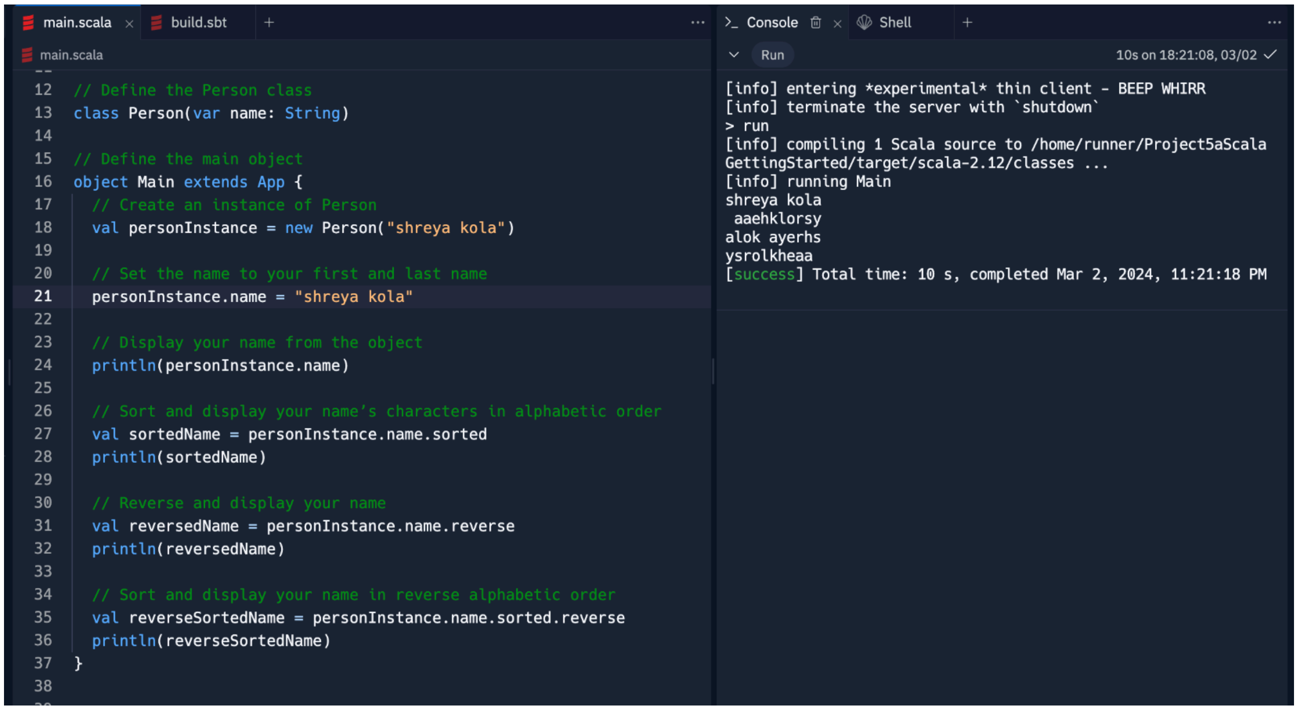
Project_5a: Scala Getting Started

Step 1 (10 points) – Install Scala to run a local interactive REPL (or run a Scala REPL online, such as

- Create a class called Person with a string attribute of name
- Create an instance of Person using your first name as the instance name
- Set that instance's name to your First and last name
- Display your name from the object
- Sort and display your name's characters in alphabetic order o Reverse and display your name Sort and display your name in reverse alphabetical order to Include comment lines in your REPL for citations of helpful or copied sources (including iolanguage.org)
- Capture all the REPL output into a Step1.txt file (or equivalent)



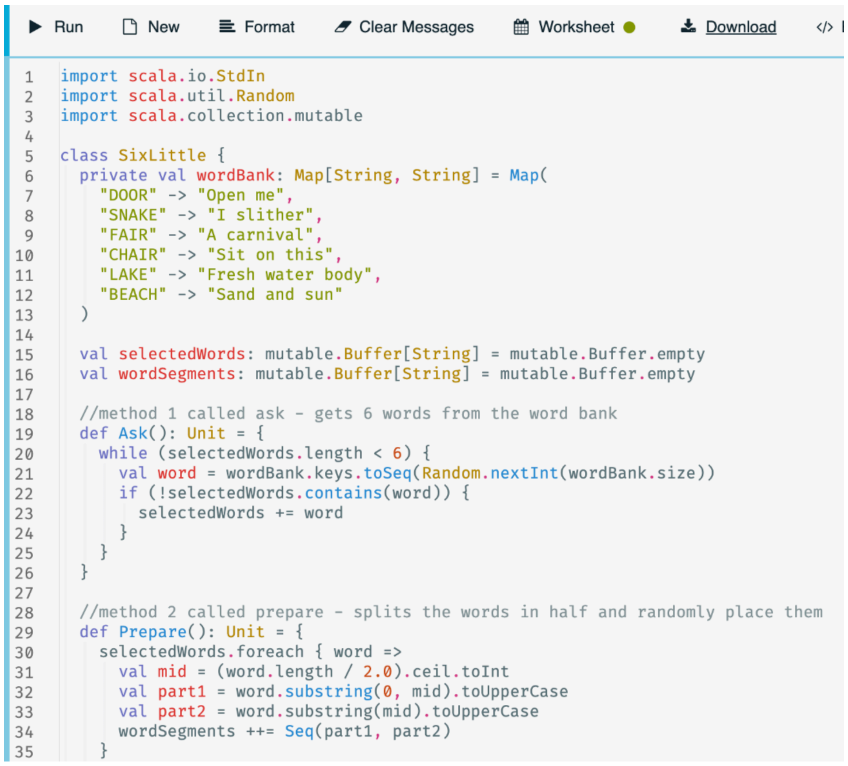
```scala
// Define the Person class
class Person(var name: String)

// Define the main object
object Main extends App {
  // Create an instance of Person
  val personInstance = new Person("shreya kola")

  // Set the name to your first and last name
  personInstance.name = "shreya kola"

  // Display your name from the object
  println(personInstance.name)

  // Sort and display your name's characters in alphabetic order
  val sortedName = personInstance.name.sorted
  println(sortedName)

  // Reverse and display your name
  val reversedName = personInstance.name.reverse
  println(reversedName)

  // Sort and display your name in reverse alphabetic order
  val reverseSortedName = personInstance.name.sorted.reverse
  println(reverseSortedName)
}
```

```
[info] entering *experimental* thin client - BEEP WHIRR
[info] terminate the server with `shutdown`
> run
[info] compiling 1 Scala source to /home/runner/Project5aScala
GettingStarted/target/scala-2.12/classes ...
[info] running Main
shreya kola
 aaehklorsy
alok ayerhs
ysrolkheaa
[success] Total time: 10 s, completed Mar 2, 2024, 11:21:18 PM
```

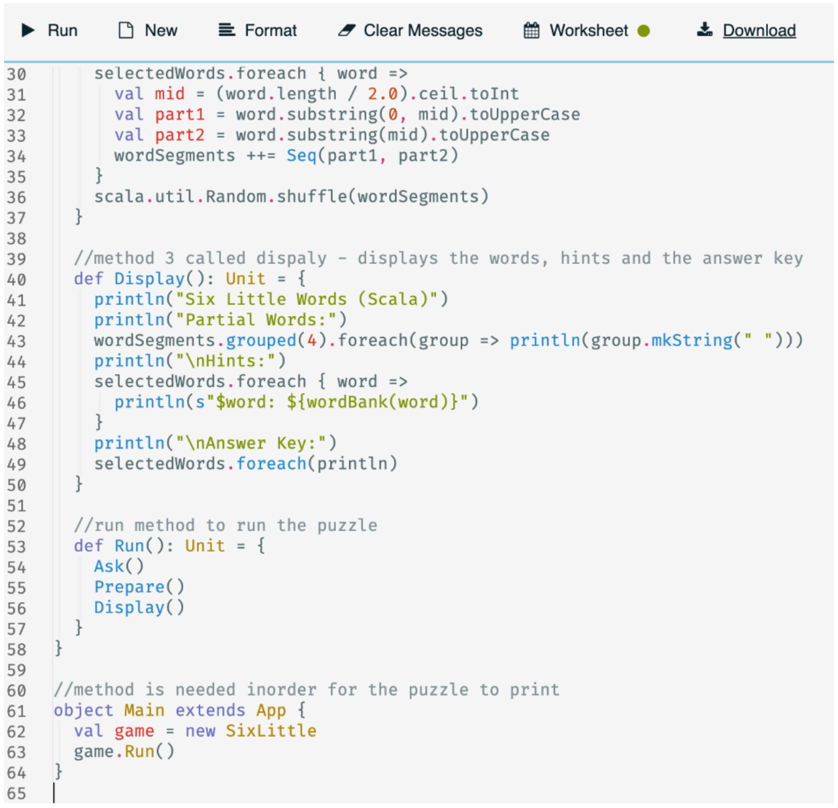
Step 2 (25 points) – Write a Scala program that prints out a Six Little Words puzzle.
Create a standalone Scala program (a .scala file) that will successfully run if called using scala. The code should instantiate an object from a class called SixLitle and call that object's Run method.
Create a SixLitle class in Scala that has the following four methods:
Ask – asks for six pairs of string inputs – a word of at least 4 characters length (should re-ask if too short) followed by a related hint string (of any length) – this pair of words should be kept in collection attributes (words, hints) inside the SixLitle class.
Prepare – The prepare method will break each word in the words collection in half (be consistent about what to do for an odd number of letters), the broken parts of the words will be capitalized and then added to a collection called tokens. Finally, the tokens collection should be randomly shuffled so that the tokens are not in the order they were originally placed.
Display – displays the title "Six Little Words (Scala)", displays a neat table titled "Tokens" of the tokens – 4 words across in 3 rows, displays the hints in a list titled "HINTS", and then display another list labeled "ANSWER KEY" and list the original words on two lines, separating each section with a blank line.
Run – should clear all collections, call Ask, Prepare, and Display in turn.

Typical output when program is run and six words/hints have been entered by the user:

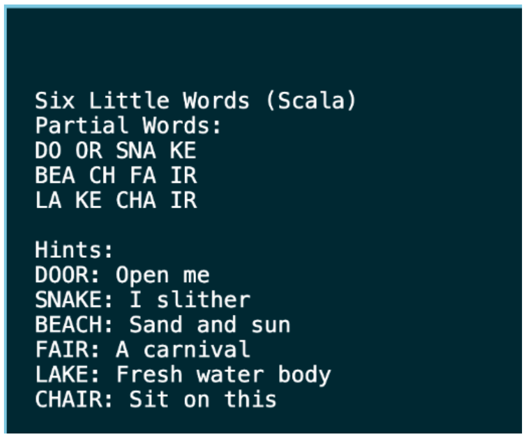
```
Six Little Words (Scala)
Partial Words:
LA ACH KE
AKE CH BE
AIR DO IR
FA SN OR

Hints:
Open me
I slither
A carnival
Sit on this
Fresh water body
Sand and sun

Answer Key:
DOOR SNAKE FAIR
CHAIR LAKE BEACH
```



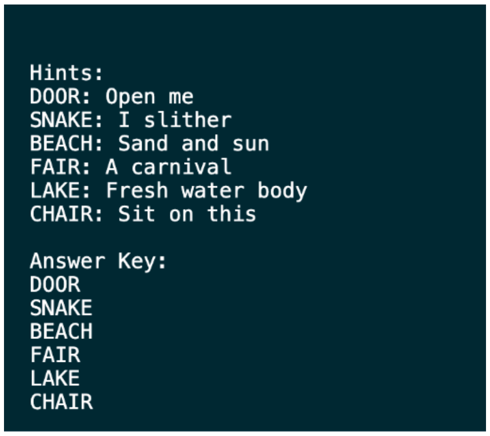
```scala
import scala.io.StdIn
import scala.util.Random
import scala.collection.mutable

class SixLittle {
  private val wordBank: Map[String, String] = Map(
    "DOOR" -> "Open me",
    "SNAKE" -> "I slither",
    "FAIR" -> "A carnival",
    "CHAIR" -> "Sit on this",
    "LAKE" -> "Fresh water body",
    "BEACH" -> "Sand and sun"
  )

  val selectedWords: mutable.Buffer[String] = mutable.Buffer.empty
  val wordSegments: mutable.Buffer[String] = mutable.Buffer.empty

  //method 1 called ask - gets 6 words from the word bank
  def Ask(): Unit = {
    while (selectedWords.length < 6) {
      val word = wordBank.keys.toSeq(Random.nextInt(wordBank.size))
      if (!selectedWords.contains(word)) {
        selectedWords += word
      }
    }
  }

  //method 2 called prepare - splits the words in half and randomly place them
  def Prepare(): Unit = {
    selectedWords.foreach { word =>
      val mid = (word.length / 2.0).ceil.toInt
      val part1 = word.substring(0, mid).toUpperCase
      val part2 = word.substring(mid).toUpperCase
      wordSegments ++= Seq(part1, part2)
    }
```



```scala
    selectedWords.foreach { word =>
      val mid = (word.length / 2.0).ceil.toInt
      val part1 = word.substring(0, mid).toUpperCase
      val part2 = word.substring(mid).toUpperCase
      wordSegments ++= Seq(part1, part2)
    }
    scala.util.Random.shuffle(wordSegments)
  }

  //method 3 called dispaly - displays the words, hints and the answer key
  def Display(): Unit = {
    println("Six Little Words (Scala)")
    println("Partial Words:")
    wordSegments.grouped(4).foreach(group => println(group.mkString(" ")))
    println("\nHints:")
    selectedWords.foreach { word =>
      println(s"$word: ${wordBank(word)}")
    }
    println("\nAnswer Key:")
    selectedWords.foreach(println)
  }

  //run method to run the puzzle
  def Run(): Unit = {
    Ask()
    Prepare()
    Display()
  }
}

//method is needed inorder for the puzzle to print
object Main extends App {
  val game = new SixLittle
  game.Run()
}
```



```
Six Little Words (Scala)
Partial Words:
DO OR SNA KE
BEA CH FA IR
LA KE CHA IR

Hints:
DOOR: Open me
SNAKE: I slither
BEACH: Sand and sun
FAIR: A carnival
LAKE: Fresh water body
CHAIR: Sit on this
```



```
Hints:
DOOR: Open me
SNAKE: I slither
BEACH: Sand and sun
FAIR: A carnival
LAKE: Fresh water body
CHAIR: Sit on this

Answer Key:
DOOR
SNAKE
BEACH
FAIR
LAKE
CHAIR
```

Comment all code appropriately, including identifying header information, internal comments as warranted, and capturing URLs for citations of helpful or copied sources.
Capture a run of the program in a text file labeled Step2Result.txt (or cut and paste an image to an image file).

- README (5 points) Include a Markdown README in GitHub for 3a that has the (1) assignment title, (2) your name, (3) any issues you had with the assignment (might be “None”).
- Turn in a GitHub link to CougarVIEW to a repo containing: Step1.txt (or image), Step2.scala, Step2Result.txt (or image), README.