Basics of Scala

Vicente De Leon

UID: 2001014594

Final Project

1. Question 1

   Resources:

   - Equals method: https://www.geeksforgeeks.org/scala-string-equals-method-with-example/
   - hashCode method: https://www.geeksforgeeks.org/scala-string-hashcode-method-with-example/
   - toString method: https://www.geeksforgeeks.org/scala-int-tostring-method-with-example/
   - copy method: https://www.educative.io/answers/what-is-copy-in-scala

   Observations:

   - Synthetic Methods using equals(), hashCode(), toString, and copy(). All these methods are related to the class “Person”. The idea for this code comes from the copy method source.
   - We proceed to create person 1 and person, which are instances, in order to compare them using the equals method.
   - We end up having False as an answer as well as different hashcodes for each “person” as seen in the image below. We finally apply the toString method to get the string representation of the object.
   - The copy method was used to create a new instance of the case class with modified fields. “Person2Modified” is the copy of person 2 and includes new values like age 30 and salary 6000.

```scala
// Question 1 (Synthetic Methods)
// Equals() method: https://www.geeksforgeeks.org/scala-string-equals-method-with-example/
// hashCode() method: https://www.geeksforgeeks.org/scala-string-hashcode-method-with-example/
// toString() method: https://www.geeksforgeeks.org/scala-int-tostring-method-with-example/
// copy() method: https://www.educative.io/answers/what-is-copy-in-scala

object FPQuestion1 {
  private case class Person(name: String, age: Int, salary: Int)

  def main(args: Array[String]): Unit = {
    val person1 = Person("Carolina", 28, 5000) // creating instance person 1
    val person2 = Person("Vicente", 27, 5000) // creating instance person 2

    // Using the Equals() method
    val EqualsMethod = person1.equals(person2) // lets compare them
    println(s"Are person1 and person2 equal? $EqualsMethod")

    // hashCode() method
    val HashCode1 = person1.hashCode() // integer
    val HashCode2 = person2.hashCode() // integer
    println(s"hashCode for person1 is: $HashCode1")
    println(s"hashCode for person2 is: $HashCode2")

    // toString method
    val toString1 = person1.toString // no need to use () -> toString() string representation
    val toString2 = person2.toString // no need to use () -> toString() string representation
    println(s"The string representation of person1 is: $toString1")
    println(s"The string representation of person2 is: $toString2")

    // Copy() method
    val person2Modified = person2.copy(age = 30, salary = 6000) // new instance along with new values
    println(s"Applying the copy() method to person2: $person2Modified") // should be Vicente, 30, 6000
  }
}
```

```
Are person1 and person2 equal? false
hashCode for person1 is: -502720932
hashCode for person2 is: -1394834999
The string representation of person1 is: Person(Carolina,28,5000)
The string representation of person2 is: Person(Vicente,27,5000)
Applying the copy() method to person2: Person(Vicente,30,6000)
```

2. Question 2

   Resources:

   - Checking objects: https://stackoverflow.com/questions/44454287/scala-how-to-check-whether-a-object-is-instance-of-array
   - Checking objects: https://stackoverflow.com/questions/11290169/how-does-isinstanceof-work
   - Stack to list: https://www.geeksforgeeks.org/scala-stack-tolist-method-with-example/
   - Arrays: https://www.geeksforgeeks.org/scala-arrays/
   - Vectors: https://www.geeksforgeeks.org/scala-vector/
   - toList: https://www.geeksforgeeks.org/scala-map-tolist-method-with-example/
   - toVector: https://www.geeksforgeeks.org/program-to-convert-java-list-to-a-vector-in-scala/
   - mkString: https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/

   Observations:

   - Basic operations on Arrays, Lists, and Vector Methods.
   - This code is very basic, we start by creating an array and proceed to create a list and eventually a vector from that array using the assign method.
   - I decided to include the "isInstanceOf[Array[_]]" to check whether the array is in fact and array. We do the same procedure for both the list and vector.
   - Array, the chosen operation for this was to calculate the sum of all its element using a for loop.
   - List, the operation was to find the maximum element within the list using the max method.
   - Vector, we finalize the code by creating a new vector by merging two vectors together as shown in the above results.
   - Operations and ideas came from Geek for Geeks sources and StackOverFlow sources above. You could also see the sources right above each code for this part.

Please continue below for results:

```scala
// Question 2 (Arrays, Lists, Vectors)
// checking objects: https://stackoverflow.com/questions/44454287/scala-how-to-check-whether-a-object-is-instance-of-array
// checking objects: https://stackoverflow.com/questions/11298169/how-does-isinstanceof-work
// stack to list: https://www.geeksforgeeks.org/scala-stack-tolist-method-with-example/
// arrays: https://www.geeksforgeeks.org/scala-arrays/
// vectors: https://www.geeksforgeeks.org/scala-vector/
// toList: https://www.geeksforgeeks.org/scala-map-tolist-method-with-example/
// toVector: https://www.geeksforgeeks.org/program-to-convert-java-list-to-a-vector-in-scala/
// mkString: https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/

object FPQuestion2 {
  def main(args: Array[String]): Unit = {
    val array1 = Array(3.0, 5.0, 5.5, 7.5, 9.0) // Creating array with elements
    if (array1.isInstanceOf[Array[_]]) // checking if it's an array
      println("array1 is an Array")
    println("The array elements are: ")
    for (elem <- array1) {
      println(elem)
    }
    println("\n")

    val list1 = array1.toList // Converting array to list using the toList method
    if (list1.isInstanceOf[List[_]]) // checking if it's a list
      println("list1 is a List")
    print("Check the list: ")
    println(list1)
    println("\n")

    val vector1 = list1.toVector // Converting list to vector using the toVector method
    if (vector1.isInstanceOf[Vector[_]]) // checking if it's a vector
      println("vector1 is a Vector")
    print("Check the vector: ")
    println(vector1)
    println("\n")

    // Basic operations -> Array (calculate the sum of all array elements)
    // total: https://www.tutorialspoint.com/scala/scala_arrays.htm
    var total = 0.0
    for (elem <- array1) {
      total += elem
    }
    println("The total number of elements in the Array is: " + total)
```

```scala
    // Basic operations -> Lists (maximum element of list)
    // max: https://www.geeksforgeeks.org/scala-list-max-method-with-example/
    val MaxList = list1.max
    println("The largest of all the elements in the list is: " + MaxList)

    // Basic operations -> Vector (creating a new vector by merging two vectors together)
    // vector: https://www.geeksforgeeks.org/scala-vector/
    val vector2 = Vector(50.0, 100.0)
    val NewVector = vector1 ++ vector2
    println("Merging Vector 1 and Vector 2:")
    println(NewVector.mkString(" "))
  }
}
```

```
array1 is an Array
The array elements are:
3.0
5.0
5.5
7.5
9.0


list1 is a List
Check the list: List(3.0, 5.0, 5.5, 7.5, 9.0)


vector1 is a Vector
Check the vector: Vector(3.0, 5.0, 5.5, 7.5, 9.0)


The total number of elements in the Array is: 30.0
The largest of all the elements in the list is: 9.0
Merging Vector 1 and Vector 2:
3.0 5.0 5.5 7.5 9.0 50.0 100.0
```

3. Question 3
    - Modifiers: https://www.geeksforgeeks.org/access-modifiers-in-scala/
    - New keyword: https://stackoverflow.com/questions/9727637/new-keyword-in

    Observations:
    - This code looks simply, but it was kind of tricky to put together. It shows the how the public, private, and protected modifiers interact with each other.
    - The idea behind these codes came from Geeks for Geeks and it explains the use of each modifier. The public modifier can be access anywhere. The private can only be used inside the defining class or through one of its objects. The protected modifier can be access from sub classes of the base class in which the member has been defined.
    - "class abc" contains public variable x.
    - "class xyz" contains private variable y variable (access only via xyz) and protected variable x (access via xyz and subclass new1).
    - Within "class xyz" the display() method has y = 5 and modifier() updates "e.x" (from class abc) to match "class xyz's" x value.
    - "class new1 extends xyz" contains method display2(). This class gets the values and methods from the "class xyz". Display2() has its own x value of 10 and sets x value from the "class abc" to its own x value -> "e.x". This code prints its own x value of 10 and it also updates the value of "e.x" which it's also 10.
    - Main method calls all methods to print results using the new instances from each class.

Please continue below for results:

```scala
class abc { // public modifier
  var x: Int = 222 // variable x -> public member
}

class xyz {
  private var y: Int = 579 // variable y -> private member (accessed via class xyz)
  protected var x: Int = 999 // variable x -> protected member (accessed via class xyz and subclass new1)

  // class xyz contains functions/methods display() and modifier()
  def display(): Unit = {
    y = 5
    println(y)
  }

  def modifier(e: abc): Unit = {
    e.x = x
    println(e.x)
  }
}

// Class that extends to xyz
class new1 extends xyz {
  // contains function/method display2()
  def display2(e: abc): Unit = {
    x = 10
    println(x)
    e.x = x
    println(e.x)
  }
}

object FPQuestion3 {
  def main(args: Array[String]): Unit = {
    val NewAbc = new abc()
    val NewXyz = new xyz()
    val New1 = new new1()
    NewXyz.display()
    NewXyz.modifier(NewAbc)
    New1.display2(NewAbc)
  }
}
```

```
5
999
10
10
```

4. Question 4

   Resources:

   - While Loop: https://www.geeksforgeeks.org/while-and-do-while-loop-in-scala/
   - For Loop: https://www.geeksforgeeks.org/for-loop-in-scala/
   - For Loop: https://stackoverflow.com/questions/6833501/efficient-iteration-with-index-in-scala
   - Foreach: https://stackoverflow.com/questions/45165065/foreach-loop-in-scala
   - Foreach: https://www.baeldung.com/scala/foreach-collections
   - Java foreach: https://www.geeksforgeeks.org/how-to-change-values-in-an-array-when-doing-foreach-loop-in-javascript/

   Observations:

   - Code shows interaction between while loop, for loop, and foreach loop.
   - While loop method continues as long as element is less than the length of the array. It adds +3 to the elements in position: 0, 3, 6, 9 etc. finally it adds +3 after the iterations.
   - For loop method: use the element index and adds +5 for each element within the array.
   - Foreach loop: this code came from the JAVA foreach loop updating elements source. Just like the for loop, it uses element indices and adds +7 to each element within the array.

```scala
// Question 4 (For Loop, While Loop, Foreach Loop Methods)
// While Loop: https://www.geeksforgeeks.org/while-and-do-while-loop-in-scala/
// For Loop: https://www.geeksforgeeks.org/for-loop-in-scala/
// For Loop: https://stackoverflow.com/questions/6833501/efficient-iteration-with-index-in-scala
// Foreach: https://stackoverflow.com/questions/45165065/foreach-loop-in-scala
// Foreach: https://www.baeldung.com/scala/foreach-collections
// Java Foreach Loop updating elements: https://www.geeksforgeeks.org/how-to-change-values-in-an-array-when-doing-foreach-loop-in-javascript/
object FPQuestion4 {
  private val numbers = Array(5, 7, 9, 11, 13, 15) // lets create an Array

  private def WhileLoop(): Unit = { // While Loop method
    var elem = 0
    while (elem < numbers.length) { // loop continues as long as element is less than length of array
      numbers(elem) = numbers(elem) + 3 // adding + 3 to elements in position 0, 3, 6, 9 etc
      elem += 3 // adding + 3 after iteration
    }
  }

  private def ForLoop(): Unit = { // For Loop method
    for (elem <- numbers.indices) {
      numbers(elem) = numbers(elem) + 5 // adding +5 of each element within the array
    }
  }

  // Foreach Loop: https://www.baeldung.com/scala/foreach-collections
  // Java Foreach Loop updating elements: https://www.geeksforgeeks.org/how-to-change-values-in-an-array-when-doing-foreach-loop-in-javascript/
  private def ForeachLoop(): Unit = { // JAVA - foreach loop
    numbers.indices.foreach(elem => numbers(elem) = numbers(elem) + 7) // add +7 to each element within the array
  }

  def main(args: Array[String]): Unit = {
    WhileLoop()
    println(s"The While Loop results are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
    ForLoop()
    println(s"The For Loop results are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
    ForeachLoop()
    println(s"The Foreach Loop results are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
  }
}
```

```
The While Loop results are: 8, 7, 9, 14, 13, 15
The For Loop results are: 13, 12, 14, 19, 18, 20
The Foreach Loop results are: 20, 19, 21, 26, 25, 27
```

5. Question 5

   Resources:

   - Filter method: https://alvinalexander.com/scala/how-to-use-filter-method-scala-collections-cookbook/
   - Yield method: https://www.geeksforgeeks.org/scala-yield-keyword/

   Observations:

   - This last code shows the basic usage of filter and yield methods. We initialize the code by creating an array named “numbers” and we proceed to filter this array in order to create the new “FilterMethod” array. This new array contains numbers that are greater than 9. “YieldKeyword” iterates over the elements from the above array and use the if statement to keep elements that are greater than 13.

```scala
// Question 5 (Filtering and Yield Methods)
// Filter() Method: https://alvinalexander.com/scala/how-to-use-filter-method-scala-collections-cookbook/
// Yield Keyword: https://www.geeksforgeeks.org/scala-yield-keyword/
object FPQuestion5 {
  def main(args: Array[String]): Unit = {
    val numbers = Array(5, 7, 9, 11, 13, 15, 17, 20, 22) // Creating Array
    val FilterMethod = numbers.filter(_ > 9) // filters elements greater than 9
    // YieldKeyword: iterates over the elements from the FilterMethod array and uses if statement to keep elements greater than 13
    val YieldKeyword = for (elem <- FilterMethod if elem > 13) yield elem // yield method

    println(s"The elements inside the array are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
    println(s"Numbers that are greater than 9: ${FilterMethod.mkString(", ")}") // format suggestion IntelliJ
    println(s"Getting numbers greater than 13: ${YieldKeyword.mkString(", ")}") // format suggestion IntelliJ
  }
}
```

```
The elements inside the array are: 5, 7, 9, 11, 13, 15, 17, 20, 22
Numbers that are greater than 9: 11, 13, 15, 17, 20, 22
Getting numbers greater than 13: 15, 17, 20, 22
```

Codes:

```
// Question 1 (Synthetic Methods)
object FPQuestion1 {
  private case class Person(name: String, age: Int, salary: Int)
  def main(args: Array[String]): Unit = {
    val person1 = Person("Carolina", 28, 5000) // creating instance person 1
    val person2 = Person("Vicente", 27, 5000) // creating instance person 2

    // Using the Equals() method
    val EqualsMethod = person1.equals(person2) // lets compare them
    println(s"Are person1 and person2 equal? $EqualsMethod")

    // hashCode() method
    val HashCode1 = person1.hashCode() // integer
    val HashCode2 = person2.hashCode() // integer
    println(s"hashCode for person1 is: $HashCode1")
    println(s"hashCode for person2 is: $HashCode2")

    // toString method
    val toString1 = person1.toString // no need to use () -> toString() string representation
    val toString2 = person2.toString // no need to use () -> toString() string representation
    println(s"The string representation of person1 is: $toString1")
    println(s"The string representation of person2 is: $toString2")

    // Copy() method
    val person2Modified = person2.copy(age = 30, salary = 6000) // new instance along with new values
    println(s"Applying the copy() method to person2: $person2Modified") // should be Vicente, 30, 6000
  }
}
```

```
// Question 2
object FPQuestion2 {
  def main(args: Array[String]): Unit = {
    val array1 = Array(3.0, 5.0, 5.5, 7.5, 9.0) // Creating array with elements
    if (array1.isInstanceOf[Array[_]]) // checking if it's an array
      println("array1 is an Array")
    println("The array elements are: ")
    for (elem <- array1) {
      println(elem)
    }
    println("\n")

    val list1 = array1.toList // Converting array to list using the toList method
    if (list1.isInstanceOf[List[_]]) // checking if it's a list
      println("list1 is a List")
    print("Check the list: ")
    println(list1)
    println("\n")

    val vector1 = list1.toVector // Converting list to vector using the toVector method
    if (vector1.isInstanceOf[Vector[_]]) // checking if it's a vector
      println("vector1 is a Vector")
    print("Check the vector: ")
    println(vector1)
    println("\n")
```

```
  // Basic operations -> Array (calculate the sum of all array elements)
  // total: https://www.tutorialspoint.com/scala/scala_arrays.htm
  var total = 0.0
  for (elem <- array1) {
    total += elem
  }
  println("The total number of elements in the Array is: " + total)

  // Basic operations -> Lists (maximum element of list)
  // max: https://www.geeksforgeeks.org/scala-list-max-method-with-example/
  val MaxList = list1.max
  println("The largest of all the elements in the list is: " + MaxList)

  // Basic operations -> Vector (creating a new vector by merging two vectors together)
  // vector: https://www.geeksforgeeks.org/scala-vector/
  val vector2 = Vector(50.0, 100.0)
  val NewVector = vector1 ++ vector2
  println("Merging Vector 1 and Vector 2:")
  println(NewVector.mkString(" "))
 }
}
```

```
// Question 3

class abc { // public modifier
  var x: Int = 222 // variable x -> public member
}

class xyz {
  private var y: Int = 579 // variable y -> private member (accessed via class xyz)
  protected var x: Int = 999 // variable x -> protected member (accessed via class xyz and subclass new1)

  // class xyz contains functions/methods display() and modifier()
  def display(): Unit = {
    y = 5
    println(y)
  }

  def modifier(e: abc): Unit = {
    e.x = x
    println(e.x)
  }
}
```

```
// Class that extends to xyz
class new1 extends xyz {
  // contains function/method display2()
  def display2(e: abc): Unit = {
    x = 10
    println(x)
    e.x = x
    println(e.x)
  }
}

object FPQuestion3 {
  def main(args: Array[String]): Unit = {
    val NewAbc = new abc()
    val NewXyz = new xyz()
    val New1 = new new1()
    NewXyz.display()
    NewXyz.modifier(NewAbc)
    New1.display2(NewAbc)
  }
}
```

```
// Question 4


object FPQuestion4 {
  private val numbers = Array(5, 7, 9, 11, 13, 15) // lets create an Array


  private def WhileLoop(): Unit = { // While Loop method
    var elem = 0
    while (elem < numbers.length) { // loop continues as long as element is less than length of array
      numbers(elem) = numbers(elem) + 3 // adding + 3 to elements in position 0, 3, 6, 9 etc
      elem += 3 // adding + 3 after iteration
    }
  }


  private def ForLoop(): Unit = { // For Loop method
    for (elem <- numbers.indices) {
      numbers(elem) = numbers(elem) + 5 // adding +5 of each element within the array
    }
  }


  // Foreach Loop: https://www.baeldung.com/scala/foreach-collections
  // Java Foreach Loop updating elements: https://www.geeksforgeeks.org/how-to-change-values-in-an-array-when-doing-foreach-loop-in-javascript/
  private def ForeachLoop(): Unit = { // JAVA - foreach loop
    numbers.indices.foreach(elem => numbers(elem) = numbers(elem) + 7) // add +7 to each element within the array
  }
```

```
  def main(args: Array[String]): Unit = {
    WhileLoop()
    println(s"The While Loop results are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
    ForLoop()
    println(s"The For Loop results are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
    ForeachLoop()
    println(s"The Foreach Loop results are: ${numbers.mkString(", ")}") // format suggestion IntelliJ
  }
}
```

```
// Question 5

object FPQuestion5 {

  def main(args: Array[String]): Unit = {

    val numbers = Array(5, 7, 9, 11, 13, 15, 17, 20, 22) // Creating Array

    val FilterMethod = numbers.filter(_ > 9) // filters elements greater than 9

    // YieldKeyword: iterates over the elements from the FilterMethod array and uses if statement to keep elements greater than 13

    val YieldKeyword = for (elem <- FilterMethod if elem > 13) yield elem // yield method


    println(s"The elements inside the array are: ${numbers.mkString(", ")}") // format suggestion IntelliJ

    println(s"Numbers that are greater than 9: ${FilterMethod.mkString(", ")}") // format suggestion IntelliJ

    println(s"Getting numbers greater than 13: ${YieldKeyword.mkString(", ")}") // format suggestion IntelliJ

  }

}
```