**Lab 2**

**OOP JAVA**

* Practice drawing stuff and working out co-ordinates
* Practice using variables and if statements in Java

This is a video of a silly game called Bugzap made in Java using the Processing libraries. Today we can make a part of this game in order to get some practice using variables in Java.

[](https://www.youtube.com/watch?v=s6PA8jtWneQ)

How you should do it:

Ok let's get the main game working first and not worry about the splash screen and the game over screen

* Update your fork of the repository from the master branch. To do this, cd to the folder where you have cloned *your* repository, (or clone it somewhere if you need to) and type:

git checkout master

git pull upstream master

* Create a branch for your work today by typing:

git checkout -b lab2

* Create a new class called BugZap.java in the ie.tudublin folder. Make it extend PApplet and add the settings, setup and draw methods. Check out HelloProcessing.java if you need examples for these.
* Call size in settings to set the size of the drawing window.
* Edit the file Main.java so that it starts the BugZap class instead of the HelloProcessing class. I'll let you figure out how to do this.
* Make sure everything works by compiling and running your program before continuing!

Now we can draw the bug.

* Make fields of type float in the BugZap class for playerX, playerY and playerWidth and give these default values. You can decide what these should be. There are built in variables called width and height that give the width and height of the drawing window. These only get assigned after size has been called, so if you want to use these to give values to playerX, playerY etc. put the code into the *setup* method.
* Write a method called void void drawPlayer(float x, float y, float w) that draws the player character, centered around the parameters x, y. You can use the line method to do this. You can pass variables as parameters to this method and also things like x + 20, w \* 0.5f etc. I made an extra variable in this method called h for the height and set it to be half the w parameter. Don't forget to set the stroke color!
* Call this method from draw, passing in the parameters playerX, playerY and playerWidth.
* Compile and run everything to make sure it's working before continuing.
* If everything is working ok, you should see the bug on the screen

Now lets get the player moving in response to the keys

Add this method to BugZap.java:

public void keyPressed()

{

if (keyCode == LEFT)

{

System.out.println("Left arrow pressed");

}

if (keyCode == RIGHT)

{

System.out.println("Right arrow pressed");

}

if (key == ' ')

{

System.out.println("SPACE key pressed");

}

}

* If you compile and run the program again you will see that some messages get printed out when you press various keys. If you are running in Visual Studio Code, you will see these messages appear in the Debug Console
* Modify this method to increment and decrement the playerX variable instead and you should be able to get the player to move left and right
* You might want to add if statements to this method to stop the player moving off the left and right side of the screens. If statements in Java are almost the same as in C!
* When the player presses SPACE you will want to draw a line for the player's laser.

The Bug

* In a similar way to how you made the player, make the Bug. Make variables and drawBug method. Don't forget to call the method from draw()
* To move the bug you can add a random amount to it's x coordinate on intervals. To generate a random number you can use the [random](https://processing.org/reference/random_.html) function.
* One way to make stuff happen on an interval rather than every frame is to use the frameCount variable. This variable is a field in PApplet and it gets incremented automatically every time draw is called. Because draw gets called 60 times a second, you can do something every second with this code:

if ((frameCount % 60) == 0)

{

// Do something

}

* You can print text to the screen using the [text](https://processing.org/reference/text_.html) function.

Ok you should now have the basics working. See if you can figure out how to check to see if the player hits the bug, add scoring, splash screen, game over screen and sound.