**Java Technical Report**

**Pong**

**By Joseph Batchelor**

**Table of Contents**

**Introduction**

The java project I decided to do was to create my own game application based on the original game, “pong”. The reason I chose to create my own 2D game was the fact that I had more freedom which has lead me to create an app from scratch, allowing the implementation of my own designs and ideas as well as to use what is needed from the criteria. If you are not familiar with Pong it is a simple 2D game that simulates a virtual game of ping pong, this involves two paddles hitting a ball back and forth until points are scored and a winner has been named. The reason I decided “Pong” is because it is like breakout in terms of paddle and ball, I felt that they would share somewhat similar functions however allowing me to introduce more features. My overall aims for this project can be seen below.

|  |  |
| --- | --- |
| **Aims** | **How I will achieve it** |
| To meet the criteria and your various statements within my project. | The way I will achieve this by introducing the necessary statements such loops and if statements to help my program function. |
| Introduce object-oriented programming within my project. | The way I will meet this aim is by adding encapsulation, overriding, and overloading as well as inheritance. |
| Use different data types within my project. | This is an easy aim to do as to achieve it I will need to introduce different data types such as short or long etc. |
| Have a functioning game. | By doing work and staying on top of it, my project should have an application that works. |

**How to run code**

The application works by Running Class Main this will call all the needed methods from class Models as well as the Necessary Methods from class function. Once the application runs you will be presented with two windows one will be from class Console which will print necessary text regarding the application. This will display instructions for both player 1 and 2 about which keys to use throughout the game. This will also print keystroke made during the application. Within the other window this will display the game, you will be presented with a title screen that prompt you to enable or disable duo player. This can be done using the Y key for yes and the N key for no. once decided, you will then need to press space to start the game when ready. This will instantly transition to the pitch where both the ball and paddles will be found, this will instantly start the game by the ball moving. Simply the game is first to five points. Every point scored will give a brief break until space is tapped again once either player scores five the game will restart from zero, essentially you keep playing until you close the program yourself.

**How the code works**

|  |  |
| --- | --- |
| **Classes** | **Description** |
| Main | Class Main is the parent class, this class inherits all methods when called from class model and function. This allows for functionality to occur as well as for graphical text and models to display on screen. Class Main is also used for holding Boolean values that can be seen and used within class function to allow user inputs to either the game or introduce different game modes. Class Main is responsible for building for the canvas as well as introducing screen refresh allowing models to move freely without any lag. Class has allowed me to demonstrate the use of Boolean values, calling methods, creating a canvas and key input control for controlling models. |
| Function | Class Function plays an important role in terms of functionality and gameplay. This class implements unique methods such as player tacking, 2 player interaction as well as collation between the ball and the paddle. Class Function allows me to demonstrate the use of if statements as well as loops such for, while and do while. I have also managed to introduce encapsulation with making variables private meaning that only methods within that class can use. |
| Models | Class Models is used create models that can be seen on screen such as the paddles, ball, and pitch. This class allows me to create method objects to be used and designed for in game visualising as well as to be called by class Main. |
| Console | This class allows me display in the console game information so that the user can understand how to play. Once class Main is running this will appear along with the game to inform the user how to play the game and what keys to use. it will also print the key action when pressed or released. I was able to demonstrate method overloading and method overriding as well as using the return statement to print my text. |
| ModelValues,  DisplayValues and MovementValuses | This class was used to store data and variables to be called to either class Models or class Function. Each class held data relevant to the parent class. This class allowed me to demonstrate the data types I was able to implement as well as creating and using expressions and statements. |

Down below is an overview of my project and how it functions. My project is made up of 7 classes that each preform a different and key role to my game. A table can be seen below explaining what each class is and how it is used

I will be explaining how each class functions and how they work with each other. My project uses one parent class which is class Main, this class is designed to run all the methods created by class Model and class Function which are the child classes. Class Main holds all my objects in method start, these objects will be used throughout my project to introduce java imports. The Main function for class Main is to call the necessary methods within class Model and class Function. Class Model and class function both preform separate jobs, but both extends to class Main, allowing class Main to inherit features. Class Model function by using individual methods to create models such as the player 1 and player 2 paddle or the ball model. Essentially it creates these models by using variables from class ModelValues and DisplayValues. Class model calls for the necessary variable need to create the displays seen. Class Function works the same as class Models this class creates methods such as Paddle tracking, methods that provide functionality to the application, these methods use data types and variables from the classes ModelValues, MovementValues and DisplayValues. Finally, classes ModelValues, DisplayValues and MovementValues these classes all function the same in terms of holding data for the child classes. Overall inheritance plays and important part in how my codes run and how the classes work with each.

**Use of data types**

|  |  |
| --- | --- |
| **int** | Int types were giving to numerical values in all three classes I gave these to variables that were fixed meaning that they aren’t altered allowing for correct conversation when called by other classes. |
| **long** | This data type was given to variables that involved the use of more complex / longer expressions as it can hold more bits allowing for larger values to be generated or used. |
| **byte** | Byte data types were given to values that were small (0-127) this used 8 bits which means that this would be the fastest. I used this type for canvas designs as this means they would load first. |
| **double** | This data type was used for values that changes constantly as well as that required precision when adjusted in game. This helped for conversion between classes as well. |
| **Boolean** | This type was simply used for user input simple yes and no scenarios can be found in class main, used for user interaction within the game as well as used within if statements. |
| **String** | String was used to represent text commonly used and found in the class Function, to display text in game and in console. |
| **Short** | This data type was used where byte was not, as byte have a range 0-127 and short would be the next appropriate type to use. |

All my data was held in three separate classes which were ModelValues ,ModelValues and DisplayValues. this made finding and calling variables easier. The following table shows the types used and how I implemented them.

**Implementation**

Within the criteria we were required to introduce various loops, method calls and statements within our code. I have implemented all the required statements in my code, within this section I will be explaining how I implemented some of the many within my code as well as how it functions.

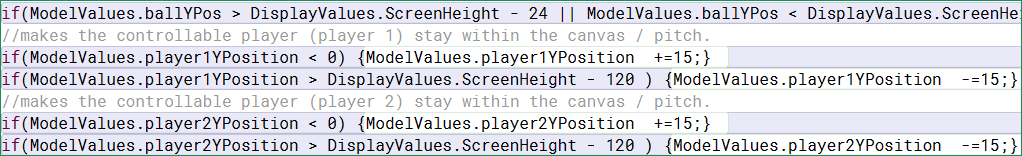
Within my project I managed to implement a for, while and do while loop, to help my game function. Mainly found within the class Function this essentially was used to display graphical text during different moments of the application as well as movement.

A screenshot of a social media post

Description automatically generatedTo the right is an example of code of a for loop which can be found in class Function ,Method Score Display , essentially what this loops says is that int i which is set to 0 is less than the size of numbers which is an array list. I then included an if statement which states that if the player 1 score is equal to i then it will display text on screen which would the score of players 1. Because player score and i start both at zero this means that they will increment at the same time until the limit is met. This can be seen when either player score text will display a value on their side.

Another type of code that I implemented was a do while loop, this works by preforming a task until a condition is met. The example here is the int variable “one” is equal to the player score, so every time the score is equal to a value it will display the desired text. However, by introducing an if statement I was able to say, that if the score was equal to 5 or less than 4 ignore until the score equalled 6. This worked because I used the continue statement which essentially tells the loop to on work beyond that condition.

A screenshot of a cell phone

Description automatically generatedAn if statement example can be seen below. This statement was in class Function, method Container and this method was designed to keep the models within the canvas. For this example, it is player 1 model. Essentially, this statement states that if the player1 Y position was to exceed 0 (which is the top) then add 15 to that player 1 Y position. The next statement says that if the player1 Y position was to exceed the height minus the paddle width then subtract 15 from the y position. The reason this work was once the Y position reached either condition the movement is prevented by adding the opposite to the increment value which 15 to create a net value of 0.

A close up of a hand

Description automatically generatedAn implementation that was require within the criteria was the use of encapsulation. The way I did this was by declaring variables of a class to private. This meant that these data types can only be accessed through methods within their current class. I applied to Class Function and Class Console. As you can see encapulation was applied to the function class with the top image using to to prevent othe rclass from altering the way the ball travel and below the images prevetns other classes from altering text that can be found in class console.

A screenshot of a cell phone

Description automatically generatedA feature I felt that was unique was adding the ability for the user to play against a friend or the computer this was an easy function to implement the way this works is by using Boolean values that can be found in class Main. The two values used “Enable” and “Disable” both equalled false. As part of my keycode which allows for keyboard user input which can be found in class Main , I bound Y to “Enable” and N to “Disable ” and set a change value to true once that key is pressed. This meant that if enable was pressed no code would be executed allowing the user to control both paddles as this a default opinion. However, if Disable is true then it will run code that will override the default option allowing the paddle to track the ball to simulate a player versus computer scenario, preventing the user from control player 2. A data control flow diagram is made which can see later within this report.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedAn additional implementation of OO programming is introducing method overloading. This technique can be found in class Console. This allowed me to use multiple of the same method within in a class.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedTo the left you can see how the method name remains same however, the parameters are different in the four examples to the left. The reason I implemented this in class console is since Class Console must return multiple variables to print relevant game information in the console. This gave me the opportunity to introduce the idea of overloading methods. I managed to introduce multiple data types with the single use of one method name.

Inheritance plays an important part within my project as the parent class (class Main) uses the methods and data from the child class such as Models and Function. My projects use a hierarchical inheritance to create an application. Class Models and Function and Console create the methods that help allow the game to run and look as it is they all extend to Class Main which calls for the methods as well as allows for Boolean values and other variables to be used within child classes. Below you can see how I implemented this feature below. The image to the right helps visualise the each one works , as you can see in the red circles.

A picture containing drawing

Description automatically generated****

|  |  |
| --- | --- |
| **Java imports** | **Description** |
| java.util.Random; | Used to generate a sequence of numbers. |
| javafx.animation.KeyFrame; | Controls methods for the interval between frames . |
| javafx.animation.Timeline; | Used to define a free form animation of any method or variable. |
| javafx.application.Application; | An entry point for JavaFX applications is the Application class. |
| javafx.scene.Scene; | A container for all content in a scene graph. |
| javafx.scene.canvas.Canvas; | Constructor to make images that can be drawn on using a set of graphics commands. |
| javafx.scene.canvas.GraphicsContext; | Used to design and draw and fill models works alongside Canvas. |
| javafx.scene.text.Font; | The Font class represents fonts, which are used to render text on screen. |
| javafx.util.Duration; | A class that defines a duration of time. |
| javafx.stage.Stage; | JavaFX container |
| javafx.scene.Group; | A Group node contains an Observable List of children that are rendered |
| javafx.scene.paint.Color; | The Color class is used to encapsulate colors in the default sRGB color space. |
| javafx.scene.text.\*; | The Text class defines a node that displays a text. |
| java.util.ArrayList; | Resizable-array implementation of the List interface. |
| java.util.\*; | contains collection framework, collection classes, classes related to date and time, event model, internationalization, and miscellaneous utility classes. |

This section only covers some the code found in my project. I have implemented more unique code and features which demonstrates my coding skills and ideas. Below you can see the implementation of all the java imports used and what they do briefly.

Links to each of the descriptions for each import can be found within the reference page below.

**Data control flow in classes**

A close up of a sign

Description automatically generatedThe flow of data within my project was implemented in a way that not only made it easy for myself but for others to understand. In its simplest of form data such as expressions and statements are held in the lower classes and are called to the classes above. Essentially working with inheritance, this can be explained with the examples below.

To the left you can see a segment of my project , with this you can notice that its broken into three levels of classes the bottom class ModelValues this class hold of values and data types that are related and needed to the Models class. The models class used different methods to create the different Models used such as the paddle, ball, and pitch. Each method used calls upon the necessary variables needed to draw the models. Simply using a call variable “*ModelValuse.name*”. Once I have made and designed everything and all methods are completed. The next inheritance which class Main which is the parent class is done by using the extend feature which means that it inherits methods from class models. Class Main uses a method call instead of calling variables, class Main is calling methods from Models allowing for these models to be displayed on the canvas and to function. Briefly below shows the process.

Class ModelValues holds data.

Class Main inherits and calls methods

Class Model calls for variables and data

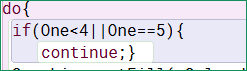
Expressions and statements were an important part of the functionality of my application, as without the uses of these my application would not have worked. All my expressions and statements can be found in classes ModelsValues , DisplayValues and MovementsValues each holding data types as well as expressions for the creations of models and functionality. Below you can see some of the expressions and statements used to implement models onto my canvas as well as help with precision when introducing function.

**Data control flow in statements**

A screenshot of a cell phone

Description automatically generatedThis section will be explaining the flow of data within statements. This involves jump statements such continue and break or selection and iteration statements. Through my project the data control flow is being used throughout. Below are some examples of it being implemented that I will be explaining how the flow of data works and what it does.

The example to the left is using a switch for user control input and break statement which is a combination of selection and jump statements. The way this works is the switch holds the multiple blocks of code to be executed in this example my paddle control, using key presses to move the paddle (adds15 to player x position ). Essentially once that key is pressed this creates an execution of code or in this case a flow of data to occur , when the key is not pressed then data isnt used and the paddle remains in its position . Now with the implementation of the break statement this stops the execution to be looped and allows for data to flow once or until the key is released once the break occurs this stops the flow of data until that key or another key is pressed again and ecstatically starting from the top of that statement.

The next example is the use of a jump and iteration statement. The way the flow of data works is the loop is executing code until the condition is met, this means that data is flowing until told to stop. In this example the data that is being used is variable ”*one*” which is player1 score, as this increases this produces values on screen to appear by using a continue statement I have said that if player 1 score is less than 4 or equals 5 then skip or continue. This will display text when the player 1 the score is equal to 4.

A picture containing drawing

Description automatically generated

Another selection statement which can found in Class Function line 370 method “Player2Tracking” this was used within my code to enable or disable two player interaction. An image can be seen above in the implementation section which explains in detail. This is a perfect example of conditional data flow which can be seen to the left.

I have made a data flow diagram to easy so how data is used to make decisions. As you can see the blue arrow is to simulate how data travels within the statement. This data starts with the method, once this is called to class Main, data is sent through to the first condition if Enable is equal to true then no code is executed as its left blank.

**If statement**

**Condition**

**Code executed**

However, if condition Disable is true then if will execute this block of code which essentially makes the paddle of player 2 track the ball to simulate a player verses computer scenario.

**Testing and JUnit and debugger**

**Black box testing**

I have five tasks that I wish to test using the black box method. The reason I wanted to use Black box testing was for the fact that anyone reading this code will gain a clear insight of the what different processes should do.

**Task 1** Running class main andchecking the Boolean value “start” works, this value enables the game to start as well as to transition between rounds. The Value start is key bound to space so that once clicked its equal true , which should enable the method “Function”.

**(***This task was easy as the option to use space is presented straight away. Simply pressing the key will initiate the game, as you can see below pressing the space key initiates the transition from the main screen to the game.* **)**

**A picture containing drawing

Description automatically generatedA close up of text on a black background

Description automatically generated**

Space key Pressed

**Task 2** is running the application to see that if Player1 obtains a score of 5 , this will display the “p1Win” method which should display on screen text saying player 1 wins.

**Task 3** is to see that once player2 or the computer reaches a score of 5 , the method “p2Win” should display player2 wins.

**(***Task 1 and 2 can be both be completed by simply playing the game as normal or by changing the initial starting integer from 0 to 5 which it the process I decided to choose as it’s a faster way of doing this below shows the transition , when the statement are met.***)**

A picture containing sitting, laptop, food, holding

Description automatically generatedA picture containing drawing, clock

Description automatically generated

Player1 scores 5

A screenshot of a cell phone

Description automatically generatedA picture containing clock

Description automatically generated

Player2 scores 5

**Task 4** Run class main and see if enabling player 2 or computer works. Check if Player2 follows the ball or requires user input.

**(** This task *involves method “Player2Tracking” this task makes sure that once the Boolean value “Disable” which is key N is false or the Boolean value “Enable” which is key Y is true. It will decide depending on the input of whether to introduce a method for Paddle and ball Tracking simulating a computer vs player game or present two Player .Once the N key is pressed this enables Player vs Computer , however if the Y key is pressed this enable player vs player. This process is easy as I have set the Boolean values when the keys are pressed, I simply ran the application until prompted to preformed one of these actions overall they both worked effectively.***)**

**Task 5** is running class main and to see that the ball model remains within the canvas.

***(***Whenclass main is ran this will prompt you to start the game , once started the ball model will begin to move , if the ball is able to hit the top and bottom without leaving the canvas then the ball will remain within the canvas , this uses the method “container”.***)***

**A screenshot of a cell phone

Description automatically generatedJUnit**

Another testing method used is Junit which is a framework for java. This was used within my project to make sure that values used within my application were correct. As you can see to the image to the left you can notice that I used Junit to test my classes that used variables. I have created methods in test classes that corresponds to the use of these data types. From this outcome it says that all values are correct. The way I check if values are correct is by using a feature that checks the data types as well as the values.

Below you can see what it looks like when one of the assert conditions are not met, you will be presented with a failure message stating what is expected.

**A screenshot of a cell phone

Description automatically generatedJunit Failure**

**A screenshot of a cell phone

Description automatically generated**The image to the left is a failure that occurred when using Junit, this failure is simply used to state an incorrect value in my test class. As you can see within the red circle you notice that it displays the value expected to the left and the value that was asserted to the right. Within this scenario both were different values meaning that Junit when complied the test class with display a failure as well as the correct value needed. Using Junit was helpful as it allowed me to maintain values during development as well as this prevented me from making mistakes with concurrent data / variables. Below you can see how manage to check value using the Junit features.

**Debugger**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**An additional method of testing is using the java debugger to inspect my code for my project. Below are images of some of the areas when using the debugger tool. The first method to be called within the sequence is the method Visual which can be seen in class Main essentially this method holds are functionality and model designs within my application. Using the debugger, I can confirm that all methods are being used and called. Additionally, we can see that the static variables being used are also correct as the debugger states the varibles used as well as their current condition. When used with Junit this helps confirm that values are set correctly.

A screenshot of a cell phone

Description automatically generated

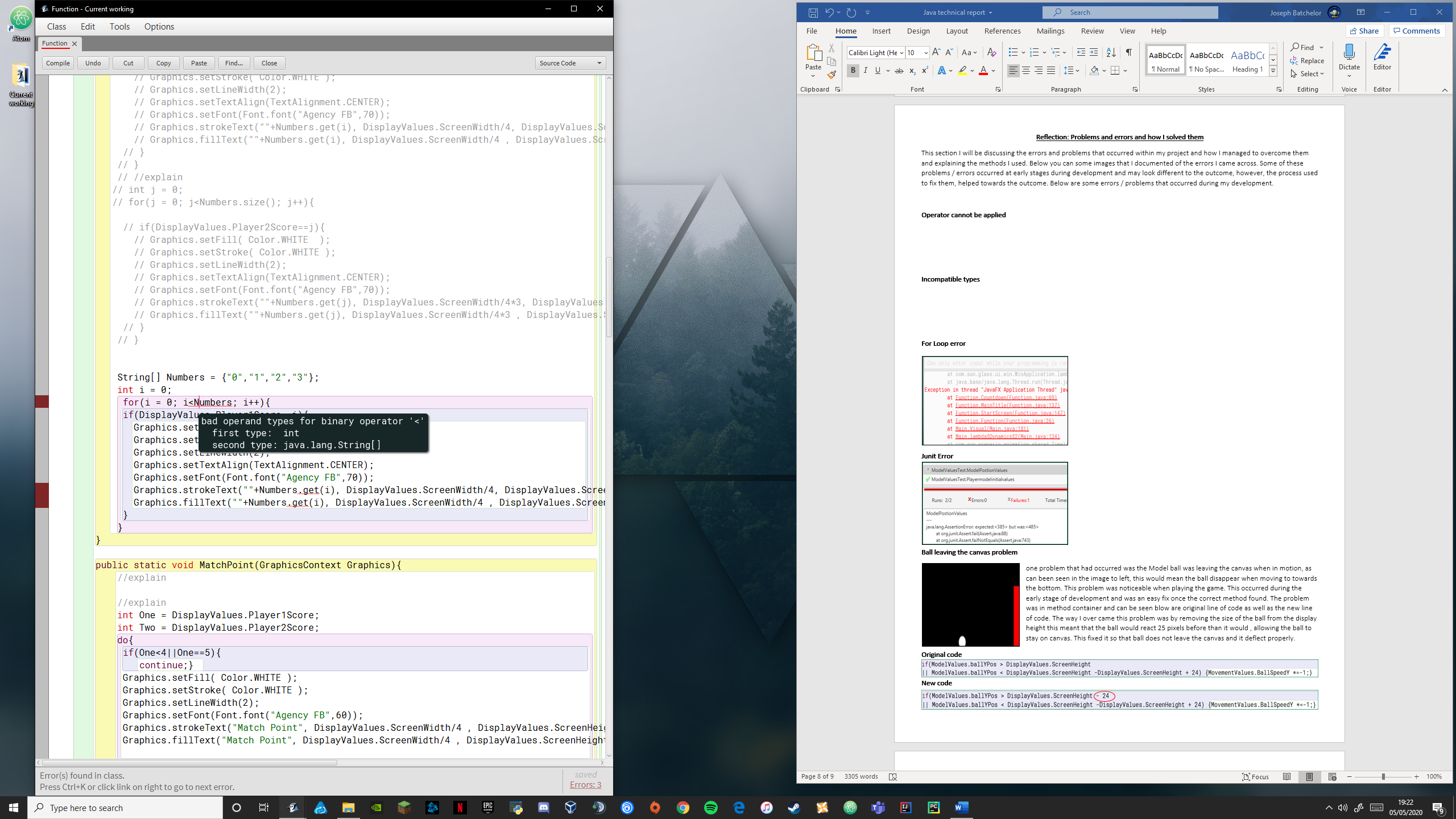
The image to the left demonstrates how the debugger works, by setting a break point just on method Dynamics in class Main I was able to check the function of the Timeline constructor when I use the jump in feature , which allows me inspect the method . This constructor creates its own varibles with the information it has been given, these instances can be seen in the image to the left. The debugger was able to provide me additional information about the method which is helpful for understanding objects in more detail.

****A screenshot of a cell phone

Description automatically generatedA line of code which can be found within the method Dynamics essentially, I want to show you how the constructor creates its own varibles and instances. The value Cycle count can be set to various values however in this case I have set the value to indefinite , which in this case means continuous. By implementing this feature the objects automatically converts the term into a variable that is protected which is equal to a large number as you can see. This means that my application will refresh its frames for as long as the application is running. It is an interesting and useful feature to see how information given to methods can converted and used.

**Problems and errors and how I solved them**

This section I will be discussing the errors and problems that occurred within my project and how I managed to overcome them as well as explaining the methods I used to fix them. Below you can some images that I documented of errors I came across. Some of these problems / errors occurred at early stages during development and may look different to the outcome, however, the process used to fix them, helped towards the outcome. Below are some errors / problems that occurred during my development.

**Bad operand types**

This error occurred in Class Function method Score Display this involves using an arraylist to hold a collection of Strings so that text could be displayed easier using a for loop. The error occurred when using an arraylist as a numeric type. I overcame this by using the method which is. size() which converts the length of the array list into an integer while having the String type.

**Incompatible types**

This error type occurred most often when using the Graphics Context import features such as text. As this would occur when using Array lists and or other collection methods it would sometimes declare an Incompatible data type. For me to overcome this error I simply used other collection methods as well as other ways to store information, by using different types of loops to help me in some areas. This error mainly occurred within Class function.

**A screenshot of a social media post

Description automatically generatedFor Loop error**

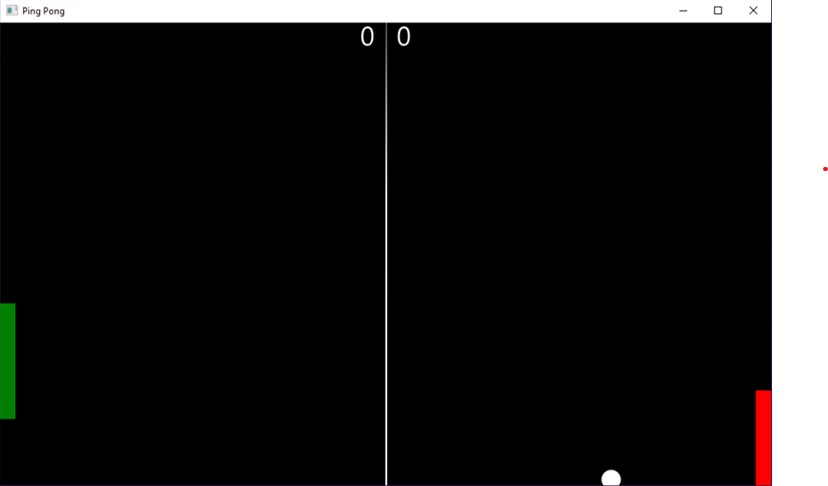
To the left is an image of an error that occurred when implementing a for loop in class function. The problem occurred when compiling this newly implemented code which can be seen below. The reason the application displayed this once complied is because a condition within the loop was set incorrectly, causing the application to freeze and then crash. I found out that I had entered the condition wrong and that it was beyond what was stated. Below you can see the original and the newly installed line of code.

**Old line of code**

****

**<**

**New line of code**

**Ball leaving the canvas problem**

One problem that had occurred was the Model ball was leaving the canvas when in motion, as can been seen in the image to left, this would mean the ball disappear when moving to towards the bottom. This problem was noticeable when playing the game. This occurred during the early stage of development and was an easy fix once the correct method found. The problem was in method container and can be seen blow are original line of code as well as the new line of code. The way I overcame this problem was by removing the size of the ball from the display height this meant that the ball would react 25 pixels before than it would, allowing the ball to stay on canvas. This fixed it so that ball does not leave the canvas and it deflect properly.

**Original code**

**New code**

**Reflection**

Within this section I will be discussing my overall thoughts throughout making my project. I felt this project was a success as I have met my aims during development as well as added additional features. For example, one of my mains was to introduce different types of data types. Within my application I managed to six different data types which can be found and used within my code. Another aim I met was the implementing different statements that are required within the criteria. I managed to include 6 different types of statements which can be found in class Function. Most if not all my aims were met which allowed me to have a fully functioning application.

Some areas I wished that I implemented was the ability have sound played. This was a feature that was part of the initial plan, however, was scrapped as finding and implementing sound onto my application became a tedious task as finding the right sounds as well as the right format became a problem. Implementing the sound within game was also a problem that took time to figure. Within the end I left the idea of sound out as its implementation did not seem possible. Another feature I hope to have introduced was the ability of importing images using a style sheet. The use of style sheets was never thought of from the initial plan and was a feature thought off after the completion of my application. Hope in the future u can introduce this feature in future projects overall my experience with this project was a success and pleased with the outcome.

**Reference**

|  |  |
| --- | --- |
| **Java imports** | **Website link** |
| java.util.Random; | <https://docs.oracle.com/javase/8/docs/api/java/util/Random.html> |
| javafx.animation.KeyFrame; | <https://docs.oracle.com/javase/8/javafx/api/javafx/animation/KeyFrame.html> |
| javafx.animation.Timeline; | <https://docs.oracle.com/javase/8/javafx/api/javafx/animation/Timeline.html> |
| javafx.application.Application; | <https://docs.oracle.com/javase/8/javafx/api/javafx/application/Application.html> |
| javafx.scene.Scene; | <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html> |
| javafx.scene.canvas.Canvas; | <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/Canvas.html> |
| javafx.scene.canvas.GraphicsContext; | <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html> |
| javafx.scene.text.Font; | <https://docs.oracle.com/javafx/2/api/javafx/scene/text/Font.html> |
| javafx.util.Duration; | <https://docs.oracle.com/javase/8/javafx/api/javafx/util/Duration.html> |
| javafx.stage.Stage; | <https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html> |
| javafx.scene.Group; | <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Group.html> |
| javafx.scene.paint.Color; | <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/paint/Color.html> |
| javafx.scene.text.\*; | <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/text/Text.html> |
| java.util.ArrayList; | <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html> |
| java.util.\*; | <https://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Line number** | **Source** | **Author** |
| Main | 79 | BreakoutJavaFX code that allows for keystroke to be printed , found in class controller line 26. | Roger Evens |
| Main | 156 | <https://www.programcreek.com/java-api-examples/?api=javafx.animation.Timeline> | Unknown |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |