The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, SPRING SEMESTER 2022-2023

PROGRAMMING PARADIGMS (COMP1009)

Time to Answer the Exam Paper: TWO HOURS

This is a take-home and open-book exam with answers to be submitted in Moodle no later than the date/time indicated in the Moodle dropbox.

Answer ALL QUESTIONS

Submit your answers in a single PDF file, with each page in the correct orientation, to the dropbox in the module's Moodle page. You are recommended to write/draw your answers on paper and then scan them to a PDF file. Alternatively, you may also type/draw your answers into electronic form directly and generate a PDF file.

Your solutions should include complete explanations and should be based on the material covered in the module. Make sure your PDF file is easily readable and does not require magnification. Text/drawing which is not in focus or is not legible for any other reason will be ignored.

Use the following naming convention for your PDF file: `StudentID_COMP1009`.
Include your student ID number at the top of each page in your PDF file.
Please do not include your name.

Staff are not permitted to answer assessment or teaching queries during the period in which your examination is live. If you spot what you think may be an error on the exam paper, note this in your submission but answer the question as written.

You must produce the answers by yourself only. You must adhere to the University's Policy on Academic Integrity and Misconduct. You are also not allowed to share this exam paper with anyone else or post it anywhere online.

1. Java / Object Oriented Total 50 marks

You need to create a system to allow players to play a card game on the computer. Assuming that two classes are created for you (see the `AIDecision` and `UserInterface` classes on page 3), create an OO design and consider how you would convert it into Java code for the requirements on page 3, providing answers to the following questions:

1 a) Describe your design [25 marks]

- Provide a table like that below, to explain the classes that you would use in your design, their purposes, their key attributes and what data each attribute stores, and their key methods and what each method does (for simplicity, ignore getters/setters).

Class	Purpose	Attributes	Methods
MyClass	The only class in this example, which is here only for illustrative purposes – do not include this in your answer	'name' - stores the name of the MyClass object 'doc' - the 'date of creation' of the object	info() – returns a string with the name of the object and its date of creation clone() – create a duplicate of this object

- Explain any inheritance or aggregation/composition relationships between classes and why each exists.
- Briefly explain how your design would work, to show why your design meets the requirements: how do your classes work together to implement the requirements and what happens when your program is executed.

1 b) Explain parts of your design [25 marks]

Explain the following parts of your design by providing a clear explanation <u>and</u> sample Java code:

- Explain what happens when your program starts if player 1 is human, player 2 is AI, player 3 is human and player 4 is AI. Explain what objects are created in what order by your design and give Java code examples for how your program would create the initial objects and link them together. You should clearly show which objects are created and where they are created.
- Explain what parameters (name, purpose and Java type) and what return types you would use for Java implementations of the methods `show()`, `askHuman()` and `askAI()` in your design.
- Explain which of the design patterns <u>that we have considered in this course</u> you used in your design, why you used them (what are they for), and how you would implement these parts of the design in Java.

Program/System Requirements:

You need to provide a design for a program to allow two or more players to play a card game on the computer. Each player may be either a human or an AI. Assume that each player has their own deck of 52 cards, numbered 1 to 52. At the beginning of the game, each deck is shuffled. Each player has a 'hand' of cards, a collection of cards that they are 'holding' and are ready to play. Each player will then keep drawing from their deck (moving the top card from their 'deck' into their 'hand') until they have 5 cards in their 'hand'.

Game rules: Starting at player 1, each player will play one card from their 'hand' onto the virtual 'table', in order, first player 1, then player 2, player 3, etc. Whoever plays the highest numbered card wins the 'round', gets a point, and the 'round' ends - the played cards are removed from play. Each player then draws another card from their deck to their hand, so that they again have 5 cards, as long as there are cards left in their deck, otherwise they just play with the cards in hand. The player who won the previous round goes first in the next round, then each other player plays in turn, keeping the same order of player numbers, and returning to player 1 after the highest numbered player. E.g., with 5 players, if player 3 started, the player order would be 3, 4, 5, 1, 2. Play continues until each player has played all 52 of their cards. At the end of the game, the player with the most points wins.

UI: You do not need to consider how to display the game to the human players and allow them to input commands: assume that you have a user interface class called `UserInterface` to do this, which has a function that you can call (called `show()`) to display the current state of the card game to each player. You need to give `show()` information about the cards that are in each player's hand, the cards that each player has played so far, the points each player has won so far, and whether it is the end of the game or not (so it displays final scores). Assume that `UserInterface` also has a function called `askHuman()` that you can call to ask any specific player what card they want to play, which allows a player to choose a card to play from their hand (which they can see from the last `show()`). The exact details of what parameters to pass to `show()` and `askHuman()` will depend upon your design and you should consider what parameters to use.

AI: You do not need to consider how the AI would be implemented for AI players: assume that you have a class called `AIDecision`, which has a single function (called `askAI()`) which will tell you what card to play if you tell it the cards that are in its hand, the cards that have been played by each player so far this round, and all of the cards that have been played so far by each player. You should decide what parameters would be passed to this function based on your design and should assume that this class would be implemented for you to take the parameters that you have chosen.

You should make it possible for the program to easily check which cards are in a hand or deck, returning each card one at a time, since the User Interface and AI may need this.

Your program should work with different numbers of players and allow any player to be either human or AI. Assume that you specify the number and type of players at the start of the game (in code). (See Q1b, explaining what happens when the program starts.)

You should make it easy for a spectator feature to be added, so that it is possible for a spectator (class) to ask to be told about each card that is played.

End of Question 1: Total 50 marks

2. Haskell / Functional Programming **Total 50 marks**

Suppose that you have been asked to write an article for a computing website on

"Recursive Functions in Haskell"

Write a short article (maximum 750 words) on this topic, explaining how recursive functions can be defined in Haskell, illustrating a range of different mechanisms. Each of the mechanisms that you mention should be illustrated with a small example in Haskell.

[50 marks]

Further instructions:

- Please provide a word count for your article.
- Your article should be based on concepts that were taught in the module, and must have a clear narrative structure rather than simply being a list of mechanisms.
- The Haskell examples should be devised by yourself rather than being copied or adapted from the course materials, and should be included in the word count.
- You may assume that your audience is familiar with the basics of programming but has no Haskell experience. Your article should be self-contained, and not include a bibliography or links to external web pages or other resources.
- This take-home examination is conducted under normal examination conditions. As such, we are not able to offer help or feedback on draft articles.
- Articles must not be posted online.

End of Question 2: Total 50 marks