Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code	CM6121/CM6621
Module Title	Fundamentals of Computing with Java
Lecturer	Dr Soumya C. Barathi
Assessment Title	Individual Project Portfolio
Assessment Number	1 of 2
Date Set	Wednesday 23rd February 2022 @ 9:30AM
Submission Date and Time	Wednesday 23rd March 2022 @ 9:30AM
Return Date	By end of Friday 22nd April 2022

This assignment is worth **50%** of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1 If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;

2 If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here: https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf

CONTENTS

SUBMISSION INSTRUCTIONS

The following elements must be submitted through Learning Central. **Do not put your name on the cover sheet or the portfolio.**

Description		**Type**	**Name**
Cover sheet	**Compulsory**	One PDF (.pdf) file	[student number].pdf
Gitlab URL	**Compulsory**	URL to Gitlab project site	Wormhole_[studentNumber]_2022
Portfolio	**Compulsory**	One PDF (.pdf) file	Portfolio_[student number]_2022.pdf
The module leader (Dr Soumya C. Barathi) and the moderator (Prof. Yukun Lai) **must** be given **Maintainer** access to your repository. The penalty for not doing this will be a mark of **zero (0)**.			

Any deviation(s) from the submission instructions above (including the number and types of files submitted) will result in a reduction in marks for that assessment or question part of **20%**

STAFF RESERVE THE RIGHT TO INVITE STUDENTS TO A MEETING TO DISCUSS COURSEWORK SUBMISSIONS

ASSIGNMENT

BACKGROUND

Since both CM6121/CM6121 and CM6123/CM6623 have Java elements, the 1st assessments of each module use a combined problem statement. The requirements of each module are different, and your work will be assessed on different criteria, but you will only have one problem to solve.

You are required to write a computerised version of a new board game called "Wormhole".

RULES OF THE GAME

- The game is played on a square board of 25 to 100 squares (i.e. 5x5 to 10x10).
- Each board can have a number of wormholes on it.
- The maximum number of wormholes is the width of the board.
- For each wormhole, we put an entrance and an exit on the board.
- The entrances and exits can be anywhere on the board except for the first and last squares.
- A square can't be both an entrance and an exit of a wormhole.
- Wormholes are either "positive" or "negative".
- When a player lands on an entrance, they will be moved to a selected exit (exception below). If the wormhole is "positive", then they move to an exit that is ahead. If the wormhole is "negative", then they move to an exit that isbehind them. If no such exits exist, then they stay where they are.
- If a player rolls a double and lands on a "negative" wormhole, then they stay where they are (i.e. they are "saved" by the double).
- The game can be played by 2 to 6 players and is played with 2 x 6-sided dice.
- Players can be manual (where players enter the outcome of a roll of the physical dice) or automatic (where the computer generates the outcome of the roll of virtual dice).
- Player names must be alphabetic.
- The winner is the first player to reach the last square. They may go beyond the end of the board to win (e.g. if at square 98 on a 100-square board, they can roll a 2+4 and win).

21	22	23	24	25
20	19	18	17	16
11	12	13	14	15
10	9	8	7	6
1	2	3	4	5

5 x5 board with 5 (max) entrances in yellow (positive) or orange (negative) and 5 (max) exits in green.

OPERATION

- At start-up, the console will ask for the size of the board and for the names of the players.
- The game will generate or configure the board with wormholes according to the rules.
- Each player is identified by a name.
- Each player is designated manual or automatic.
- It will then give each player a turn.
- It will ask manual players to enter dice rolls as 2 numbers.
- It will randomly pick dice rolls for automatic players.
- After the roll, it will move the current player and determine whether that player has won.
- If a player lands on a wormhole entrance, they will be moved to an exit square**. Whether this is random or constant should be part of the game design. That is the game can pick an exit at random each time, or it can be pre-determined when the game is created.**
- The current position of each players will be output and, if there is a winner, the winning player will be identified, and the game will end. The players should be asked if they want to play again.

SAMPLE CONSOLE LOG

```
Welcome to Wormhole.

Please enter the width dimension of your board?

>5

Thank you.  The board has 25 squares.  There are wormhole entrances at 13,17
(both positive) and at 9, 20 & 22 (negative) and wormhole exits at 2,6,11,14 &
23.

Please enter the number of players?

>2

Please enter the name of player 1?

>Carl

Please enter the name of player 2?

>Hélène

Carl – do you want to roll the dice, or shall I do it for you? Type “Y” to roll
yourself or “N” to let me do it.

>Y

Hélène - do you want to roll the dice, or shall I do it for you? Type “Y” to roll
yourself or “N” to let me do it.

>N

Let’s play.

Carl is on square 1.

Hélène is on square 1.

Carl - please roll the dice.

>6,6

Carl moves to square 13 which is a POSITIVE wormhole…

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> exit at 23

Carl is on square 23.

Hélène is on square 1

Hélène rolls 4,2.

Hélène moves to square 7.

Carl is on square 23

Hélène is on square 7

Carl - please roll the dice.

>2,3

Carl moves to square 25

Carl is in square 25 and has won!

Hélène is on square 7

Would you like to play again? (Type “Y” to play again, or “N” to exit)

>y
```

This is indicative. You may implement a different user interface.

THE PORTFOLIO

You are required to produce a portfolio containing two sets of artifacts: code and writing.

Codebase	**Document**
The code artifacts will provide evidence that you can take a given problem and implement a high- quality solution to it using the OOP paradigm. The code will demonstrate your understanding of some of the concepts presented in this module and you will be assessed on these: • Design of the classes and methods • Data structure • Encapsulation • Inheritance/Abstract classes/Interface • Exception • Validation The solution will be tested using manual tests in addition to the automated checks.	The writing artifacts will provide evidence you can understand the basic principles of OOP and Java Syntax using the correct vocabulary. You can justify your choice using good quality references and can compare them to other possibilities.
60% of the overall marks	**40% of the overall marks**

CODE

Your code will be judged both automatically and manually. You must fork the starter project in Gitlab and you must not change the provided configuration files without permission from the module leader. For the Jar to run, it will need to be the shadow jar. This is built into the starter project. Your work will only add Java packages and classes whose source files go into `src/main/java` and `src/test/java` folders. https://git.cardiff.ac.uk/ase-sds21/ase-sds2/assessment-1-2022-starter

Your code will be built every time you push code to a tag on Gitlab.

Assessment and marking of the code will be done by checking the output of your program against intensive tests run by the marker. Your program should be launched using:

java –jar wormhole_studentNumber.jar [arguments]

arguments are optional

If your jar fails to run (or you don't provide one) the award will be 0 point for the code section even if you have implemented all the features. Make sure that your software is running following the instructions you provided in the README.md. Your program must run on any computer (including mine). The java version that you should use is the one included by default in the NSA laptop (JDK 11). **IntelliJ will NOT be used to run your code.** The module leader will run some automatic and manual look up of the code, to assess the different concepts mentioned above.

CORE SCENARIOS

You are required to implement tests for the following core scenarios.

Given	**When**	**Then**
A game is being played on a boardof size 4 with no wormholes and all players are on square 1 and it is player 1's turn	A 3 and 4 are rolled	Player 1 ends on square 7 and it is player 2's turn
A game is being played on a boardof size 3 with no wormholes and all players are on square 1 and it is player 1's turn	A 5 and 6 are rolled	Player 1 is on square 9 and is declared the winner.
A game is being played on a boardof size 5 with no wormholes and all players are on square 1 and it is player 1's turn	When all players have rolled	It is Player 1's turn
A game is being played on a boardof size 5 with a positive wormholefrom square 6 to square 20 and all players are on square 1 and it is player 1's turn	Player 1 rolls a 4 and 2	Then Player 1 should be on square 20 and it is player 2's turn
A game is being played on a boardof size 5 with a positive wormholefrom square 6 to square 20 and all players are on square 1 and it is player 1's turn	Player 1 rolls a 3 and 3	Then Player 1 should be on square 20 and it is player 2's turn
A game is being played on a boardof size 5 with a negative wormhole from square 10 to square 2 and all players are on square 1 and it is player 1's turn	Player 1 rolls a 6 and 4	Then Player 1 should be on square 2 and it is player 2's turn
A game is being played on a boardof size 5 with a negative wormhole from square 10 to square 2 and all players are on square 1 and it is player 1's turn	Player 1 rolls a 5 and 5	Then Player 1 should be on square 10 and it is player 2's turn
A game is being played on a board of size 5 with a positive wormhole from square 6 to square 20 and 2 players are on square 1 and it is player 1's turn	When Player 1 rolls a 4 and 2 And Player 2 rolls a 1 and 1 And Player 1 rolls a 2 and 3	Then Player 1 should be on square 25 and should be declaredthe winner

A game is being played on a boardof size 5 with a positive wormholefrom square 6 to square 20 and 2 players are on square 1 and it is player 1's turn	When Player 1 rolls a 4 and 2 And Player 2 rolls a 1 and 1 And Player 1 rolls a 3 and 3	Then Player 1 should be on square 25 and should be declaredthe winner

You should also use tests to ensure that:

- The size of the board does not exceed the stated minimum and maximum
- The number of players does not exceed the stated minimum and maximum
- Wormhole entrances and exits are not placed outside the bounds of the board
- Wormholes entrances and exit are unique (i.e. no square has dual purposes).
- Positive wormholes do not send the player backwards
- Negative wormholes do not send the player forwards

CRITERIA FOR ASSESSMENT OF YOUR WRITING (40%)

Your assessment requires you to prepare a portfolio that demonstrates your understanding of Object-Oriented Programming and Java syntax. The aim of this portfolio is to present the Java side of your application.

Your portfolio should consist of the following sections. You should aim to write no more than 1200 words overall (this is not including the headings, footnotes, references, titles, annotation of images, tables). You should refer to evidence from your project to demonstrate your discussion, and you should cite external works correctly. Any screenshots should be readable on A4 paper and captioned.

Artefact	**Contents**
Section 1 (50%): Design of classes and methods according to OO paradigm (max 600 words) Before starting to write your code, you should design your classes. In Git, you will find an initial structure for the code. You will need to take this code and start to design your classes based on this code. You will use the UML's class diagram to design these classes with a tool such as Visual Paradigm. These class diagrams will certainly change with your implementation and your understanding of OOP and Java. To demonstrate these changes, you will add all the versions of your classes diagrams in your Git repository in a folder named UML_diagram. Each version will be saved in different files. The name of your file(s) needs to follow this pattern UML_Wormhole_[studentNumber]_[pre_coding|in_coding|post_coding]_[version number] (e.g. UML_Wormhole_c000000_pre-coding_v1, or UML_Wormhole_c000000_in_coding_v2). When you commit these files, the comment should mention the UML class diagram and which version it is (e.g. commit -m "UML class diagram pre-coding v1"). You will have (as a minimum), the pre-coding version and the post-coding version of your class-diagram. this will serve you as evidence for the first discussion (see below). You will discuss the evolution of your class diagram through the weeks and the rationale of the choice of your classes and methods design. You can reference the different classes diagram using the name of your file in Git and a link to the class diagram in your Git repository. You should have at least two versions, the first one and the last one.	

Your discussion should demonstrate and explain (using OOP terms) how your implementation/design fits to the OOP paradigm. You will use references to explain the rationale of your choice. To go above 49%, this section needs to have external references.
Section 2 (25%): Visibility of classes, methods and fields (max 300 words)
In this section, you will discuss the visibility of your classes, methods, and fields. You should discuss the reasoning behind the visibility of your classes, methods, and fields and justify these choices using good quality references. To go above 49%, this section needs to have external references. Provide at least one example of appropriate visibility for Classes, methods, and fields (each of them, setter and getter not accepted).
Section 3 (25%): Exception handling and validation (max 300 words)
In this section, you will discuss how you are handling exceptions and validate the data. You will provide at least one example of a method that demonstrates how you handle exceptions, and one example on how you are validating your data. You should discuss the reasoning behind how you are handling exceptions and validating the data and justify these choices using good quality references. To go above 49%, this section needs to have external references.
Please ensure that there is no repetition of discussion. Be aware that if you choose to discuss about the same OOP concept (encapsulation, inheritance,abstraction, or polymorphism) several times only one of your discussions will be assessed.

References to code must be via a hyperlink to the Git commits or files. Please use Microsoft Word's hyperlink feature rather than simply copying and pasting URLs into the document.

All influences on your thinking should be provided as citations and cited using a consistent style (e.g University Cardiff Harvard style). Ideally, you should use a broad range of references. The bibliography must be provided at the end of the document, and no inline references will be counted as no references at all.

LEARNING OUTCOMES ASSESSED

1. Design, write, read, and debug Java code effectively, through knowledge of syntax and understanding of core concepts.
2. Effectively design classes and interfaces, according to the principles of object-orientation (OO), for real world problems.
3. Design code effectively; for example, selecting appropriate visibility for class members, re-using code where appropriate.
4. Effectively use the Java compiler, and Java Virtual Machine (JVM), by understanding their functionality, purpose, and errors.

CRITERIA FOR ASSESSMENT

CRITERIA FOR ASSESSMENT OF YOUR CODE (60%)

- Design of the classes and methods
- Data structure
- Encapsulation
- Inheritance/Abstract classes/Interface

Assessment Criteria **(**Equal division of marks for each assessment criteria.**)**	***Poor (<40%)***	***Tolerable (40% - 49%)***	***Good (50% – 69%)***	***Excellent (>69%)***
Requirement Fulfilment (40%)	*Limited/no task completion or requirements not met*	*Some completion of more than a third of the tasks.*	*More than half of the tasks completed and fulfilled to meet requirements*	*Initiative shown-creativity. All of tasks completed.*
Use of object oriented design patterns and java syntax. (40%)	*Project shows very little adherence to structure: very poor use of variables types, methods and classes.*	*Project shows some adherence to structure: correct use of variables types, methods and classes.*	*Project shows good adherence to structure: correct use of variables types, methods and classes, tentative exploration of advances concepts of OOP such as Inheritance, Interfaces,*	*Project shows very good adherence to structure: well, thought through use of variables types, methods, classes, and inheritance, use of advanced OOP concepts such as Inheritance, Interfaces,*

			Abstract classes	*Abstract classes*
Selecting appropriate modifiers for package, class, and class members (10%)	*Project does not follow the principle of Encapsulation.* *The selection of inappropriate visibility of the classes or class members.* *The type of modifier is not appropriate.*	*Some classes members or classes follow encapsulation, and appropriate selection of visibility showing some understanding of this OOP.* *The type of modifier is appropriate for some of the methods and variables*	*Most classes members or classes follow encapsulation, and appropriate selection of visibility showing some understanding of this OOP.* *The type of modifier is appropriate for most of the methods and variables*	*All classes members or classes follow encapsulation, and appropriate selection of visibility showing complete understanding of this OOP.* *The type of modifier is appropriate for all the methodsand variables*
Implementation of methods: signature (10%)	*None or very few methods have an appropriate signature that follows good logic.*	*Some methods have a signature that follows good logic.*	*Most methods have a signature that follows good logic*	*All methods have a signature that follows good logic*

CRITERIA FOR ASSESSMENT OF YOUR WRITING

Credit will be awarded against the following criteria for each section.

Please ensure that there is no repetition of discussion. Be aware that if you choose to discuss about the same OOP concept (encapsulation, inheritance,abstraction, or polymorphism) several times only one of your discussions will be assessed.

Please follow the different sections to write your discussion.

Assessment Criteria	***Poor (<40%)***	***Tolerable (40% - 49%)***	***Good (50% – 69%)***	***Excellent (>69%)***
Section 1: Design of classes and methods according to OO paradigm (max 600 words) (35%)	Limited or no discussion. The vocabulary used is very limited and not OOP Your points are not justified and are incoherent	Some discussion is provided. Limited usage of the vocabulary OOP Your points are not justified with references or the references are inconsistent.	A good discussion is provided. The vocabulary is OOP and is correctly used. Your points are justified with references.	A very good and original discussion is provided. The vocabulary is OOP and is correctly used. Your points are justified with very good quality references.
Examples provided and UML class diagram (15%)	No or very few examples provided and they are not appropriate or don't support the discussion. Your UML class diagram doesn't correspond to your implementation	Examples provided are supportive and appropriate to the discussion The code provided shows some evidence of logic.	Good examples provided which are supportive and appropriate to the discussion, The code provided in the example is logical and clear.	Excellent and original examples provided which are supportive and appropriate to the discussion. The code provided in the example is logical, clear, and exempt of error

	or is not appropriate.	Your UML class diagram shows little matching with your code.	Your UML class diagram shows all your classes, provides a clear and logical description of your classes, and shows a good understanding of the OOP design principle.	Your UML class diagram shows all your classes, provides a very clear and logical description of your classes, and shows a good separation of concerns
Section 2: Visibility of classes, methods and fields (20%)	*Limited or no discussion*. The vocabulary used is very limited and not OOP Your points are not justified and are incoherent	*Some discussion is provided.* *Limited usage of the vocabulary OOP* *Your points are not justified with references or the references are inconsistent.*	*A good discussion is provided.* *The vocabulary is OOP and is correctly used.* *Your points are justified with references.*	*A very good andoriginal discussion is provided.* *The vocabulary isOOP and is correctly used.* *Your points are justified with very good quality references.*
Examples provided for section 2, minimum one for each (5%)	*No example(s) provided or is not appropriate or doesn't support the discussion.*	*Example(s) provided is supportive and appropriate to the discussion.* *The code provided shows some evidence of logic.*	*Good example(s) provided which is supportive and appropriate to the discussion.* *The code provided in the example is logical and clear.*	*Excellent and original example(s) provided which is supportive and appropriate to the discussion.* *The code provided in the example is logical, clear, and exempt of error.*

			.	
Section 3: Exception handling and validation (20%)	*Limited or no discussion*. The vocabulary used is very limited and not OOP Your points are not justified and are incoherent	*Some discussion is provided.* *Limited usage of the vocabulary OOP* *Your points are not justified with references or the references are inconsistent.*	*A good discussion is provided* *The vocabulary is OOP and is correctly used.* *Your points are justified with references.*	*A very good and original discussion is provided.* *The vocabulary is OOP and is correctly used.* *Your points are justified with very good quality references.*
Examples provided for section 3 (one for each) (5%)	*No or very few examples provided and they arenot appropriate or don't support the discussion.*	*Examples provided are supportive and appropriate to the discussion.* *The code provided show some evidence of logic.*	*Good examples provided which are supportive and appropriate to the discussion.* *The code provided in the example is logical and clear.*	*Excellent and original examples provided which are supporting the discussion.* *The code provided in the example is logical,clear, and exempt of error*

FEEDBACK AND SUGGESTION FOR FUTURE LEARNING

Feedback on your coursework will address the above criteria and will be provided via Learning Central by the stated return date.

The Feedback given will be useful for the next coursework and other coursework in the course.