CE203 Game Report

**GRID OF FILLED SQUARES:**

I made a “SquareGrid” class that extends JFrame, I called the variable names “Rows” and “Columns” and set them to private static final int since the values don’t need to be changed once instantiated. I then create a private List called grid, and have the type parameter as the JFrame, and initialise it as an ArrayList. I didn’t use the 20x20 grid as I didn’t need to use a grid for my game; since I was doing Pong, I felt the grid layout would affect the fluidity of the ball moving, and wouldn’t be the correct dimensions since its more elongated vertically. My current JFrame in the game does include my name and registration in the title.

**SHAPES IN THEIR OWN CLASSES:**

I first created an abstract class called shapes that will hold all the types of shapes the report specified we should draw, that held common parameters that occur in most of the shapes. Inside the abstract class there are class’s for each shape, and each class has a constructor and paint method. I then have a class called “Canvas” that extends JPanel and calls an instance of another class created earlier that has a JPanel instantiated, then the paintComponent method is called to paint every shape to the JPanel. I added a main function to test the class. I created all shapes besides the Pie shape. I didn’t use the shape class in my game, instead opting to use 2d graphics and creating the pads and ball in the game java file.

**EVENT HANDLERS:**

I created a “Handler” class that had event handlers for key events and mouse events. For the arrow events, I used a switch case for getting keyCodes, and then had 4 cases, one for each key (up, down, left, right). My second “Handler2” class handles events for mouse related events, including mouseClick, mouseReleased, mouseEntered, mouseExited and mouseClicked. I didn’t use my handler classes, however did make use of a “keyPressed” and “keyReleased” methods in my game that used similar principles.

**RECORD OF SCORES:**

I made a “Scores” class that records the score of the bottom and top pad (or AI pad). It creates a new scorefile if it doesn’t exist and uses a fileWriter and BufferedWriter to write the score to the text file. It also contains a method to sort the top 10 scores and uses collections to sort them, though my program throws an ArrayIndexOutOfBoundsException and this hasn’t been fixed, so it doesn’t quite work. I also have failed to create a method that records the time associated with the 10 scores.

**SHORT REPORT:**

My short report will describe each stage of the mark scheme, and how I wrote code to capture the requirement asked, and then how/if I implemented it inside my game. My code features comments and correct indentation.

**USE OF OBJECT-ORIENTED PRINCIBLES:**

My project is split into 4 .java files, and so makes use of object-oriented practises. Main.java handles all aspects about the frame and creates a new instance of the game that can be ran. As the specification requires, all the shapes are all stored in their own collection. Scores, writing to file and sorting the scores are all encapsulated in their own java file. Finally, the game is in its own .java file. The Game.java file makes use of object-oriented techniques such as encapsulation through the use of private variables and inheritance from the package and extending the JPanel and implementing actionListeners.

**MY GAME:**

The game I chose to make is ping pong; the score should increment on every ball you return at the opposing AI pad (which just mirrors the movement of the ball, and is not possible to beat). The game should end once you miss the ball, however I haven’t implemented this in my game, nor have I displayed the top 10 scores of all time on death, with timestamp. It makes use of getting score through inheritance from the Scores.java class, uses keypress and keyRealsed with 2d graphics to repaint the position of them, uses basic AI that moves left or right to return ball dependant on ball position relative to the pad. The score is displayed on the left and right of the centre of the JPanel, though it doesn’t update. Before the game is instantiated, all the variables applicable to the game are declared private, and mainly relate to the velocity of ball, movement speed of pad, size of the pad, and delay of the timer in the game.