Curtin University — Discipline of Computing

Object Oriented Software Engineering (COMP2003)

Semester 1, 2020

Take-Home Examination

Weight: 35% of the unit mark.

Questions. This question paper contains FOUR (4) questions. Answer all of them. There are 70 marks available in total. The marks allocated for each question are shown beside the question.

Answer Format. Enter all your answers into a single Word (.docx), LibreOffice (.odt) or text (.txt) file (your "answer document"). When you write an answer, clearly indicate the relevant question number/letter. Include your name and student ID at the start.

Timeframe. You have 7 days (168 hours) to complete and submit your answers, from 8:30am on Tuesday 16 June 2020 until 8:30am on Tuesday 23 June (UTC+8). You may schedule your work however you wish within this period. However, late submissions will not be accepted.

Declaration of Originality. You must also complete and submit a declaration of originality (whether scanned, photographed, or filled-in electronically.) A blank form is available from the same place as this question paper.

Submission. Submit (1) your answer document, and (2) your declaration of originality. Submit both to the "Take-Home Examination" area on Blackboard. You may submit as many times as you wish prior to the deadline. Only your final submission will be marked.

You *must* verify that your submission worked (by re-downloading your submission). Correctly submitting is entirely your responsibility.

Reference Material. This is an OPEN BOOK assessment. You may refer to any written material, including your notes, course materials, books, websites, etc. However:

- You must complete this assessment *entirely on your own*.
- During the assessment, you *may not* communicate with any other student regarding the test.
- During the assessment, you *may not* communicate with any other person in order to seek or receive an answer to any part of the assessment.
- Your answer document will be checked by text matching software for signs of cheating, collusion and/or plagiarism.
- The questions have been designed such that any two students, working independently, should not produce the same answers.

Question 1 (20 marks)

This question concerns the Observer Pattern.

Consider a game (either 2D *or* 3D) in which the player controls a character moving through a world, collecting items and possibly fighting creatures. (This could be as simple as Mario Bros, or as complex as a modern first-person shooter.)

One of the events that the game must handle is a physical collision between two things in the game world. That is, one thing bumps into or falls onto another thing. The result of a collision depends on which things collide; e.g., if the character collides with a "bonus" of some sort, the character's health or inventory will be modified accordingly, and the bonus will disappear. Part of the system must be responsible for detecting such collisions, and another part (or parts) must be responsible for handling what happens.

There are various other events too, including the character reaching a particular location or using a particular item, a particular keyboard/mouse button being pressed, a countdown timer running out, etc.). For every event, something must generate it, and something must handle it (by, say, modifying the character's status, making the character do something, modifying enemies, or modifying the world itself).

Your task is to implement such event handling using the Observer Pattern. One of the event types you must handle is the collision between two game-world objects (as mentioned).

However, you will also need to pick:

- Two other different kinds of events within the game (apart from object collisions).
- Two different classes within the game that may cause events to happen. (Combined, the two classes should be responsible for causing all three event types.)
- Two different classes within the game that would react to these events. (Combined, the two classes should react to all three event types.)

  Assume that all event-handling classes will need to know the details of the event when it happens.

(There is a large scope here for you to imagine what kind of game this is, and how it may be structured.)

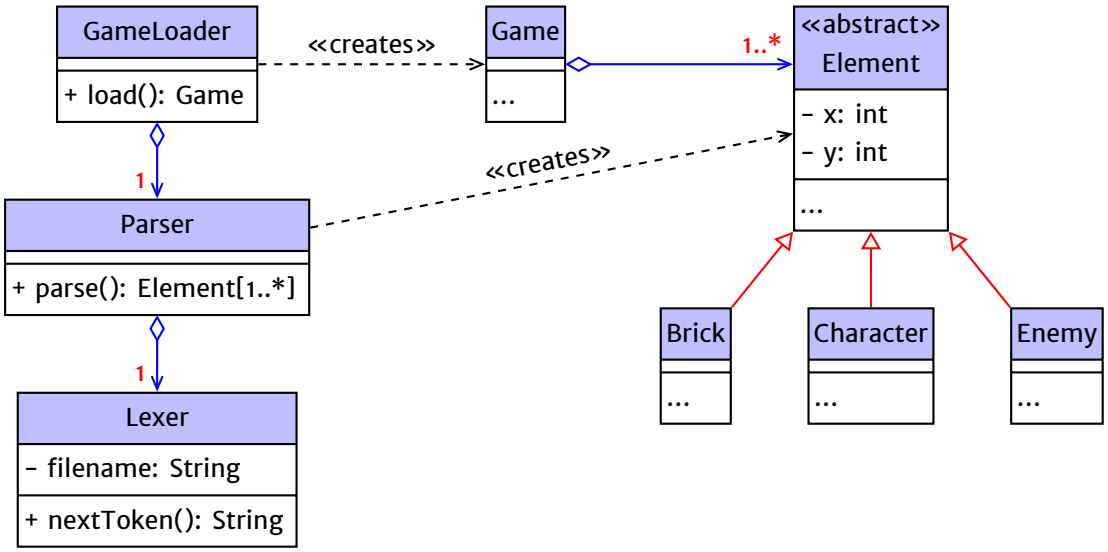
To answer this question, *write the code* for each class/interface. You may use Java, C++, Python or C#, but avoid the use of duck typing. It is not necessary to provide the UML.

Show all relevant fields and methods within each class/interface. You *do not* need to show exactly what happens, in the actual event handling code, but do describe it briefly as a comment.

Question 2 appears on the next page

Question 2 (20 marks)

This question concerns dependency injection, factories, and error handling. Say you have the following class structure:



This is the preliminary UML. You will need to introduce at least one additional class, and refactor some of the relationships, in order to answer this question.

Here's what's going on:

- `GameLoader.load()` has overall responsibility for loading a saved game from file.
- A `Game` object has several `Element` objects, each representing an element of the game. There are three kinds, represented by the subclasses `Brick`, `Character` and `Enemy`.
- The input file format contains the word "Brick", "Character" or "Enemy", then a value for x, then a value for y. This pattern repeats for each game object; e.g.:

```
Character 5 10 Enemy 1 1 Enemy 10 1 Brick 0 0 Brick 3 9 Enemy 5 3
```

- `Lexer.nextToken()` directly reads the file, or rather the next word or number, which it returns as a String. It returns `null` instead when it reaches the end.
- `Parser.parse()` gets tokens and creates a list of `Element` objects accordingly.
- `Lexer` and `Parser` may encounter errors. If so, a `GameLoadException` must be thrown. Nothing within this set of classes needs to catch that exception.

Your task is to implement the `GameLoader` and `Parser` classes using dependency injection. To do this, as you can hopefully see, you also need to create a factory (or factories). The classes shown above should not be considered factories themselves. You can create `GameLoadException` objects normally, though, without factories or dependency injection.

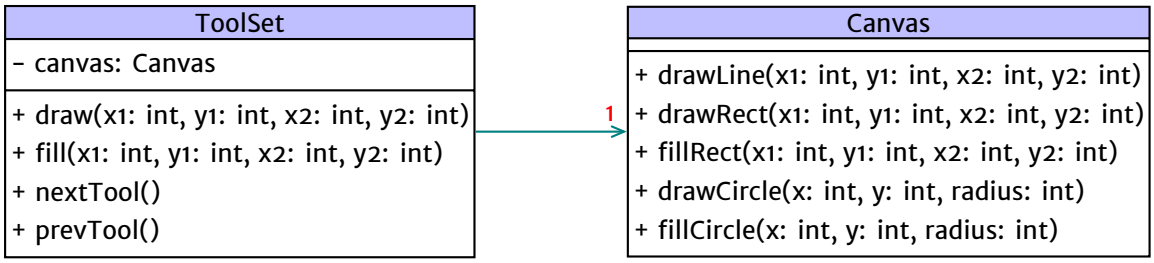
Provide the complete code for both `GameLoader` and `Parser`, and the factory(ies) (but not the other classes). Make reasonable assumptions about the design of the other classes.

You may use Java, C++, Python or C#.

Question 3 appears on the next page

Question 3 (20 marks)

This question concerns the State Pattern. Consider the following classes:



As you might guess, these are part of an image drawing program.

`ToolSet` keeps track of what "tool" is currently active. This is its state. There are three tools in the current design: line (the default), rectangle, and circle. It then calls the appropriate method in `Canvas` to perform the actual operation when requested. `ToolSet` can be told to move forwards or backwards, in terms of which tool is active (`nextTool()` and `prevTool()`).

Each tool is capable of *drawing*, where an outline is shown, and also *filling*, where the shape is solidly coloured-in. Well, this is true for rectangles and circles, anyway. There is no conceptual difference between drawing and filling a line, and `Canvas` only provides one line operation to cater for both cases.

In all cases, when the user wants to draw or fill something, they click and drag the mouse from one point on the screen to another. (x_1, y_1) is where the mouse was first clicked, and (x_2, y_2) is where the mouse was dragged to.

- For lines, these points represent the start and end of a line.
- For rectangles, these points represent two opposite corners of the rectangle.
- For circles, (x_1, y_1) is the centre of the circle, and radius $= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Your task is as follows:

(a) Implement `ToolSet` using the State Pattern.

Provide the complete code for `ToolSet`, any interfaces, and *one* of the state subclasses (your choice). **[14 marks]**

(b) For each of the following cases, describe how you would need to modify your design. (You do not need to write any additional code. A plain-English explanation will suffice.)

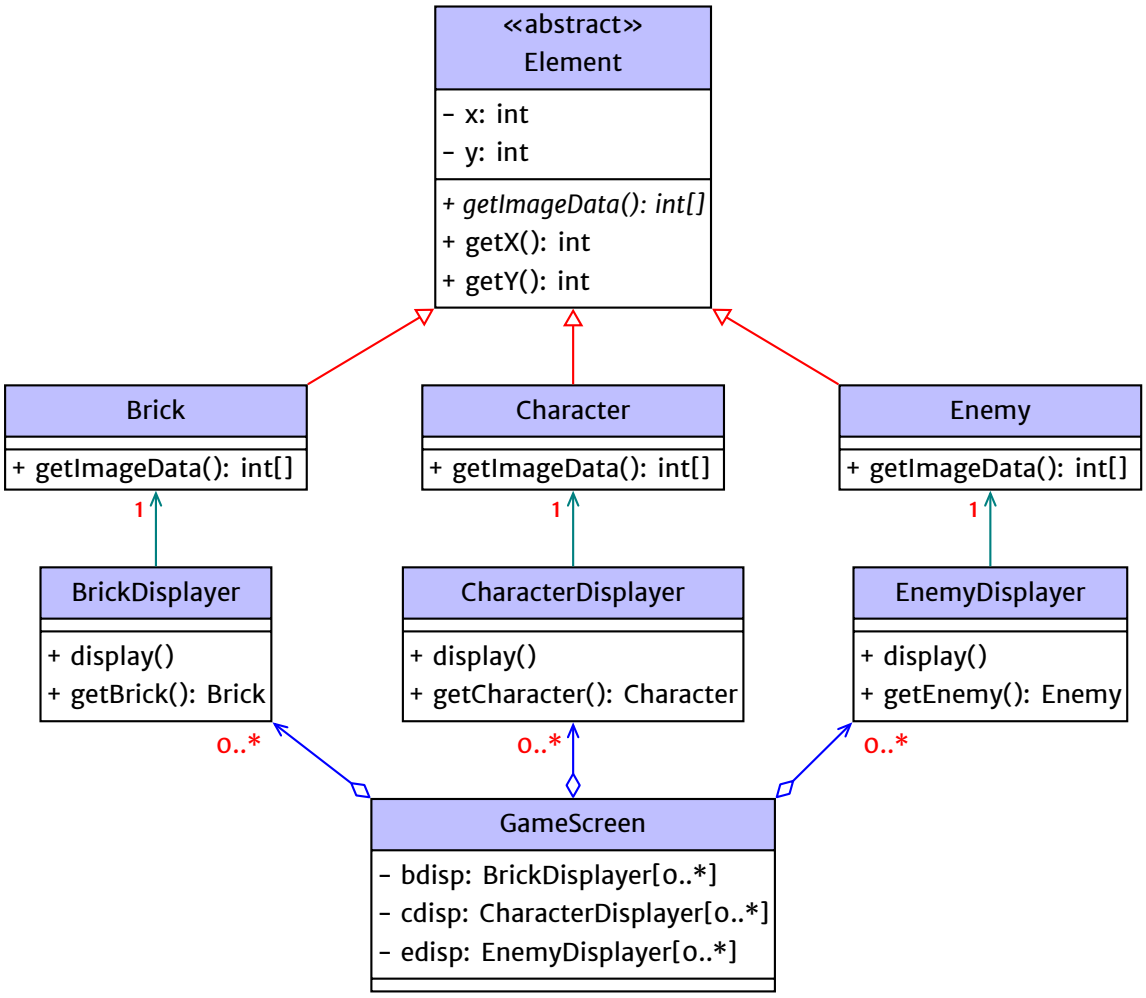
(i) Adding another operation, `gradient(x1, y1, x2, y2)`. Assume we've already added `gradientLine()`, `gradientRect()` and `gradientCircle()` to `Canvas`. **[3 marks]**

(ii) Adding another tool that represents a different kind of shape. Assume we've already added the appropriate methods to `Canvas` for drawing and filling this shape. **[3 marks]**

Question 4 appears on the next page

Question 4 (10 marks)

This question concerns generics. Consider the following class structure. (The classes are similar to ones in a previous question, but there is no connection between the questions.)



`Brick`, `Character` and `Enemy` each have an x and y position, and can return an integer array representing pixel data (i.e., what they look like).

Each `...Displayer` class is responsible for displaying a particular element type on the screen. For each, the `display()` method works in the same way, retrieving x, y and the image data from the element object, and actually displaying it. Each also has an accessor that returns the corresponding model object. (The accessor is important, for reasons to do with the rest of the system design.)

Identify the problem that generics can solve in this situation, and refactor the design accordingly. Explain the changes, and use Java or C# to show how each affected class would need to change. (You cannot use Python or C++ for this question.)

You do not need to write the contents of methods or constructors.

End of Test