**Section C**

You are advised to spend no more than **10 minutes** on this section.

Type your answers to **Section C** in your Electronic Answer Document.

You **must save** this document at regular intervals.

These questions refer to the **Preliminary Material** and the **Skeleton Program**, but **do not** require any additional programming.

Refer **either** to the **Preliminary Material** issued with this question paper **or** your electronic copy.

**02**

**Figure 1** shows a class diagram that describes the relationships between the classes used in the Skeleton Program.

**Figure 1**

Icon

Description automatically generated with medium confidence

1. State the identifier of a protected attribute in the skeleton code.
2. State the identifier of a private attribute in the skeleton code.

Aggregation, composition and inheritance are three different types of relationship that can exist between classes.

1. Briefly describe each type of relationships.

**03**

Summarise the functionality of CheckSquareIsValid method in domain-specific terms (specific to the game, not the software implementation – i.e. that do not refer to the variables/code.)

For example, in addition to specifying what the parameters refer to, the summary will start off something like: Is the square reference within the board dimensions.

**Turn over for Section D**

**Section D**

You are advised to spend no more than **30 minutes** on this section.

Type your answers to **Section D** in your Electronic Answer Document.

You **must save** this document at regular intervals.

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

make programming changes to it.

**04**

The following question requires you to modify existing functionality in the Dastan class.

A logical error was found in the PlayGame method of the Dastan class.

After a player selects a move, and a valid player piece to move, the game asks the player to select a position to move the piece to.

Should the player select an invalid position that is not in the list of Player moves, the game skips the turn and proceeds to Player 2’s turn.

**What you need to do:**

What should happen, is that Player 1 should be informed of the illegal move – and should be able to try again.

**e.g.**

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 22

1 2 3 4 5 6

╭──┬──┬──┬──┬──┬──╮

1 │ │ │K1│ │ │ │

2 │ │ !│ !│ !│ !│ │

3 │ │ │ │ │ │ │

4 │ │ │ │ │ │ │

5 │ │ "│ "│ "│ "│ │

6 │ │ │ │k2│ │ │

╰──┴──┴──┴──┴──┴──╯

Enter the square to move to (row number followed by column number): 31

**Illegal move, try again.**

**Evidence that you need to provide**

Include the following evidence in your Electronic Answer Document.

1. Your PROGRAM SOURCE CODE for the amended method PlayGame.
2. SCREEN CAPTURE(S) showing the results of testing the output with a legal-move and an illegal move.

**05**

The following question requires you to make a new class Promotion and to alter the functionality of an existing method createboard.

A new rule is to be introduced like that in Chess or Checkers. If a player advances their pawn to a promotion square, they exchange any piece for a mirza. The promotion squares are those in the corners of the board and are marked with a P or a p according to which player’s pieces can be upgraded.

1 2 3 4 5 6

╭──┬──┬──┬──┬──┬──╮

1 │p │ │K1│ │ │p │

2 │ │ !│ !│ !│ !│ │

3 │ │ │ │ │ │ │

4 │ │ │ │ │ │ │

5 │ │ "│ "│ "│ "│ │

6 │P │ │ │k2│ │P │

╰──┴──┴──┴──┴──┴──╯

If the corresponding player’s piece lands on a promotion square, it is automatically upgraded to a mirza. You should assume Player 1’s pieces to be those with a "!" symbol (which you can check against.)

**What you need to do:**

**Task 1:**

Create a new Class called Promotion that sets the Piece to a mirza, according to which player owns the square, which piece lands on it, and what the symbol of the piece is.

**Task 2:**

Amend the createboard method to add the relevant promotion squares at the corners of the board (as above) (Player 2’s Promotion squares on the corners where player 1 resides, and vice-versa.)

**Task 3:**

Test that the functionality works by allowing a player’s piece to land there and seeing it change to a mirza (only show the current move, and the previous move.)

**Evidence that you need to provide**

Include the following evidence in your Electronic Answer Document.

1. Your PROGRAM SOURCE CODE for the class Promotion.
2. Your PROGRAM SOURCE CODE for the amended method createboard.
3. SCREEN CAPTURE(S) showing the results of the requested test.

**06**

**Display Move Options:**

The following question requires you to make a new method printMoves in the dastan class, and to amend the PlayGame method.

After selecting a move option, and then selecting a piece – it isn’t clear where the piece can move. A list of possible moves, highlighted on the board would be beneficial.

**The following shows the expected result when Player 1 selects Ryott.**

Choose move option to use from queue (1 to 3) or 9 to take the offer: 1

Enter the square containing the piece to move (row number followed by column number): 22

Possible Moves: 12 21 32

1 2 3 4 5 6

╭──┬──┬──┬──┬──┬──╮

1 │ │ m│K1│ │ │ │

2 │ m│ !│ !│ !│ !│ │

3 │ │ m│ │ │ │ │

4 │ │ │ │ │ │ │

5 │ │ "│ "│ "│ "│ │

6 │ │ │ │k2│ │ │

╰──┴──┴──┴──┴──┴──╯

Enter the square to move to (row number followed by column number): …

**What you need to do:**

To do this, you will create a method printMoves that takes the moveOption chosen by the player, and a valid StartSquareReference. This must be called in the relevant location in PlayGame.

This method will iterate over the entire board, selecting a valid move, in a valid square with the current moveOption. These moves are displayed in a string as above “Possible Moves: 12 21 32”

A temporary piece called a movePiece must be generated. This will be of type “move” and be given a symbol ‘m’, the only function the movePiece serves is to replace the pieces located in the relevant areas of the board to show a valid move.

Finally – the board is displayed to the player.

In order that these movePieces do NOT affect the functionality of the game [these pieces may or may not replace other pieces when placed on the board] – the state of the board must be recorded before adding the movePieces, and then recovered after displaying the board.

Notice in the example above – a valid move is a move that can capture an enemy piece, or that falls on a blank square.

**Test the functionality:**

Start the game. Select option 3, and then 24. Show the result of the display.

**Evidence that you need to provide**

Include the following evidence in your electronic Answer Document.

1. Your PROGRAM SOURCE CODE for the few lines of code showing the region where you called printMoves.
2. Your PROGRAM SOURCE CODE for the added method printMoves.
3. SCREEN CAPTURE(S) showing the results of the requested test.