidence you need to provide:

- Your amended SOURCE CODE for `PlayGame`
- Your SOURCE CODE for the new `FillVessel` subroutine
- One screenshot showing the output results from the following:
  - Type 'flag1' when prompted for a file name
  - At the next prompt, type 'open trapdoor'
  - At the next prompt, type 'go down'
  - At the next prompt, type 'fill flask'
  - At the next prompt, type 'fill apple'
  - At the next prompt, type 'fill flask of water'
  - At the next prompt, type 'examine inventory'

Task 11

This question relates to `Go`, `PlayGame` and a new class, `History`.

Create a new class called `History`, with two attributes:

- An integer called 'Pointer', that is initially set to *-1*
- A five-element array called 'Locations'

`History` will use a stack, to allow players to backtrack through previous locations. You should use an implementation of the `Locations` array, which will be subject to two subroutines:

- `Push` should require an integer as a parameter. If the stack is full, all existing items move down by 1 element (so the content of element 0 is lost), and the value of the parameter is added to the top. If the stack is not full, the pointer attribute should be incremented and the value of the parameter added to the element indicated by the `Pointer`. The `Push` subroutine does not return a value.

- `Pop` requires no parameters but returns an integer. If the stack is empty, it returns -1. Otherwise the value in the element indicated by `Pointer` should be returned and the pointer decremented.

In `PlayGame`, create an instance of the `History` class called 'past'. When the player enters a movement command, this instance should be passed to the `Go` subroutine as an additional parameter.

In `Go`, modify the parameters to accept the instance of the `History` class. As a result, players will add their location onto the stack before they move to a new one. If they enter the instruction 'go back' they will move to their previous location using a call to the `Pop` subroutine of `History`. If the call to `Pop` returns -1, display the text '**You cannot go back**'.

Evidence you need to upload:

- Your SOURCE CODE for the new `History` class
- Your amended SOURCE CODE for `PlayGame`
- Your amended SOURCE CODE for `Go`
- One screenshot showing the output that results from the following:
  - Type 'flag2' when prompted for a file name
  - At the next prompt, type 'go east'
  - At the next prompt, type 'go back'
  - At the next prompt, type 'go back' a second time

Task 12

This question relates to the `Character` structure, the `PlayGame` subroutine, t
new subroutine `EatItem`.

Modify the `Character` structure so that each character in the game has a `Heal`

Modify the `LoadGame` subroutine so that each chara[illegible]tial `Health` value

Create a new subroutine called `EatIte[illegible]` requires as parameters a `Chara`
player's character, the `Items[illegible]ra[illegible]list` and a string called `ItemToEat`. If t
name of an item in th[illegible]'s inventory, and that item's `Status` attribute cont
player's `He[illegible]` property by 5 and remove the eaten item from the `ArrayList`
met, the tex[illegible]d cannot eat that' should be displayed. If an item is successfully
current player's health.

Modify `PlayGame` so that the player's health is displayed immediately after the
`DisplayGettableItemsInLocation`. Include a call in `PlayGame` to `Eat`
instruction beginning with 'eat'.

Evidence you need to provide:

- Your amended SOURCE CODE for the `Character` structure
- Your amended SOURCE CODE for `LoadGame`
- Your SOURCE CODE for the new `EatItem` subroutine
- Your amended SOURCE CODE for `PlayGame`
- One screenshot showing the output that results from the following:
  - Type 'flag1' when prompted for a file name
  - At the next prompt, type '[illegible]op'
  - At the next prom[illegible]'eat apple' a second time

Task 13

This question relates to `TakeItemFromOtherCharacter`.

Currently, if you win a dice game against an opponent and mis-type the item you
told that you do not receive an item and you are given no opportunity to re-type.

Modify `TakeItemFromOtherCharacter` so that if you request an item that
'**They don't have a _____ – try again**' is displayed (with _____ replaced by the iter
to take).

The player is then asked again which item they want to take, with the available ite
happen until the player has entered an item that the oth[illegible]yer possesses, at w
player's inventory as normal.

Evidence you need to provide:

- Your a[illegible]CE CODE for the `TakeItemFromOtherCharact`
- On[illegible]ns[illegible]ut showing the output that results from the following:
  - [illegible]ype 'flag2' when prompted for a file name
  - At the next prompt, type 'playdice guard'
  - At each of the next two prompts, type '6' (repeat the previous ste
  - When asked to choose an item, type 'box'
  - At the next prompt, type 'jar'

Task 14

This question relates to `CheckIfDiceGamePossible`.

Currently, if a dice game is not possible, the program outputs the message '**You c** whether the player has no die, the opponent has no die or the player and the opp

Modify `CheckIfDiceGamePossible` so that the following messages are outp

1. If the player has no die, the output should read '**Player has no die**'
2. If the player and opponent are not in the same room, the output shou ____ being replaced by the name of the other character)
3. If the opponent has no die, the output should be '**_____ has no die**' (with of the other character)

In all cases, these messages should still be followed by the message '**You can't pla** more than one of these conditions are met, only one should be displayed, with co which they are numbered above. So if the player has no die, the program will not or that the opponent has a die. If the opponent is not present, no check will be m

No code should be modified besides the content of `CheckIfDiceGamePossi`

Evidence you need to provide:

- Your amended SOURCE CODE for `CheckIfDiceGamePossible`
- One screenshot showing the output that results from the following:
  - Type 'flag1' when prompted for a file name
  - At the next prompt, type 'playdice guard'
  - At the next prompt, type 'get red die'
  - At the next prompt, type 'playdice guard'

Task 15

This question relates to `DisplayInventory`.

Currently, when the user enters 'examine inventory', a list of the names of the ite displayed.

Modify `DisplayInventory` so that descriptions are displayed alongside the n displayed in round brackets. Additionally, the opening brackets of the description regardless of variations in an item's name.

Evidence you need to provide:

- Your amended SOURCE CODE for `DisplayInventory`
- One screenshot showing the output that results from the following:
  - Type 'flag1' when prompted for a file name
  - At the next prompt, type 'examine inventory'

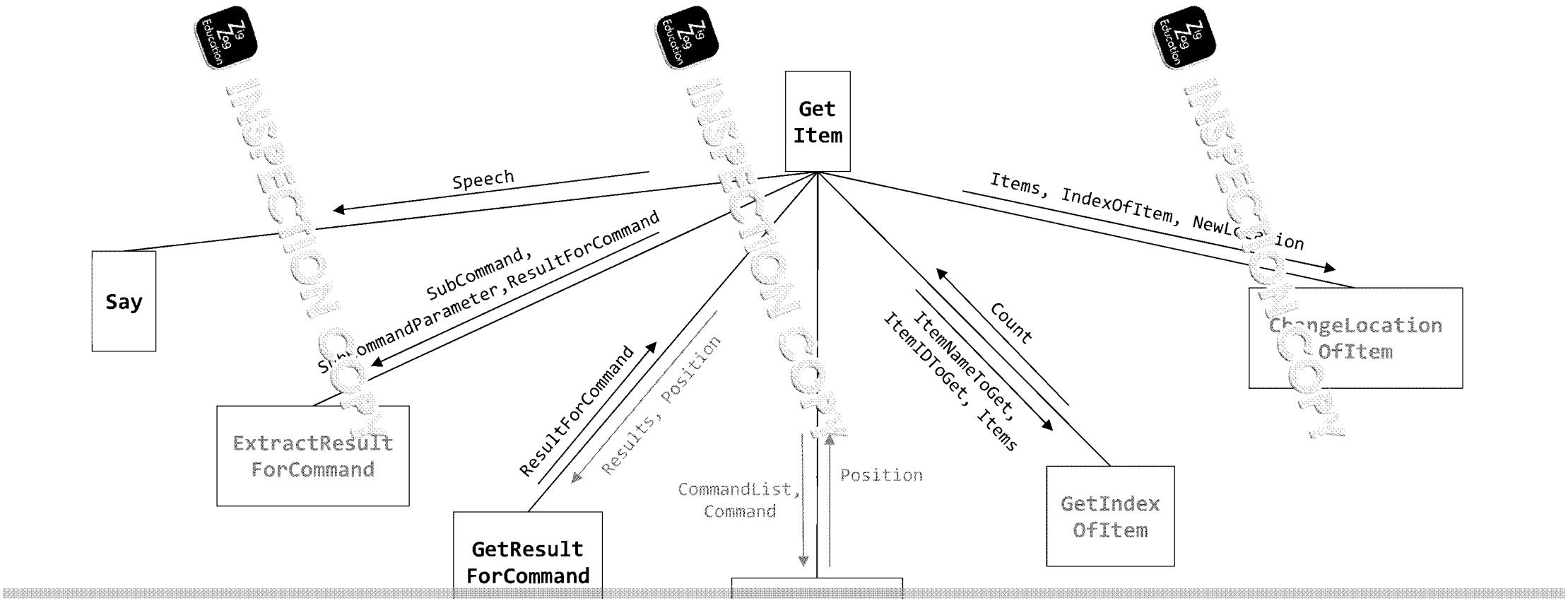
AQA Text Adventures

Programming Tasks - Extension (VB.NET)

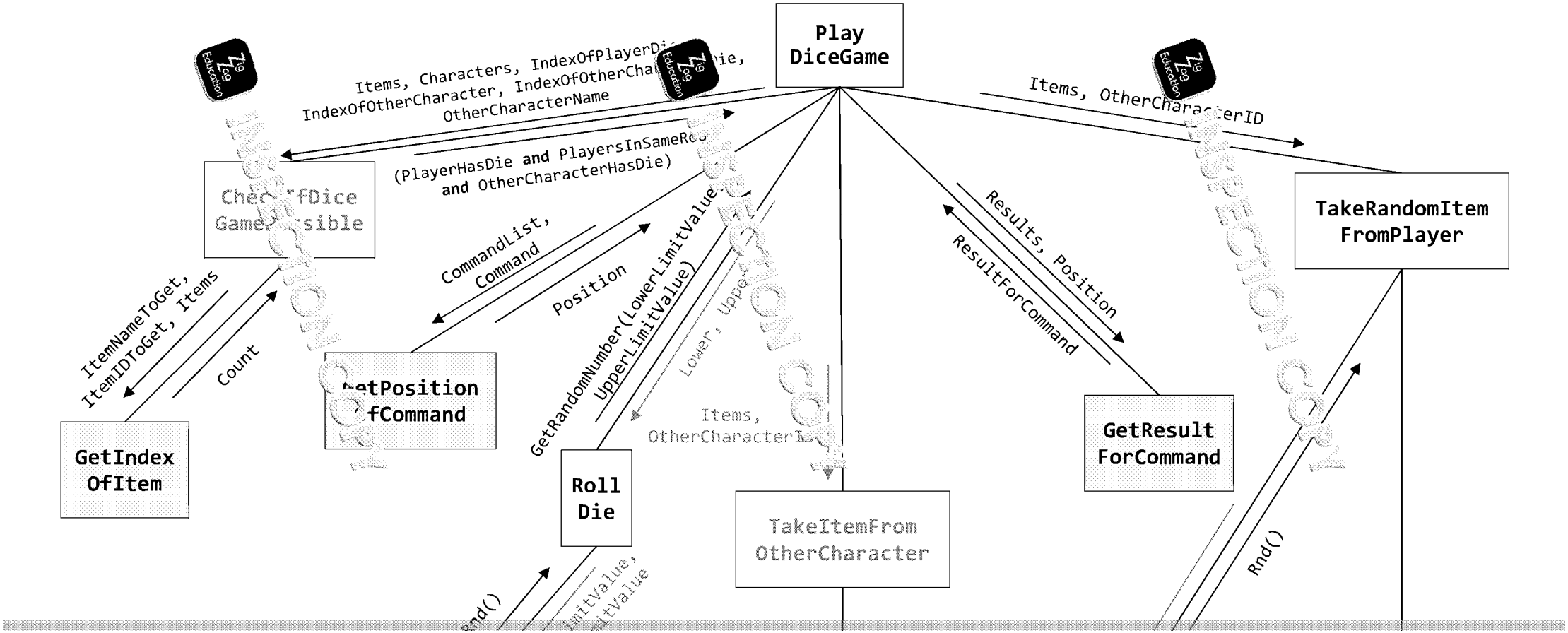
These challenges are presented without solutions, and offer opportunities to further explore and develop the program. They are listed here in approximate order of difficulty, beginning with the easiest.

1. Add a 'restart' command that will begin the current game again from the start
2. Modify `UseItem` to incorporate a win condition that involves using an object in the flag
3. Randomise the location of the guard
4. In the original program, the guard's blue die is not presented as an option for the player to take after winning a dice game unless it is the guard's only remaining item. Modify `TakeItemFrom` so that the player can only take the items that are listed after they have won a dice game
5. Modify `Go` so that the player will automatically try to unlock any locked doors in their way
6. Incorporate an additional command 'look', which could be used instead of 'go' to view the current location, assuming 'go' wouldn't have been blocked by a closed door
7. Attempting to pick up an item by using part of its name (such as 'die' instead of 'blue die') successfully picks up the item
8. If the player is carrying two dice, both can be rolled and the total used again
9. In the starting room for 'flag1.gme', the trapdoor is hidden by a rug but can be moved by the rug. Edit the game so that the rug must be moved before the trapdoor can be opened
10. Edit `PlayDiceGame` so that characters cannot lose a die in a dice game (so that they possess nothing but dice)
11. If you enter 'use truncheon' in the same room as the guard, the guard is knocked out and items can be taken without the need to play dice
12. Incorporate a 'save' command for players to save their progress, which could be loaded
13. An option to create a game file from the console, saving the new game file

Structure Diagram 1 (GetItem) Solution

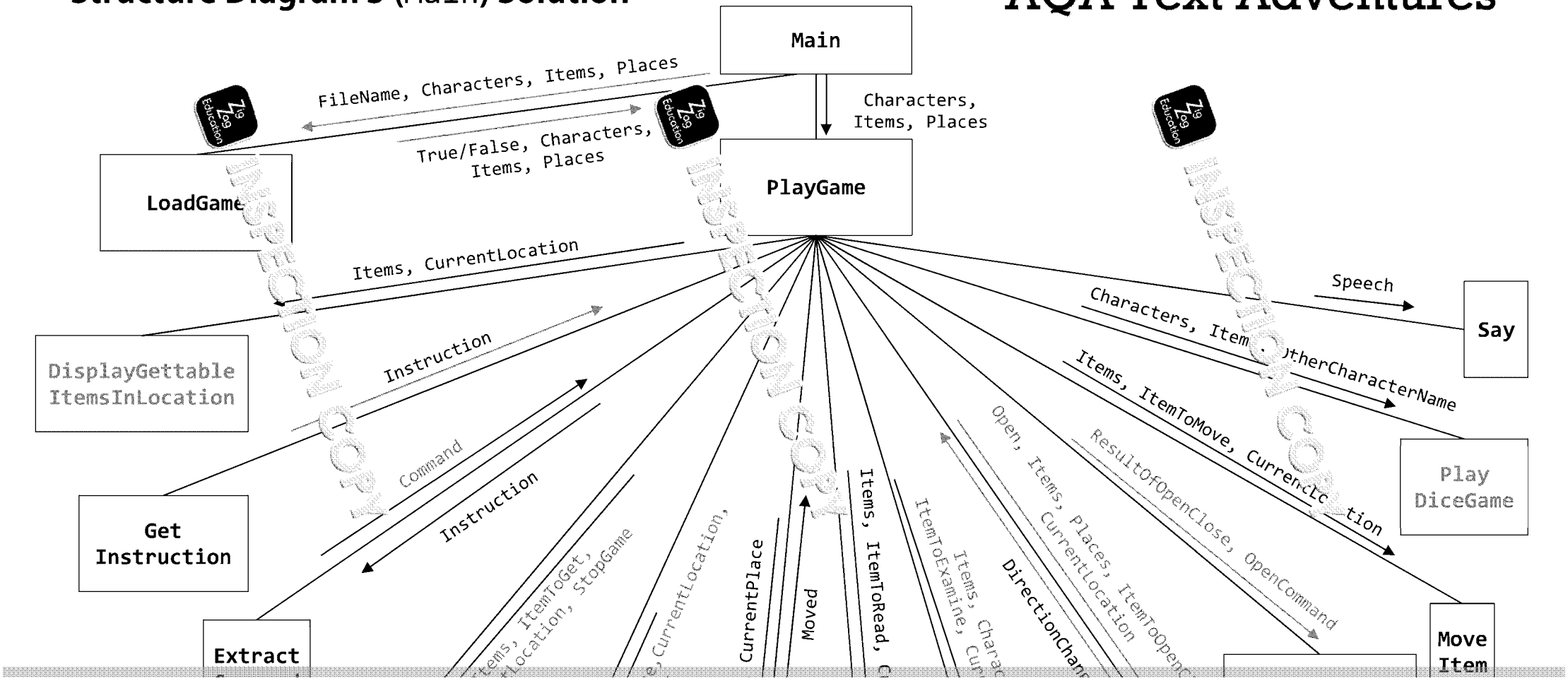


Structure Diagram 2 (PlayDiceGame) Solution



Structure Diagram 3 (Main) Solution

AQA Text Adventures



AQA Text Adventures

Written Questions (VB.NET) - Solutions

Q	Answer/Evidence
1a	2 marks: • `NoOfCharacters` • `NoOfPlaces` • `Count` • `[illegible]`
1b	3 marks: • `GetInstruction` • `ExtractCommand` • `GetResultForCommand`
1c	2 marks: • `Contains` • `Remove`
1d	`Character`
1e	2 marks • `Inventory` • `MinimumIDForItem` • `IDDifferenceForObjectInTwoLocations`
2	2 marks: • Convert the user input to lowercase • Because subsequent comparisons will be made with lowercase strings / to prevent from being compared with otherwise identical lowercase strings
3	3 marks: • [illegible] whether the player has successfully moved • If the player attempts an invalid move, it is set to *False* • If it remains *False*, a message (you are not able to go in that direction) appears
4	2 marks: • Compares user input with (available) commands... • ...to allow for the correct subroutine(s) to be called / instruction(s) to be executed

5	2 marks:
	• (Process of) connecting/joining two (or more) strings together
	• Any instance of a & used between two strings, e.g.
	  ○ `& Command & "."`
	  ○ `Command &= Instruction(0)`
	
	(Does not need to include entire line)

6	5 marks: • Accepts a string as a parameter • Attempts to convert the [illegible] to an integer • Returns *True* if successful • Returns *False* if unsuccessful • Checks to see if Lower/Upper are numeric
7	3 marks: • Temporarily holds an instance of `Character` • Attributes are set by reading data from the file • Once attributes are set it is added to the `Characters ArrayList`

Q	Answer/Guidance
8	Two marks for advantages, two for linking advantages to `AQATextAdventure` in • `ArrayLists` are dynamic / have no fixed max size... • ...so an item will always be able to be added to the inventory • `ArrayLists` can be more easily added to in the middle of the data struc • ...so items could more easily be stored in a specified order (e.g. alphabetic
9	3 marks: • It may or may not be set to a different value in the following `If` clause • If it is not set to a value, then any following `While` loop would cause an er `LowerLimitVa` was checked • Initialising `` to 0 prevents this error from occurring
10	3 mar `While` loop): • First `While` loop checks for there being no space in the first character • If there is not a space, the first character is added to `Command` • The first character is (also) discarded from `Instruction` / the `Instruc` a substring from the second character onward • Once the loop is complete, the `Instruction` variable contains everythi onward and the `Command` variable contains the contents of the initial stri before the first space 2 marks (second `While` loop): • The second `While` loop checks for a space (in the first character of what • If there is a space, it is discarded • This loop removes leading spaces from `Instruction`
11	2 marks: • Loops through each item in the `Items ArrayList` • Displays each item whose location is `Inventory`
12	4 marks: • Reference made to changing re `DisplayInventory` • Selection statement before 'You are currently carrying' output • Selection st checks if any items have the location of `Inventory` • ems are in `Inventory`, display alternative message, othe age Alternative implementations potentially worth full marks would include the selectic opposite order (i.e. checking that it is not empty) or pre-selection-statement instruc items in the `ArrayList` then compare this with zero.
13	3 marks: • Assignment statement for `Instruction` variable to be placed within a l • Check to be made within the loop that the length of the string is at least 2 • Loop to continue until that check returns *True* / length is at least 2 charact
14	2 marks: • Execution would move to the `Catch` block / an exception would be throw • The function `LoadGame` would return a value of *False*
TOTAL MARKS	

AQA Text Adventures

Programming Tasks (VB.NET) - Suggested Solutions and Mark Scheme

Note that the following are recommended solutions and not an exhaustive list of a task. The marking guidance should be used as a guide only. Discretion should be alternative solutions are giv

Task 1

1 mark correct declaration of `ShowHelp`

1 mark valid call to `Console.WriteLine` or `Say` with text indicated in questi wording **must match**, including the line break, although any alternative m two lines is acceptable.

```
Sub ShowHelp()
    Console.WriteLine("You can enter one of the following commands:
    Console.WriteLine("go, get, use, examine, say, quit, read, mov
End Sub
```

1 mark additional `Case` statement in an appropriate locati within `PlayGame`

```
    ReadItem(Items, Instruc  acters(0).CurrentLocatio
Case "help"
    ShowHelp()
Case "exa
```

1 mark scre ot showing output that matches text added to `ShowHelp`

```
Enter filename> flag1



You are in a room with blue walls.  There is a green d
yellow door in the western wall and a cupboard door in
is a Persian rug on the floor.
On the floor there is: torch, red die.

> help
You can enter one of the following commands:
go, get, use, examine, say, quit, re  move, open, cl

> _
```

Task 2

1 mark additional argument in call to `Examine` from `PlayGame`

```
    ReadItem(Items, Instruction, Characters(0).CurrentLocatio
Case "examine"
    Examine(Items, Characters, Places, Instruction, Characte
Case "open"
```

1 mark additional parameter added to `Examine` declaration

1 mark addition to selection structure (`ElseIf`) to accommodate additional ec
c[illegible]luded in `If` clause, `ElseIf` clause or `Else` clause)

1 mark current location's description displayed in appropriate part of selection

1 mark valid call to `DisplayGettableItems` in appropriate part of selectio

1 mark examining inventory and other objects remains unaffected

```
Sub Examine(ByVal Items As ArrayList, ByVal Characters As ArrayL
            ByVal ItemToExamine As String, ByVal CurrentLocation
    Dim Count As Integer = 0
    If ItemToExamine = "inventory" Then
        DisplayInventory(Items)
    ElseIf ItemToExamine = "room" Then
        Console.WriteLine(Places(Characters(0).CurrentLocation -
        DisplayGettableItemsInLocation(Items, Characters(0).Curre
    Else
```

1 mark screenshot showing repeated ro[illegible]cription after user enters 'examin

```
Enter filen[illegible]gl

You are in a room with blue walls.  There is a green
yellow door in the western wall and a cupboard door i
is a Persian rug on the floor.
On the floor there is: torch, red die.

> examine room
You are in a room with blue walls.  There is a green
yellow door in the western wall and a cupboard door i
is a Persian rug on the floor.
On the floor there is: torch, red die.

>
```

Task 3

1 mark code within `Case "quit"` to display prompt 'Are you sure? (y/n)' (**R.** a

1 mark user input is stored in a string variable

1 mark selection clause that catches either 'y' or 'yes' (**I.** case)

1 mark selection clause catches both 'y' and 'yes' in any combination of cases, ei
lower or by checking for ev[illegible]os[illegible] combination (including yES, yEs, e

1 mark output and [illegible] assignment inside the selection structure and break

```
Ca[illegible]it"
    Console.WriteLine("Are you sure? (y/n)")
    Dim response As String = Console.ReadLine
    If response.ToUpper = "Y" Or response.ToUpper = "YES" The
        Say("You decide to give up, try again another time.")
        StopGame = True
    End If
Case Else
```

1 mark output showing 'no' resulting in a prompt, followed by 'yes' resulting in t

```
> quit
Are you sure? (y/n)
no

> quit
Are you sure? (y/n)
yes

You decide t[illegible]p, try again another time.
```

Task 4

1 mark valid declaration of `Say` with string array parameter

1 mark suitable loop to iterate through the array

1 mark output of each element within the loop

1 mark blank lines output both before and after the loop

```
Sub Say(ByVal Speech() As String)
    Console.WriteLine()
    For x = 0 To Speech.Length
        Console.WriteLine(Speech(x - 1))
    Next
    Console.WriteLine()
End Sub
```

1 mark valid declaration and instantiation of a string array

1 mark passing array to `Say`

```
If LoadGame(Filename, Characters, Items, Places) Then
    Say(New String() {"WELCOME", "TO", "AQATEXTADVENTURES"})
    PlayGame(Characters, Items, Places)
Else
```

1 mark screenshot showing each element of the array on a separate line before t

Task 5

1 mark declaring temporary `ArrayList` of inventory items

1 mark loop to add all inventory items to the temporary `ArrayList`

```
Dim inventoryItems As New ArrayList

For Each Thing In Items
    If Thing.Location = Inventory me
        inventoryItems.Add(
    End If
Next
```

1 mark loo[illegible] through each pass of the sort operation (`x` in this example

1 mark loo[illegible]erate through each pair for individual comparisons (`y` in this exa

1 mark neither iteration would cause a comparison to be made beyond the `Arr`

1 mark selection structure that attempts to compare adjacent values, even if syn

1 mark selection structure accesses the name attribute of an item for the compa

1 mark selection structure is valid, comparing appropriate name values and addr

1 mark temporary variable used to store one of the items to be switched inside s

1 mark pair of items switched inside selection structure

```
Dim temp As Item
For x = 0 To inventoryItems.Count - 2
    For y = 0 To inventoryItems.Count - 2
        If inventoryItems(y).name > inventoryItems(y + 1).nam
            temp = inventoryItems(y)
            inventoryItems(y) = inven[illegible]e s(y + 1)
            inventoryItems(y [illegible]) t[illegible]
        End If
    Next
Next
```

1 mark dis[illegible] items in the order they are found in the sorted array

```
Console.WriteLine()
Console.WriteLine("You are currently carrying the following i
For Each Thing In inventoryItems
    Console.WriteLine(Thing.Name)
Next
Console.WriteLine()
```

1 mark screenshot showing flag1 being loaded, 'examine inventory' being entere
flask, jar) being displayed in alphabetical order. After 'get red die' and 'ex
the inventory should again be displayed in alphabetical order (apple, flas

```
You are in a room with blue walls.  There is a green door
On the floor there is: torch, red die.

> examine inventory

You are currently carrying the following items:
apple
flask
jar

>
Yo[illegible]ot that now.

> e[illegible]e inventory

You are currently carrying the following items:
apple
flask
jar
red die
```

Task 6

1 mark constant correctly declared and set to 4

```
Const Inventory As Integer = 1001
Const MinimumIDForItem As Integer = 2001
Const IDDifferenceForObjectInTwoLocations As Integer = 10000
Const MaxInventory As Integer = 4
```

1 mark variable to count inventory items

1 mark iteration through Items ArrayList

1 mark selection or equivalent to identify items in the inventory

1 mark incrementing variable inside selection statement

```
Dim InventorySize As Integer = 0
For Each Thing In Items
    If Thing.Location = Inventory Then
        InventorySize += 1
    End If
Next
```

1 mark selection statement to check that InventorySize >= constant (**R.** ==)

1 mark 'Inventory full' displayed only under correct conditions when the player t

1 mark remainder of selection structure would not be executed if inventory is fu

```
If InventorySize >= MaxInventory Then
    Console.WriteLine("Inventory full")
ElseIf Ind... -1 Then
```

1 mark implementing InventorySize check in both GetItem and TakeIt

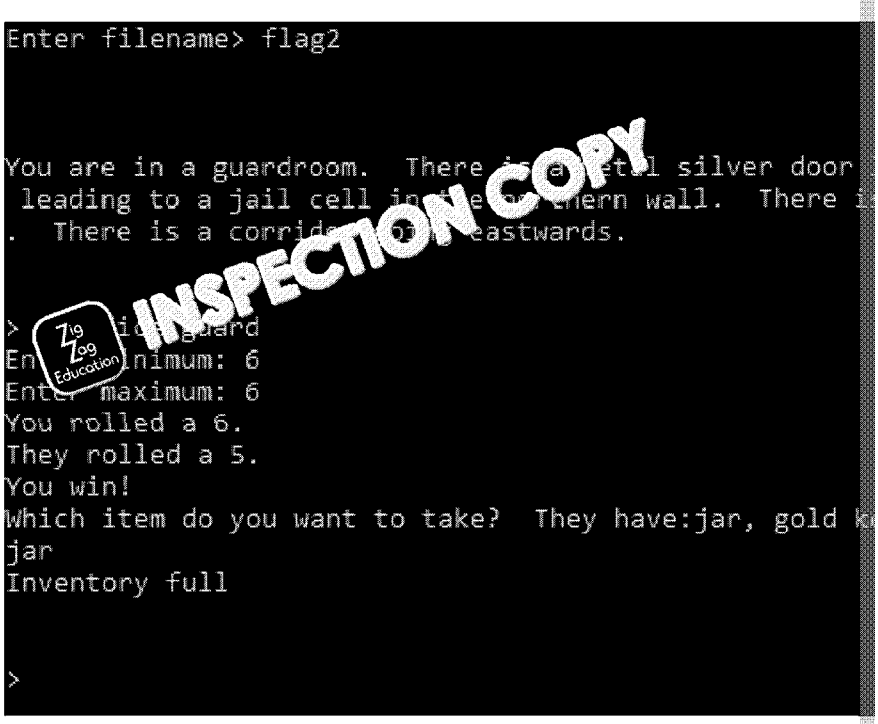
1 mark 'Inventory full' displayed only under correct conditions when the player tri

```
Dim InventorySize As Integer = 0
For Each Thing In Items
    If Thing.Location = Inventory Then
        InventorySize += 1
    End If
Next

If ListOfNamesOfItemsInInventory.Contains(ChosenItem) And Inver
    Console.WriteLine("You have that now.")
    Dim Pos As Integer = ListOfNamesOfIte... Inventory.IndexOf(
    ChangeLocationOfItem(Items, Lis... esOfItemsInInventory
ElseIf InventorySize >= Max... or ... hen
    Console.WriteLine("... ... full")
Else
    Cons... Line("They don't have that item, so you don't
E...
```

1 mark first screenshot shows torch being picked up, red die not being picked up accordingly

1 mark second screenshot shows dice game being played and won, jar not being accordingly



Task 7

1 mark `DropItem` correctly declared with three parameters

1 mark call to `GetIndexOfItem` (**A**. a correctly-implemented loop that has the

1 mark selection statement to check that item exists at all

1 mark selection statement to check whether the item being dropped is in the in

1 mark if item is not in inventory, correct message displayed (**R**. if non-existent it same error message)

1 mark if item in inventory, location is changed to `CurrentLocation`

1 mark if item is in inventory, correct message displayed

```
Sub DropItem(ByVal items As ArrayList, ByVal itemToDrop As String, By
    Dim indexOfItem As Integer = GetIndexOfItem(itemToDrop, -1, item
    If indexOfItem > -1 Then
        If Not (items(indexOfItem).Location = Inventory) Then
            Console.WriteLine("You are not carrying a " & itemToDrop
        Else
            ChangeLocationOfItem(items, indexOfItem, currentLocation
            Console.WriteLine(itemToDrop & " dropped")
        End If
    Else
        Console.WriteLine("You are not carrying a " & itemToDrop)
    End If
End Sub
```

1 mark selection structure of `PlayGame` incorporates 'drop' command

1 mark parameters correctly passed to `DropItem`

```
Case "examine"
    Examine(Items, Characters, Instruction, Characters(0).Curr
Case "drop"
    DropItem(Items, Instruction, Characters(0).CurrentLocatio
```

1 mark screenshot shows 'flask dropped' and 'You are not carrying a flask' as wel

```
On the floor there is: torch, red die.

> drop flask
flask dropped

> drop flask
You are not carrying a flask

> examine inventory

You are currently carrying the following items:
apple
ja
>
```

Task 8

1 mark variable to track whether loop should repeat, or equivalent

1 mark loop repeats until a file name has successfully loaded

1 mark prompt amended to include quit option

1 mark selection (or inclusion in a loop) to address entry of 'Q'

1 mark execution always terminates when 'Q' is entered

1 mark selection statement to detect '.gme' within file name

1 mark '.gme' added only if it is not already within the string

1 mark successfully loading a file ensures loop would not run again

```
Dim isLoaded As Boolean = False
While isLoaded = False
    Console.Write("Enter filename> ")
    Filename = Console.ReadLine
    If Filename = "Q" Then
        Exit Sub
    End If
    If Not (Filename.Contains(".gme")) Then
        Filename = Filename & ".gme"
    End If
    Console.WriteLine()
    If LoadGame(Filename, Characters, Items, Places) Then
        isLoaded = True
        PlayGame(Characters, Items, Places)
    Else
        Console.WriteLine("Unable to load game.")
    End If
End While
```

1 mark 'flag3' causes prompt to repeat (I. extra line breaks)

1 mark 'flag2.gme' causes game to begin

```
Enter filename> flag3

Unable to load game.
Enter filename> flag2.gme



You are in a guardroom.  There is a
in the chair is a guard.  There is

>
```

Task 9

1 mark correct declaration of `DisplayCharacters`, including parameter

```
Sub DisplayCharacters(ByVal Characters As ArrayList)
```

1 mark header row contains names of all fields as appropri

1 mark spacing between headers is correct

```
Dim Header As S
Heade = "
He      eader & "NAME          "
Hea     Header & "DESCRIPTION".PadRight(64)
Header = Header & "CURRENTLOCATION"

Console.WriteLine(Header)
```

1 mark loop to iterate through each element of the `Characters ArrayList`

1 mark each entry (even if inaccurate) will display on a separate line

1 mark all four attributes displayed on a line, with no additional text

1 mark spacing will be correct regardless of size of a field

```
For Each c In Characters
    Dim line As String = ""
    line = line & CType(c.ID, Stri    ).ght(5)
    line = line & c.Name.Pad    (
    line = line & c.D     .PadRight(64)
    line = line &     .CurrLocation

    o  .WriteLine(line)
Ne
```

1 mark selection clause adapted to include checking for input of 'debugchar'

1 mark valid call to `DisplayCharacters` from `PlayGame`

```
Case "say"
    Say(Instruction)
Case "debugchar"
    DisplayCharacters(Characters)
```

1 mark two characters displayed, lined up correctly in columns in the correct orc

```
> debugchar
ID   NAME        DESCRIPTION
1001 me          You think
1002 guard       The g      t 180cm tall.  He is wearing a r

> _
```

Task 10

1 mark valid declaration

```
Sub FillVessel(ByVal Characters As ArrayList, ByVal Items As
               ByVal Places As ArrayList, ByVal vessel As Str
```

1 mark variable, or equivalent, to store whether the item is a container

1 mark variable, or equivalent, to store whether the item is large *(equivalence in 'isContainer' is not true only if the item is both a container and not conditions to a single variable, and all applicable marks can still be*

1 mark variable, or equivalent, to store whether the item is in the player's invent

1 mark variable, or equivalent, to store whether the player is in the same locatio

1 mark loop to examine items in turn (**A**. multiple loops, calls to `GetIndexOfI`

1 mark selection statement to check whether the vessel is a container

1 mark selection statement to check whether the vessel is (not) large

1 mark selection statement to check whether the vessel is in the player's invento

1 mark selection statement to check whether the player and the barrel are in the

1 mark all variables are set to the correct values under all circumstances

```
For Each i In Items
    If vessel = i.Name And i.Status.Contains("container") And
                           Not i.Status.contains("large") Then
        isContainer = True
        fillable = i
        ref = Items.IndexOf(i)
    End If
    If vessel = i.Name And i.Location = Inventory Then
        inInventory = True
    End If
    If i.Name = "barrel" And i.Location = Characters(0).Curren
        byBarrel = True
    End If
Next
```

1 mark for ensuring that the program will not crash even if the fillable item is null

1 mark check for presence of 'of water' within the name attribute

1 mark 'already full' displayed under the correct circumstances

1 mark selection statement to check that all conditions are met

1 mark inside selection statement, append ' of water' to the vessel item's name (identifier of the item or call to `GetIndexOfItem`; this example code uses a loop above and re-inserts into the `ArrayList` having kept a note of the

1 mark output, comprising the vessel appended to 'You have filled the'

1 mark 'You cannot do that' output if vessel has not been filled or if a null pointe

```
If IsNothing(fillable) Then
    Console.WriteLine("You cannot do that")
Else
    If fillable.Name.Contains("of water") Then
        Console.WriteLine("Already full")
    ElseIf isContainer And inInventory And byBarrel And Not fillable
        fillable.Name = vessel & " of water"
        Items(ref) = fillable
        Console.WriteLine("You
    Else
        Console.Wri
    End If
En
```

1 mark selection structure in `PlayGame` adjusted to include 'fill' command

1 mark valid call to `FillVessel` at valid point in selection structure

```
Case "read"
    ReadItem(Items, Instruction, Characters(0).CurrentLocatio
Case "fill"
    FillVessel(Characters, Items, Places, Instruction)
```

1 mark input of 'fill flask' displays 'You have filled the flask' and 'fill apple' displa

1 mark input of 'fill flask of water' displays 'Already full' and 'examine inventory'

```
> go down

You are in a cellar
> fill flask
You have filled the flask

> fill apple
You cannot do that

> fill flask of water
Already full

> examine inventory

You are currently carrying the following items:
flask of water
apple
jar

>
```

Task 11

1 mark `History` class correctly declared

1 mark integer variable `Pointer` and integer array `Locations` correctly decla

1 mark `Pointer` and `Locations` initialised to -1 and 5 elements respectively

```
Class History

    Dim Pointer As Integer = -1
    Dim Locations(4) As Integer
```

1 mark `Push` cla[illegible]ctly, including integer parameter

1 mark sele[illegible]tatement to check whether `Pointer` is < 4 or <= 3

1 mark if so, `Pointer` is incremented…

1 mark …and the parameter is correctly inserted into the array

1 mark if not, contents of elements 4, 3, 2 and 1 are shifted down, with contents

1 mark …and the parameter is inserted into element 4 of the array

```
Sub Push(ByVal NewLocation As Integer)
    If Pointer < 4 Then
        Pointer += 1
        locations(Pointer) = NewLocation
    Else
        For x = 0 To 3
            locations(x) = locations(x + 1)
        Next
        locations(4) = NewLocation
    End If
End Sub
```

1 mark `Pop` correctly declared with no parameters and an integer return

1 mark che[illegible] `Pointer` value of -1

1 mark if the `Pointer` is set to -1, then -1 should be the return value

1 mark `Pointer` is decremented

1 mark value that was at the index previously indicated by pointer is returned

```
Function Pop() As Integer
    If Pointer > -1 Then
        Pointer -= 1
        Return locations(Pointer + 1)
    Else
        Return -1
    End If
End Function
```

1 mark past declared as an instance of Histo[illegible]in `PlayGame`

```
Sub PlayGame(ByVal Characters As ArrayList, ByVal Items As ArrayL
    Dim Past As New History
    Dim Stay Alive As Boolean = False
```

1 mark `Past` instance passed to `Go` when 'go' command entered by user (**A**. any

```
Case "go"
    Moved = Go(Characters(0), Instruction, Places(Characters(0)
Case "read"
```

1 mark declaration of `Go` now includes a parameter of type `History` in same po

```
Function Go(ByRef You As Character, ByVal Direction As String,
            ByVal CurrentPlace As Place, ByVal Past As History
```

1 mark attempt to call `Push` within at least one `Case` clause (even if unsuccessf

1 mark `Push` called correctly in all six cases, before p... is moved to the new lo current location before movement, th... ar... pushed after movement should contain all six calls ... of ...ion will be structured as below:

```
Case "nor"
    ...urrentPlace.North = 0 Then
        ...Moved = False
    Else
        Past.Push(You.CurrentLocation)
        You.CurrentLocation = CurrentPlace.North
    End If
```

1 mark new `Case` clause added for 'back'

1 mark call to `Pop` (**R.** if any pass could call `Pop` repeatedly)

1 mark selection statement to address return value of -1

1 mark message 'You cannot go back' displayed if -1 has been returned

1 mark location set to popped value only if it was not -1

1 mark rest of `Select Case` structure ... by new `Case`

```
Case "back"
    ... NewLocation As Integer = Past.Pop
    ...ewLocation = -1 Then
        Console.WriteLine("You cannot go back")
    Else
        You.CurrentLocation = NewLocation
    End If
Case Else
    Moved = False
```

1 mark having gone East, then back, the original room description should re-displ second time should result in 'You cannot go back' being displayed.

Task 12

1 mark addition of `Health` attribute to `Character` structure (**R.** if not integer)

```
Structure Character
    Dim Name, Description As String
    Dim ID, CurrentLocation, Health As Integer
End Structure
```

1 mark each `Character` added to the `Characters ArrayList` given `Heal`

```
TempCharacter.CurrentLocation = Reader.ReadInt32
TempCharacter.Health = 1
Characters.Add(TempCharacter)
```

1 mark correct declaration for `EatItem`

```
Sub EatItem(ByVal You As Character, ByVal Items As ArrayList,
```

1 mark index of item acquired by iteration or call to `GetIndexOfItem`

1 mark `ItemIndex` of -1 would not throw an exception

1 mark selection checks for 'edible' within the status of an item (**A.** if wrong item)

1 mark selection checks that an item is in the player's inventory (**A.** if wrong item)

1 mark five added to `Health` only if item is edible and in the inventory (**R.** if wro

1 mark output indicates that the item was eaten and new `Health` value displaye

1 mark 'You cannot eat that' displayed in all other circumstances, including if an i non-existent or misspelled item) was 'eaten'.

```
Dim ItemIndex As Integer = GetIndexOfItem(ItemToEat, -1, Items
If ItemIndex > -1 Then
    If Items(ItemIndex).Status.Contains("edible") And
            Items(ItemIndex).Location = Inventory Then
        You.Health += 5
        Items.RemoveAt(ItemIndex)
        Console.WriteLine("You have eaten the " & ItemToEat)
        Console.WriteLine("Your health is " & You.Health)
    Else
        Console.WriteLine("You cannot eat that")
    End If
Else
    Console.WriteLine("You cannot eat that")
End If
```

1 mark `Health` displayed after call to `DisplayGettableItemsInLocati`

```
DisplayGettableItemsInLocation(Items, Characters(0).CurrentLo
Console.WriteLine("Your health is " & Characters(0).Health)
Moved = False
```

1 mark valid `Case` clause added within `PlayGame`

```
    ReadItem(Items, Instruction, Characters(0).CurrentLocatio
Case "eat"
    EatItem(Characters(0), Items, Instruction)
```

1 mark `Health` initially output as 10; first response to 'eat apple' is that the appl is now 15; second call to 'eat apple' results in 'You cannot eat that'.

```
You are in a room with blue walls.  There is
On the floor is: torch, red die.
Your health is 10

> eat apple
You have eaten the apple
Your health is 15

> eat apple
You cannot eat that
```

Task 13

1 mark variable to track whether item has been taken (**A**. Exit command if item ta

1 mark loop to begin before `Count = 1;`

1 mark an item being successfully taken will terminate the loop

1 mark an item not being taken will cause the loop to repeat

1 mark output 'They don't ha... ... object including concatenation with `Chose`

```
Di  Ta  n As Boolean = False
Do
    Count = 1
    Console.Write("Which item do you want to take?  They have:
    Console.Write(ListOfNamesOfItemsInInventory(0))
    While Count < ListOfNamesOfItemsInInventory.Count - 1
        Console.Write(", " & ListOfNamesOfItemsInInventory(Cou
        Count += 1
    End While
    Console.WriteLine(".")
    Dim ChosenItem As String = Console.ReadLine
    If ListOfNamesOfItemsInInventory.Contains(ChosenItem) Ther
        Console.WriteLine("You have that now.")
        Dim Pos As Integer = ListOfNamesOfItemsInInventory.Ind
        ChangeLocationOfItem(Items, ListofIndicesOfItemsInInve
        ItemTaken = True
    Else
        Console.WriteLine("They don't ha  a " & ChosenItem &
    End If
Loop Until ItemTaken =
```

1 mark attem to answered with 'They don't have a box - try again'; a wit have that now'.

```
> playdice guard
Enter minimum: 6
Enter maximum: 6
You rolled a 6.
They rolled a 1.
You win!
Which item do you want to take?  They have: jar, gold
box
They don't have a box - try again
Which item do you want to take?  They have: jar, gold
jar
You have that now.

> _
```

Task 14

1 mark selection structure within `CheckIfDiceGamePossible` after `Playe` `PlayersInSameRoom` and `OtherCharacterHasDie` have been as statement

1 mark 'Player has no die' displayed only if `PlayerHasDi` is *False*

1 mark '_____ is not here' displayed only if `ay` `sInSameRoom` is *False*, to i

1 mark '_____ has no die' displayed only if `OtherCharacterHasDie` is *False* previous mark not awarded due to failure in concatenation)

```
End While

If Not (PlayerHasDie) Then
    Console.WriteLine("Player has no die")
ElseIf Not (PlayersInSameRoom) Then
    Console.WriteLine(OtherCharacterName & " is not here")
ElseIf Not (OtherCharacterHasDie) Then
    Console.WriteLine(OtherCharacterName & " has no die")
End If

Return PlayerHasDie And PlayersInSameRoom And OtherCharacterHa
```

1 mark 'Player has no die' on first attempt; 'guard is not here' on second attempt should still appear both times (**R.** if any additional output)

```
Enter filename> flag1


You [illegible]oom with blue walls.  There is a green d
On the floor there is: torch, red die.

> playdice guard
Player has no die
You can't play a dice game.

> get red die
You have got that now.

> playdice guard
guard is not here
You can't play a dice game.

>
```

Task 15

1 mark item names are padded with space to make them the same width

1 mark descriptions are displayed on the same line as names (**R**. if multiple inven

1 mark descriptions, but not names, are displayed in round brackets

```
For Each Thing In Items
    If Thing.Location = Th
        Console.Wri
        Cons
        C
        Console.Write(")" & vbCrLf)
    End If
Next
```

1 mark descriptions in brackets in a straight-line column

```
Enter filename> flag1


You are in a room with blue walls.  There is a green d
On the floor there is: torch, red die.

> examine inventory

You are currently carrying the follo
flask     (It is an empty plast
apple     (It is an appl
jar       (It is an
```

Name	

ZigZag Education supporting

A Level AQA Computer Science Pap

Summer 2019

AQA Text Adventures

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 any additional pages

- Answer **all** questions
- The marks available 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:

Written Questions

Answer all questions.
Remember to save this document regularly.

Q		
1	(a)	
	(b)	
	(c)	
	(d)	
	(e)	
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		

Programming Tasks

Answer all questions.
Remember to save this document regularly.

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