**Lab 13**

Goal of this is to implement a small game! In the process, you will use the Timer class and use mouse listeners as well as menu items (bonus).

Let’s start!



Try to complete all the tasks in a sequential manner.

**Task 1** Drawing Mario on the screen and moving Mario with the mouse

**Implement the following JAVA classes:**

1. Make a new class, MyFrame which extends JFrame. Set the size to 800x600. Add a ContainerPanel object of class described below.
2. ContainerPanel.java:

Should extend JComponent

This class should implement *MouseMotionListener* to capture the mouse movement on screen. In particular, we want to use the *mouseMoved* method.

Inside *mouseMoved* method, store the x and y of the MouseEvent object. Call these variables, mouseX and mouseY.

For instance, if your implementation is:

@Override  
**public void** mouseMoved(MouseEvent event) {

}

then event.getX() gives you the x coordinate of the mouse and event.getY() gives you the y coordinate.

You want to store the x and y and *repaint*. We will be using our mouse movement to draw Mario on the screen. To do this, implement paintComponent. Inside, you will read the following image: <https://drive.google.com/file/d/1eP04lJXRC2lSmj0g36MF0QyO3KMx0TMk/view?usp=sharing>

And read and draw it on coordinates mouseX and mouseY. To read an image, you can simply do:

BufferedImage image = ImageIO.*read*(**new** File(**"mario.png"**));

ImageIO.read throws IOException so you would need to handle that.

You should now run your program and you should be able to control Mario with your mouse.

Hurray!

But wait, this is quite boring. Let’s add some game play. The overall idea is to add obstacles which Mario has to avoid! We will add bombs but you are free to add more!

**Task 2**

Add an Obstacle class. It should have the following member variables:

* x (the x coordinate)
* y (the y coordinate) c
* dX (translation in x)
* dY (translation in y)
* image

Write a constructor which takes in all these parameters and initializes them.

Write a move method, which changes the x and y coordinate by adding dx to x and dy to y. Make sure that if you reach the boundary, it should bounce back. To do this, you need the width and height of the MyFrame. Go back to MyFrame and make them static to be able to access them.

Go back to ContainerPanel and add an ArrayList of Obstacles. We will just use 4 obstacles (bombs) for now. You are free to decide how to initialize them (the starting x, y, dx, dy) but they should start at different locations and have different directions. Use this image for the bombs: <https://drive.google.com/file/d/1-h6OMeOJbuLCLWoOvoAx2YJwOAVayBK3/view?usp=sharing>

Inside paintComponent of ContainerPanel, draw the each of the bombs.

Run the program and you should see your bombs, like below:

A screenshot of a video game

Description automatically generated with medium confidence

Currently, they cannot move. Lets work on moving them:

**Task 3** Moving the obstacles, and implementing the game play.

Implement a Timer class and start it. Inside the action listener, for each obstacle, move the obstacle and call repaint. If your move method is correct, the bombs should bounce off the walls.

The game objective is to avoid the bombs. So as soon as Mario touches an obstacle, the game should end.

To do this, check the distance between each obstacles x, and y and mouseX and mouseY (compute the Euclidean distance). If the distance is less than a threshold, the game should end. Estimate what the distance should be.

Think of how you the game should end. Hint: use a variable to signal if the game is ongoing or not. If it is, then paintComponent should work as before. Otherwise, it should draw a string saying you “Game Over”.

**Once you are done, show your work to the TA to get full credit.**

**Extra enhancements:**

Inside MyFrame add this line to not show the cursor:

setCursor( getToolkit().createCustomCursor(  
 **new** BufferedImage( 1, 1, BufferedImage.***TYPE\_INT\_ARGB*** ),  
 **new** Point(),  
 **null** ) );

**Bonus features:**

1. Write a score on the top left depending on how long the user survives.
2. Make the game more difficult as time goes by. To do this, you can either
   1. Increase dX and dY of each obstacle with time.
   2. Add more obstacles.
3. Let user have a number of lives. Each obstacle decreases life by 1. You could also have good obstacles that increase your life by 1.