Documentation Paper 2

COMPARATIVE PROGRAMMING LANGUAGES [(1229-2865) 12292865](https://elearning.nbcc.ca/d2l/home/140538)

Assignment: Documentation Paper

Student: Larry Loic Kemmo Tejiokem

Instructor: Stephen Johnson

Due: Monday, November 21, 2022

**INTRODUCTION**

Building this application brought a lot of awareness about these three languages (Java, C#, and Python) and their respective libraries (libGDX, monogame, and pygame). Moreover, this document will be comparing the languages stated above in addition to F# (chosen programming language for the presentation in this course) based on the topics below:

**File Input / Output**

|  |  |  |
| --- | --- | --- |
| Java | Python | C# |
| **public** **void** WriteToFile() {    **try** {  FileWriter myWriter = **new** FileWriter("file1.txt"); myWriter.write(HighScore.toString());  myWriter.close(); System.***out***.println("Successfully wrote to the file.");  } **catch** (IOException e) { System.***out***.println("An error occurred.");  e.printStackTrace();  }  } | def **fileWriter**():    title = open(*"file1.txt"*, *"w"*)  title.write(str(highScore))  title.close() | public void Write()  {  string writeText = Highscore.ToString();  File.WriteAllText("file1.txt", writeText);  } |

|  |
| --- |
| F# |
| Type write() =  Open system.I.O  FileWriteText(HighScore, Assets//File1/txt) |

As shown on the table above, Java python and F# has the open functionality applied to each file before it is opened as compared to C# who doesn’t. Which to my opinion doesn’t make these languages efficient for the file is always accessible (in the case of C#) or never closed (in the case of F#).

**String Handling**

|  |  |  |
| --- | --- | --- |
| **Java** | **Python** | **C#** |
| ScoreFont.draw(batch, "HighScore: " + HighScore, 20, 710); | text\_highscore = my\_font3.render(*'HighScore: '* + str(highScore), False, (0, 0, 0)) | spriteBatch.DrawString(font, "HighScore " + Highscore, HighScorePosition, Color.Black); |

|  |
| --- |
| **F#** |
| let someString = "123"  let someInt = someString |> int // int being any variable |

As it can be observed above, In Java, and C#, it is possible to concatenate with both strings and Integers to form a single message just with the use of the + operator. This is completely different in python and F# where in python although still using the + operator, concatenation must only be of type String, so whenever a message should contain both string and integers, the integers should be passed in a function called str() before the system is capable of reading it. On the other hand, in F#, strings and integer can perfectly be linked together in the same method but only with the use of the |> operator which concatenates both datatypes.

**Parameter passing**

|  |  |  |
| --- | --- | --- |
| Java | Python | C# |
| **public** **void** Draw(SpriteBatch batch, Texture TempTexture) {  batch.draw(TempTexture, rockRectangle.x, rockRectangle.y);  } | def **draw**(Player, PlayerRectangle, textposx, rocklist):  global current\_time  screen.blit(background, (0,0))  screen.blit(text\_welcome, (textposx,10))    screen.blit(Player, PlayerRectangle)  } | void Draw(GameTime gameTime)  {  GraphicsDevice.Clear(Color.CornflowerBlue); spriteBatch.Begin();  spriteBatch.Draw(background, rectangleBackground, Color.White);  spriteBatch.Draw(Player, PlayerRectangle, Color.White);  spriteBatch.End(); |

|  |
| --- |
| F# |
| type Connection(?rate0 : int, ?duplex0 : DuplexType, ?parity0 : bool) =  let conn3 = Connection(300, Half, true) |

As far as parameter passing is concerned, Java and C# are built in a pass information to a procedure way, which implies a call by value, and use of constant value. Also, in order for a method to take in any parameter, its data type has to be specified. On the other hand, python and F# as shown above go by a pass information to a procedure where it is updated before return this explains why to edit a global data parameter in a method in python, it is required to type “global”; followed by the variable. Both these languages do not require any data type specification for parameter passing.

**Object Oriented Programming**

|  |  |  |
| --- | --- | --- |
| **Java** | **Python** | **C#** |
| Rock = **new** rocks(rockXPos, rockYPos, rockSpeed, rock); | rock = rocks.Rocks() | Rocks Rock = new Rocks(rectangle, new Random().Next(5, 8), RockTexture); |

|  |
| --- |
| **F#** |
| let p1 = new Person(name="John", birthDate=DateTime.Now)  **or**  let p1 = Person(BirthDate = DateTime.Now, Name = "John") |

When it comes to Object Oriented Programming, the main difference is that in Java and C#, to be able to create an instance of a class (object), the word “new” is recommended. On the other hand, in Python, no key word is needed. F# surprisingly has the possibility to do both, whereby the key word new can or can’t be applied for object instantiation as long as syntactical components of the language are respected. I find this feature very interesting because it both takes into consideration “old and modern programers”.

**Exception Handling**

|  |  |  |
| --- | --- | --- |
| **Java** | **Python** | **C#** |
| try {  int[] myNumbers = {1, 2, 3};  System.out.println(myNumbers[10]);  } catch (Exception e) {  System.out.println("Something went wrong.");  } | try:   print(x) except NameError:   print("Variable x is not defined") except:   print("Something else went wrong") | try  {  lines = File.ReadAllLines(file);  }  catch (FileNotFoundException exnotfound)  {  Console.writeLine(“file not found exception”)  }  catch (Exception ex)  {  Console.writeLine(“Other exception”)  } |

|  |
| --- |
| **F#** |
| let divide x y=  try  (x+1) / y  with  | :? System.DivideByZeroException as ex ->  printfn "%s" ex.Message |

Java, Python, F# and C# error handling is are all similar in functionality but differ syntactically (java and C#: try – catch, Python: try-Except, F#: try-with). The try usually handles code for the testing of the event whereas the catch handles code for the exception. Each of them has the ability of catching multiple exceptions at the time.

**CONCLUSION**

Building this application in three different languages and documenting research made based on the topics above was very interesting, not only because it helped know more about these languages and their libraries (libGDX, pygame, and monogame), but also because it made us go a mile further to research about these topics in regards to our upcoming presentation which indeed will be very handy for that exercise.

**References**

*C# Programming Guide*. (2022, March 31). Microsoft Learn. Retrieved October 3, 2022, from https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/

*fw\_error\_www*. (n.d.). Retrieved October 3, 2022, from https://www.java.com/en/

*Welcome to*. (n.d.). Python.org. Retrieved October 3, 2022, from https://www.python.org/doc/

*Documentation | The F# Software Foundation*. (n.d.). https://fsharp.org/docs/