|  |  |
| --- | --- |
| **20 minutes** | **Task 1: HashMaps** |
| Take a look at the W3Schools article on HashMaps in Java: <https://www.w3schools.com/java/java_hashmap.asp>  HashMaps are a way of collecting key-value pairs of information, allowing you to access information based on the data’s name.  This is a self-taught task, and minimal guidance will be given. This encourages you to play around with the code, explore solutions yourself, and begin to think like a programmer when faced with new topics and features.   1. Create a simple program to practice using HashMaps that will store key-value pairs. 2. The key value pairs can be of anything you want, but if you struggle to think of an idea, you can choose from the options:    1. Catchphrases or quotes of famous people or books    2. Sports results for world finals for each year    3. Network error codes and descriptions (404 page not found, 501 not implemented, etc)    4. Capital Cities (like the W3Schools examples) – this should be a last resort! Try something different yourself, first! | |

|  |  |
| --- | --- |
| **4 hours** | **Task 2: OOP – Text Based Game** |
| This task will begin to utilise your knowledge on OOP. Your task will be to create a text-based game, the start of which we covered in our class this week when creating the Character class.   1. Create a new project and give it any name, but something relevant to the type of game you are going to make. PunchingPerson, HideAndSeek etc. 2. Create a Character class, like we did in this week’s lesson. This character class will be used to create the characters in your game. The characters could be people, animals, robots, whatever you like!    1. This class should have a minimum of 3 attributes (name, health, alignment, etc)    2. This class should be created with encapsulation in mind. This means Getters and Setters       1. Ensure that your variables are private, and your methods are public 3. Create a Map class. This class will be used to generate the area that the player can move in.    1. This class should be used to create and X and a Y value that will define the size of your map    2. This class should also contain a currentX and currentY value that will show you where the player currently is on that map.    3. You will only want to make Getters for the X and Y value    4. You will need to make a method called “move(String direction)” which will accept “N”, “E”, “S”, or “W” values.       1. Depending on the direction passed to the method, move the character’s current position by incrementing or decrementing X and Y.       2. If the direction moves the currentX or currentY higher than the X or Y values, the value should loop round to zero       3. If the direction moves the currentX or currentY lower than 0, the value should loop round to X or Y       5. **PLEASE NOTE THAT TO COMPARE A STRING .equals() MUST BE USED** 4. Inside your main method    1. Create a new main character, the player    2. Create a map with any X and Y values you want. 10, 10 would be a sensible size.    3. using Math.random(), place the player somewhere in your map by setting the X and Y values of the map.    4. Create a loop that will ask the user for a direction and then use the map’s move method to move the player.    5. When the player reaches a destination of your choice (maybe 5, 5) end the loop – they have found the exit!   STRETCH CHALLENGE: You can be as creative as you want in this. Try adding events that happen when the player reaches a specific XY location. This could be a combat phase, or it could be a poem printed to the screen, or it could randomly teleport the player to a new location (or a new map!?).  STRETCH CHALLENGE: Introduce HashMaps to your game. You could use this for things like equipment slots for your player (“head”:”crown”, ”torso”:”jacket” etc). | |

## **Marking Criteria Task 1/3/4**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Syntax** | * Attempts to use Java syntax with some success | * Java syntax is largely accurate with some errors | * Java syntax is consistently accurate and appropriate to the task |
| **Presentation** | * Some whitespace used to good effect * Indentation attempted but inaccurate * Correct naming convention | * Whitespace used appropriately, at times * Indentation largely accurate | * Clear commentary provided throughout * Code is explicitly clean and easy to read |

## **Marking criteria Task 1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempted using hashmaps | * Created a hashmap and added in some values | * Create a hashmap, added in values, and accessed those values |

## **Marking criteria Task 2**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempted to create new class beyond the first main class * Attempted to create an object of the new class * Attempted to access/manipulate data within that class | * Created a new class to store the character * Attempted to create a new class that stored the map * Attempted to use Scanner to take a user input * Attempted to update the map location based on the user input | * Successfully creates a character and map class * Successfully allows the user to navigate the map without “falling off the edge” or “going out of bounds” * Successfully ends the loop when the player gets to a specific X/Y location |