Basics of Scala

Vicente De Leon

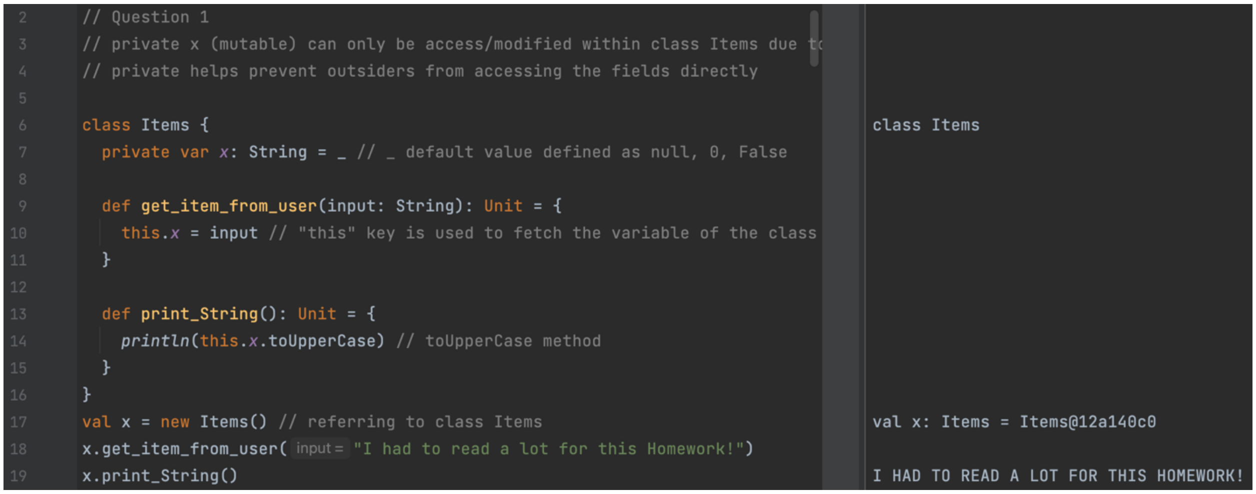
UID: 2001014594

Assignment 2

Note- Canvas: don’t forget to mention ONLY the question numbers (not the text) and write the resources you use to solve the problems.

1. Question 1
   Resources (Please check Scala Worksheet Resources for more detail):
   - https://docs.scala-lang.org/overviews/scala-book/two-types-variables.html#:~:text=The%20difference%20between%20val%20and%20var%20is%20that%20val%20makes,as%20values%20rather%20than%20variables.
   - https://stackoverflow.com/questions/10040886/what-does-it-mean-assign-to-a-field-in-scala
   - https://www.includehelp.com/scala/this-keyword-in-scala.aspx
   - https://www.geeksforgeeks.org/scala-string-touppercase-method-with-example/
   - https://stackoverflow.com/questions/9727637/new-keyword-in-scala

   Observations:
   - Setting x to private (mutable) means that it can only be modified or access within class Item. I t helps prevent outsides from accessing the fields directly.
   - When using “_”, it will assign a default value as null, 0, or False.
   - Keyword “this” is used to fetch the variable of the class.



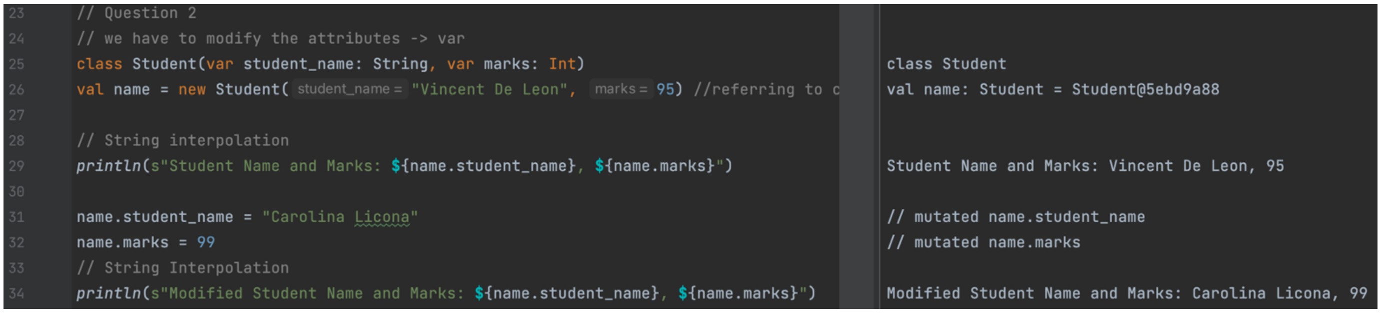
```scala
// Question 1
// private x (mutable) can only be access/modified within class Items due t
// private helps prevent outsiders from accessing the fields directly

class Items {
  private var x: String = _ // _ default value defined as null, 0, False

  def get_item_from_user(input: String): Unit = {
    this.x = input // "this" key is used to fetch the variable of the class
  }

  def print_String(): Unit = {
    println(this.x.toUpperCase) // toUpperCase method
  }
}
val x = new Items() // referring to class Items
x.get_item_from_user( input = "I had to read a lot for this Homework!")
x.print_String()
```

```
class Items

val x: Items = Items@12a140c0

I HAD TO READ A LOT FOR THIS HOMEWORK!
```

2. Question 2

   Resources (Please check Scala Worksheet Resources for more detail):
   - https://leportella.com/scala-iii/
   - https://www.w3resource.com/python-exercises/class-exercises/python-class-basic-1-exercise-9.php
   - https://docs.scala-lang.org/overviews/core/string-interpolation.html

   Observations:
   - We are using var, just like in question 1, because we want to modify the attributes.
   - We are also doing string interpolation using the “$” sign.



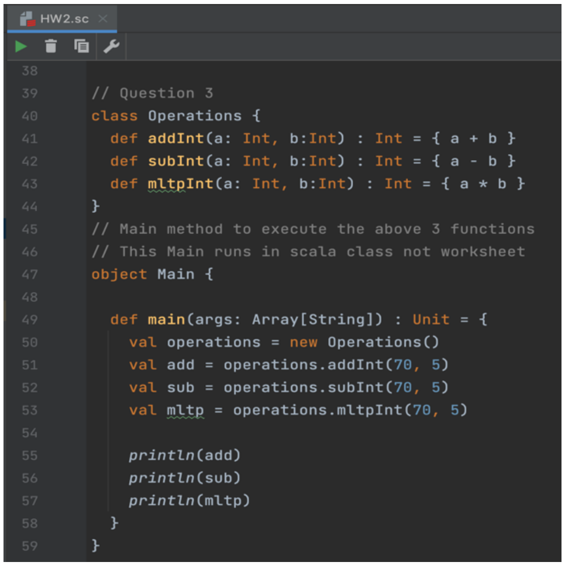
```scala
// Question 2
// we have to modify the attributes -> var
class Student(var student_name: String, var marks: Int)
val name = new Student( student_name = "Vincent De Leon",  marks = 95) //referring to c

// String interpolation
println(s"Student Name and Marks: ${name.student_name}, ${name.marks}")

name.student_name = "Carolina Licona"
name.marks = 99
// String Interpolation
println(s"Modified Student Name and Marks: ${name.student_name}, ${name.marks}")
```

```
class Student
val name: Student = Student@5ebd9a88


Student Name and Marks: Vincent De Leon, 95

// mutated name.student_name
// mutated name.marks

Modified Student Name and Marks: Carolina Licona, 99
```

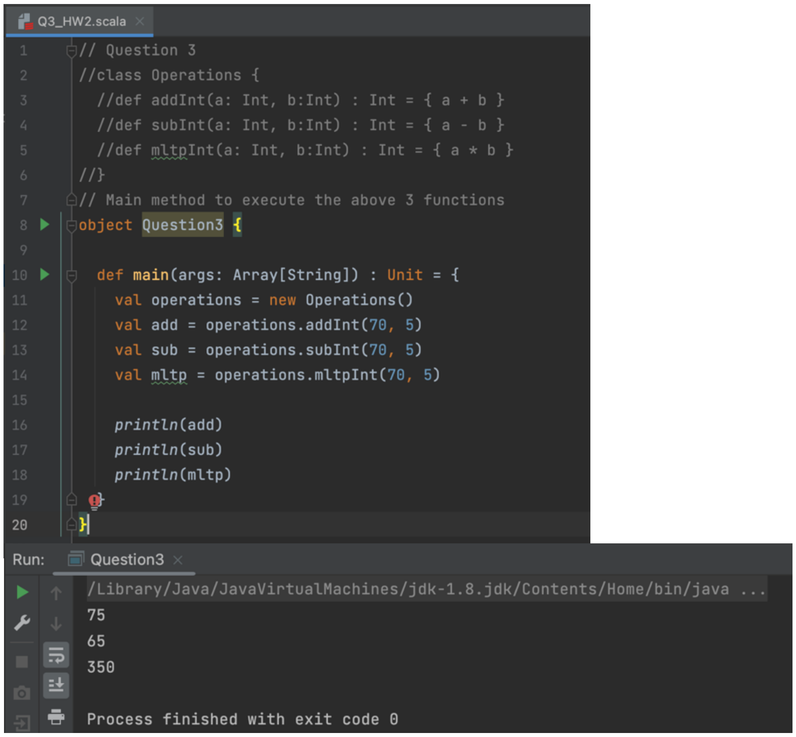
3. Question 3

   Resources (Please check Scala Worksheet Resources for more detail):
   - https://www.educative.io/answers/how-to-create-a-class-in-scala
   - https://www.tutorialspoint.com/scala/scala_functions.htm
   - https://www.geeksforgeeks.org/class-and-object-in-scala/

   Observations:
   - Object Main runs in Scala Class (using IntelliJ) not in Scala Worksheet. I had to change name from Object Main to Object Question3 to run the object and get the answers. This might be related to the duplicates in both the worksheet and class.



```scala

// Question 3
class Operations {
  def addInt(a: Int, b:Int) : Int = { a + b }
  def subInt(a: Int, b:Int) : Int = { a - b }
  def mltpInt(a: Int, b:Int) : Int = { a * b }
}
// Main method to execute the above 3 functions
// This Main runs in scala class not worksheet
object Main {

  def main(args: Array[String]) : Unit = {
    val operations = new Operations()
    val add = operations.addInt(70, 5)
    val sub = operations.subInt(70, 5)
    val mltp = operations.mltpInt(70, 5)

    println(add)
    println(sub)
    println(mltp)
  }
}
```



```scala
// Question 3
//class Operations {
  //def addInt(a: Int, b:Int) : Int = { a + b }
  //def subInt(a: Int, b:Int) : Int = { a - b }
  //def mltpInt(a: Int, b:Int) : Int = { a * b }
//}
// Main method to execute the above 3 functions
object Question3 {

  def main(args: Array[String]) : Unit = {
    val operations = new Operations()
    val add = operations.addInt(70, 5)
    val sub = operations.subInt(70, 5)
    val mltp = operations.mltpInt(70, 5)

    println(add)
    println(sub)
    println(mltp)
  }
}
```

```
/Library/Java/JavaVirtualMachines/jdk-1.8.jdk/Contents/Home/bin/java ...
75
65
350

Process finished with exit code 0
```

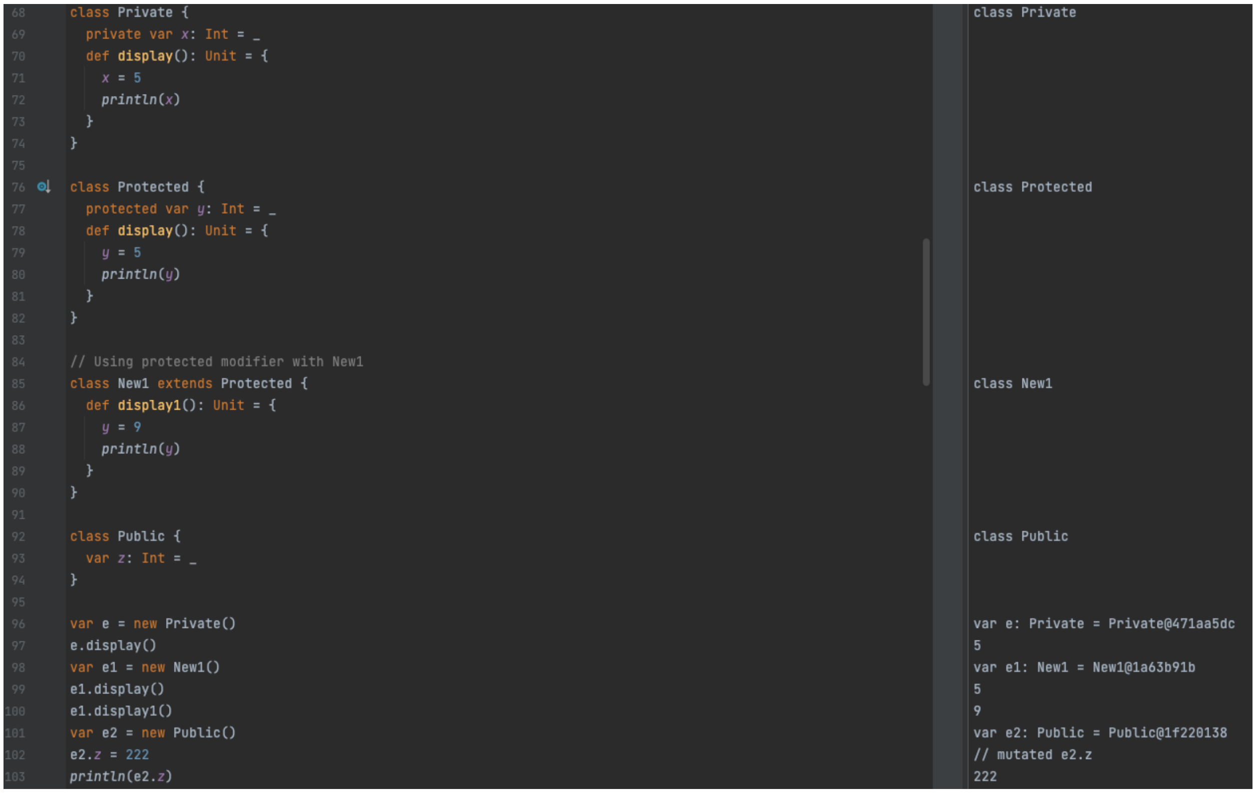
4. Question 4

   Resources:

   - Base code from: https://www.geeksforgeeks.org/access-modifiers-in-scala/

   Observations:

   - The following code were taking from the above resource in order to implement what was asked.
   - Class Private -> has private var x.
   - Class Protected -> protected var y. This is similar to private; it can only be accessible from sub classes of the base class in which the member has been defined. Var y can be accessed and modified in "New1" class, which extends Protected.
   - Class Public -> can be easily access anywhere.
   - We use "e2" to access Pubic var z without any constraint/problem.



```scala
class Private {
  private var x: Int = _
  def display(): Unit = {
    x = 5
    println(x)
  }
}

class Protected {
  protected var y: Int = _
  def display(): Unit = {
    y = 5
    println(y)
  }
}

// Using protected modifier with New1
class New1 extends Protected {
  def display1(): Unit = {
    y = 9
    println(y)
  }
}

class Public {
  var z: Int = _
}

var e = new Private()
e.display()
var e1 = new New1()
e1.display()
e1.display1()
var e2 = new Public()
e2.z = 222
println(e2.z)
```

```
class Private







class Protected








class New1






class Public



var e: Private = Private@471aa5dc
5
var e1: New1 = New1@1a63b91b
5
9
var e2: Public = Public@1f220138
// mutated e2.z
222
```

5. Question 5

Case classes:

- defined in a single statement with parameters.
- Support pattern matching.
- Can’t be extended by other classes.
- Doesn’t required “new” keyword to create objects of case classes.
- Objects of case class compare the structure of objects.
- Have predefined hashcode and equals methods.

Regular classes:

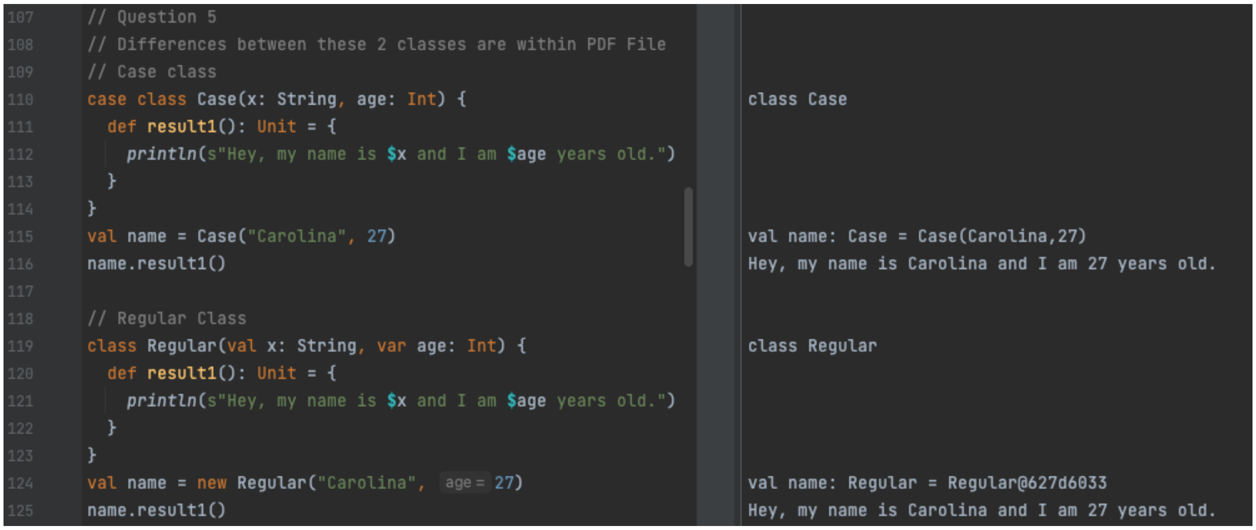
- Defined by defining methods and fields.
- Doesn’t support pattern matching.
- Can be extended by other classes.
- Keyword “new” is required for creating objects of regular classes.
- Comparison of objects of regular class is done using reference comparison.
- Have no predefined methods.

Resources:

- https://www.includehelp.com/scala/what-is-the-difference-between-scala-s-case-class-and-class.aspx#:~:text=Points%20of%20difference%20between%20Case%20Class%20and%20Class%20in%20Scala&text=The%20case%20class%20is%20defined,(syntax%20for%20defining%20class).
- https://docs.scala-lang.org/tour/case-classes.html

Observations:

- Basic code snippets for both classes were created for this question. Since Scala is a new language for me, I decided to create these code snippets so it could be easily understood. besides being initiated differently, we can see that the regular class “Regular” uses the keyword “new”. The differences above might be describing more details about these 2 classes, however I wanted to create basic examples for both.



```
// Question 5
// Differences between these 2 classes are within PDF File
// Case class
case class Case(x: String, age: Int) {
  def result1(): Unit = {
    println(s"Hey, my name is $x and I am $age years old.")
  }
}
val name = Case("Carolina", 27)
name.result1()

// Regular Class
class Regular(val x: String, var age: Int) {
  def result1(): Unit = {
    println(s"Hey, my name is $x and I am $age years old.")
  }
}
val name = new Regular("Carolina", age = 27)
name.result1()
```

```
class Case


val name: Case = Case(Carolina,27)
Hey, my name is Carolina and I am 27 years old.

class Regular


val name: Regular = Regular@627d6033
Hey, my name is Carolina and I am 27 years old.
```

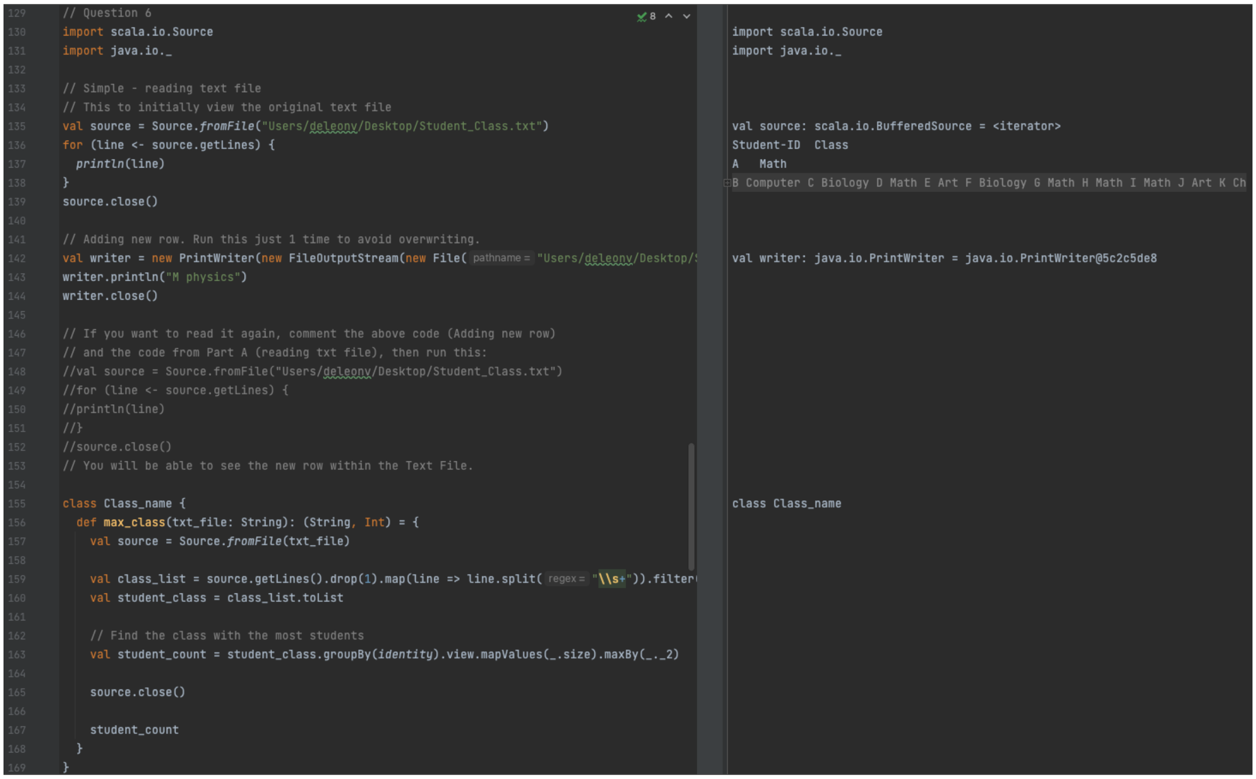
6. Question 5

   Resources (Please check Scala Worksheet Resources for more detail):

   - https://stackoverflow.com/questions/40526231/how-to-write-the-import-scala-io-source-import-java-io-libraries-in-sbt
   - https://alvinalexander.com/scala/how-to-open-read-text-files-in-scala-cookbook-examples/
   - https://www.oreilly.com/library/view/scala-cookbook/9781449340292/ch12s02.html
   - https://stackoverflow.com/questions/34613697/add-a-new-line-of-text-to-an-existing-file-in-scala
   - https://stackoverflow.com/questions/41953388/java-split-and-trim-in-one-shot
   - https://www.scala-lang.org/api/2.13.5/scala/io/Source.html
   - https://alvinalexander.com/scala/scala-split-string-example/
   - https://stackoverflow.com/questions/26943460/how-to-create-map-for-each-line-based-on-the-column-using-scala
   - https://stackoverflow.com/questions/19165977/how-does-scalas-groupby-identity-work
   - https://alvinalexander.com/scala/scala-split-string-example/

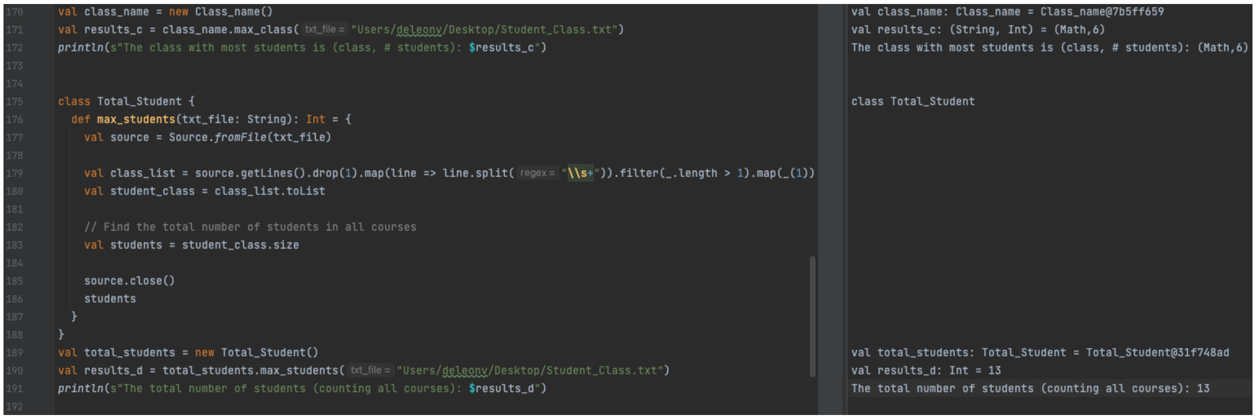
   Observations:

   - Code was creating using a mixture of Scala, O'Reilly, Stack Overflow documentation as well as internet recourses. We simply read the text file, which is located on my Desktop, and print line. It follows by adding the new row using "PrintWriter()" and 2 classes which are the "Class_name" (Math repeats itself 6 times) and "Total_Student" (get the total number of students in all courses -> 13).



```scala
// Question 6
import scala.io.Source
import java.io._

// Simple - reading text file
// This to initially view the original text file
val source = Source.fromFile("Users/deleonv/Desktop/Student_Class.txt")
for (line <- source.getLines) {
  println(line)
}
source.close()

// Adding new row. Run this just 1 time to avoid overwriting.
val writer = new PrintWriter(new FileOutputStream(new File( pathname = "Users/deleonv/Desktop/S
writer.println("M physics")
writer.close()

// If you want to read it again, comment the above code (Adding new row)
// and the code from Part A (reading txt file), then run this:
//val source = Source.fromFile("Users/deleonv/Desktop/Student_Class.txt")
//for (line <- source.getLines) {
//println(line)
//}
//source.close()
// You will be able to see the new row within the Text File.

class Class_name {
  def max_class(txt_file: String): (String, Int) = {
    val source = Source.fromFile(txt_file)

    val class_list = source.getLines().drop(1).map(line => line.split( regex = "\\s+")).filter

    val student_class = class_list.toList

    // Find the class with the most students
    val student_count = student_class.groupBy(identity).view.mapValues(_.size).maxBy(_._2)

    source.close()

    student_count
  }
}
```

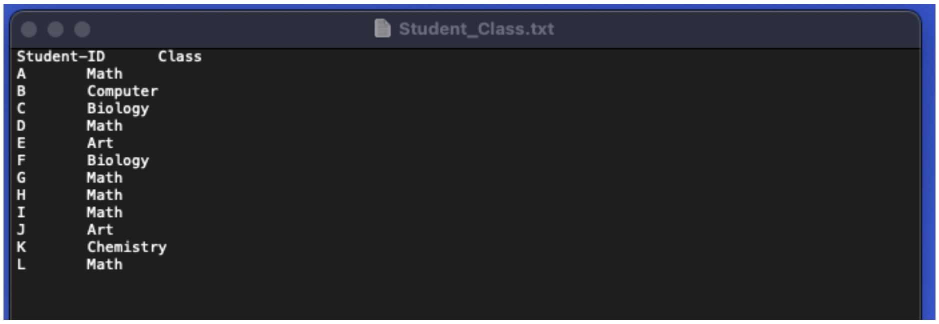
```
import scala.io.Source
import java.io._



val source: scala.io.BufferedSource = <iterator>
Student-ID  Class
A   Math
B Computer C Biology D Math E Art F Biology G Math H Math I Math J Art K Ch



val writer: java.io.PrintWriter = java.io.PrintWriter@5c2c5de8













class Class_name
```



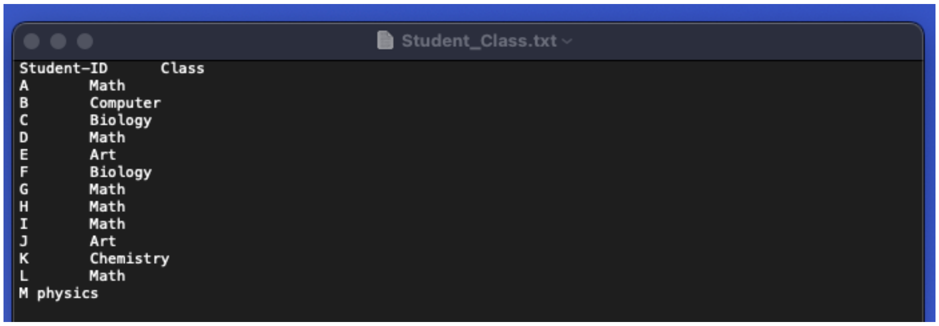
```scala
val class_name = new Class_name()
val results_c = class_name.max_class( txt_file = "Users/deleonv/Desktop/Student_Class.txt")
println(s"The class with most students is (class, # students): $results_c")


class Total_Student {
  def max_students(txt_file: String): Int = {
    val source = Source.fromFile(txt_file)

    val class_list = source.getLines().drop(1).map(line => line.split( regex = "\\s+")).filter(_.length > 1).map(_(1))
    val student_class = class_list.toList

    // Find the total number of students in all courses
    val students = student_class.size

    source.close()
    students
  }
}
val total_students = new Total_Student()
val results_d = total_students.max_students( txt_file = "Users/deleonv/Desktop/Student_Class.txt")
println(s"The total number of students (counting all courses): $results_d")
```

```
val class_name: Class_name = Class_name@7b5ff659
val results_c: (String, Int) = (Math,6)
The class with most students is (class, # students): (Math,6)


class Total_Student










val total_students: Total_Student = Total_Student@31f748ad
val results_d: Int = 13
The total number of students (counting all courses): 13
```

Text file before/after adding the new row:

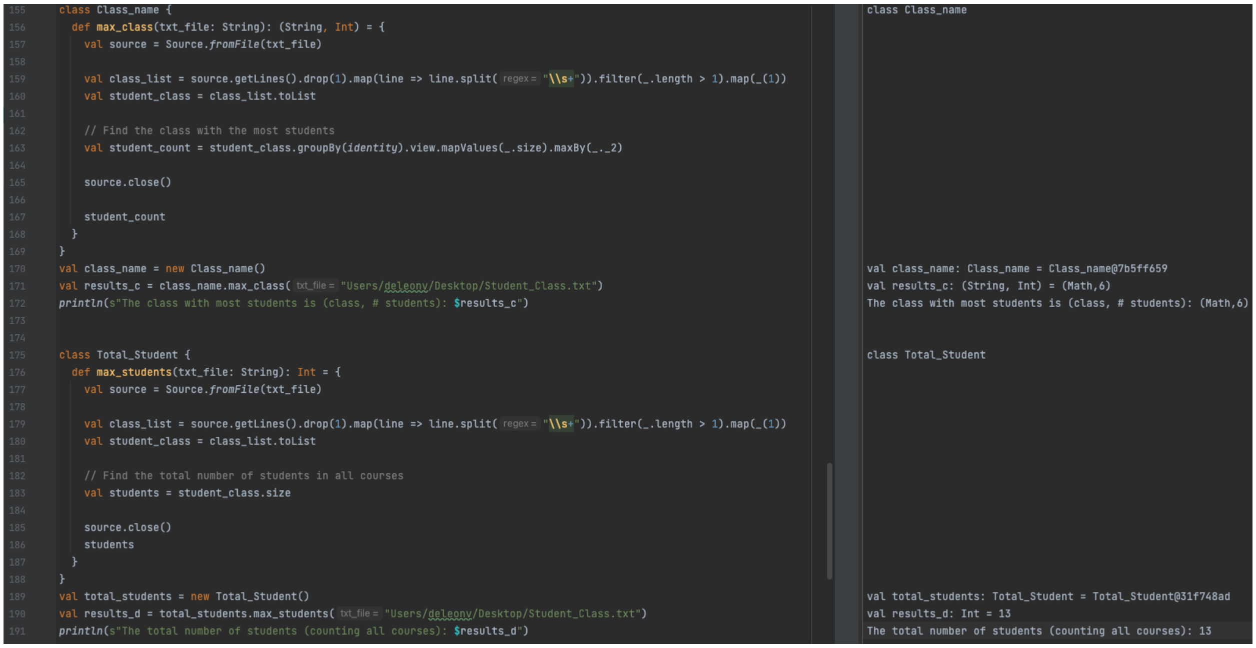


```
Student-ID      Class
A       Math
B       Computer
C       Biology
D       Math
E       Art
F       Biology
G       Math
H       Math
I       Math
J       Art
K       Chemistry
L       Math
```



```
Student-ID      Class
A       Math
B       Computer
C       Biology
D       Math
E       Art
F       Biology
G       Math
H       Math
I       Math
J       Art
K       Chemistry
L       Math
M physics
```

Closer look to both classes:



```scala
class Class_name {
  def max_class(txt_file: String): (String, Int) = {
    val source = Source.fromFile(txt_file)

    val class_list = source.getLines().drop(1).map(line => line.split("\\s+")).filter(_.length > 1).map(_(1))
    val student_class = class_list.toList

    // Find the class with the most students
    val student_count = student_class.groupBy(identity).view.mapValues(_.size).maxBy(_._2)

    source.close()

    student_count
  }
}
val class_name = new Class_name()
val results_c = class_name.max_class("Users/deleonv/Desktop/Student_Class.txt")
println(s"The class with most students is (class, # students): $results_c")


class Total_Student {
  def max_students(txt_file: String): Int = {
    val source = Source.fromFile(txt_file)

    val class_list = source.getLines().drop(1).map(line => line.split("\\s+")).filter(_.length > 1).map(_(1))
    val student_class = class_list.toList

    // Find the total number of students in all courses
    val students = student_class.size

    source.close()
    students
  }
}
val total_students = new Total_Student()
val results_d = total_students.max_students("Users/deleonv/Desktop/Student_Class.txt")
println(s"The total number of students (counting all courses): $results_d")
```

```
class Class_name

val class_name: Class_name = Class_name@7b5ff659
val results_c: (String, Int) = (Math,6)
The class with most students is (class, # students): (Math,6)

class Total_Student

val total_students: Total_Student = Total_Student@31f748ad
val results_d: Int = 13
The total number of students (counting all courses): 13
```

- .map(line => line.split(\\s+)) was used to transform each line. Splitting line into array of strings with \\s+ (split string based on a regular expression. Splitting strings on whitespace characters).

Code:

```
// Question 1
class Items {
  private var x: String = _ // _ default value defined as null, 0, False

  def get_item_from_user(input: String): Unit = {
    this.x = input // "this" key is used to fetch the variable of the class
  }

  def print_String(): Unit = {
    println(this.x.toUpperCase) // toUpperCase method
  }
}
val x = new Items() // referring to class Items
x.get_item_from_user("I had to read a lot for this Homework!")
x.print_String()

/ Question 2
class Student(var student_name: String, var marks: Int)
val name = new Student("Vincent De Leon", 95) //referring to class Student

// String interpolation
println(s"Student Name and Marks: ${name.student_name}, ${name.marks}")

name.student_name = "Carolina Licona"
name.marks = 99
// String Interpolation
println(s"Modified Student Name and Marks: ${name.student_name}, ${name.marks}")
```

```
// Question 3
class Operations {
  def addInt(a: Int, b:Int) : Int = { a + b }
  def subInt(a: Int, b:Int) : Int = { a - b }
  def mltpInt(a: Int, b:Int) : Int = { a * b }
}
// Main method to execute the above 3 functions
// This Main runs in scala class not worksheet
object Main {

  def main(args: Array[String]) : Unit = {
    val operations = new Operations()
    val add = operations.addInt(70, 5)
    val sub = operations.subInt(70, 5)
    val mltp = operations.mltpInt(70, 5)

    println(add)
    println(sub)
    println(mltp)
  }
}
```

```
// Question 4
class Private {
  private var x: Int = _
  def display(): Unit = {
    x = 5
    println(x)
  }
}

class Protected {
  protected var y: Int = _
  def display(): Unit = {
    y = 5
    println(y)
  }
}

// Using protected modifier with New1
class New1 extends Protected {
  def display1(): Unit = {
    y = 9
    println(y)
  }
}
```

```
class Public {
  var z: Int = _
}

var e = new Private()
e.display()
var e1 = new New1()
e1.display()
e1.display1()
var e2 = new Public()
e2.z = 222
println(e2.z)
```

```
// Question 5

// Differences between these 2 classes are within PDF File

// Case class

case class Case(x: String, age: Int) {

  def result1(): Unit = {

    println(s"Hey, my name is $x and I am $age years old.")

  }

}

val name = Case("Carolina", 27)

name.result1()


// Regular Class

class Regular(val x: String, var age: Int) {

  def result1(): Unit = {

    println(s"Hey, my name is $x and I am $age years old.")

  }

}

val name = new Regular("Carolina", 27)

name.result1()
```

```
// Question 6

import scala.io.Source

import java.io._


val source = Source.fromFile("Users/deleonv/Desktop/Student_Class.txt")

for (line <- source.getLines) {

  println(line)

}

source.close()


// Adding new row. Run this just 1 time to avoid overwriting.

val writer = new PrintWriter(new FileOutputStream(new File("Users/deleonv/Desktop/Student_Class.txt"), true))

writer.println("M physics")

writer.close()


// If you want to read it again, comment the above code (Adding new row)

// and the code from Part A (reading txt file), then run this:

//val source = Source.fromFile("Users/deleonv/Desktop/Student_Class.txt")

//for (line <- source.getLines) {

//println(line)

//}

//source.close()

// You will be able to see the new row within the Text File.
```

```
class Class_name {
  def max_class(txt_file: String): (String, Int) = {
    val source = Source.fromFile(txt_file)

    val class_list = source.getLines().drop(1).map(line => line.split("\\s+")).filter(_.length > 1).map(_(1))
    val student_class = class_list.toList

    // Find the class with the most students
    val student_count = student_class.groupBy(identity).view.mapValues(_.size).maxBy(_._2)

    source.close()

    student_count
  }
}
val class_name = new Class_name()
val results_c = class_name.max_class("Users/deleonv/Desktop/Student_Class.txt")
println(s"The class with most students is (class, # students): $results_c")
```

```
class Total_Student {
  def max_students(txt_file: String): Int = {
    val source = Source.fromFile(txt_file)

    val class_list = source.getLines().drop(1).map(line => line.split("\\s+")).filter(_.length > 1).map(_(1))
    val student_class = class_list.toList

    // Find the total number of students in all courses
    val students = student_class.size

    source.close()
    students
  }
}
val total_students = new Total_Student()
val results_d = total_students.max_students("Users/deleonv/Desktop/Student_Class.txt")
println(s"The total number of students (counting all courses): $results_d")
```