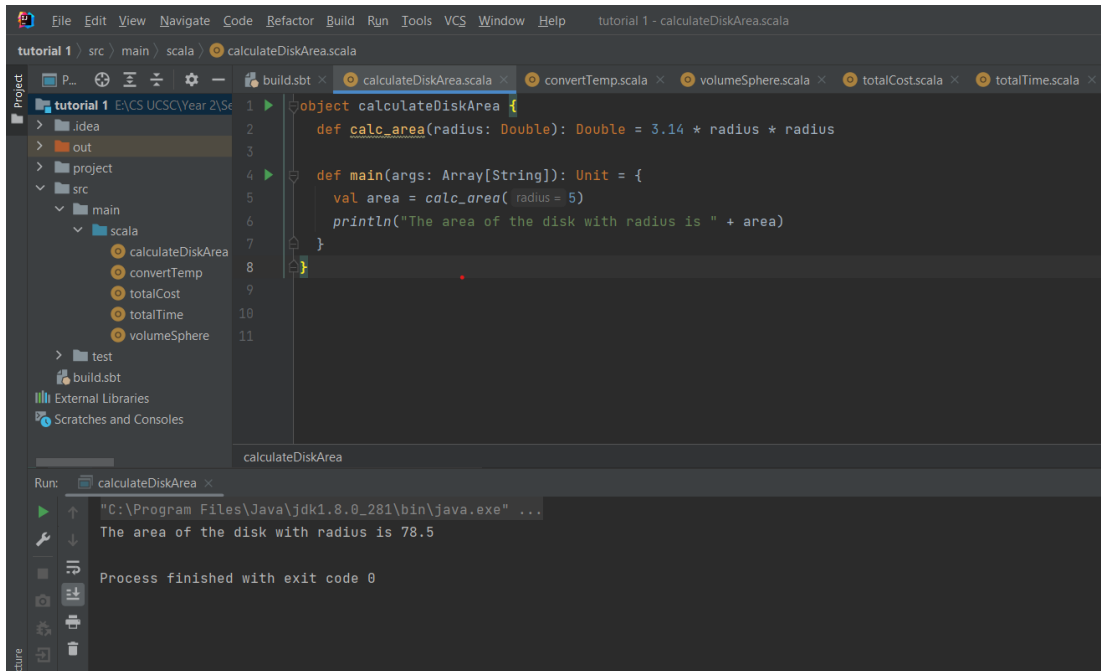


Functional Programming – Tutorial 1

Question 1



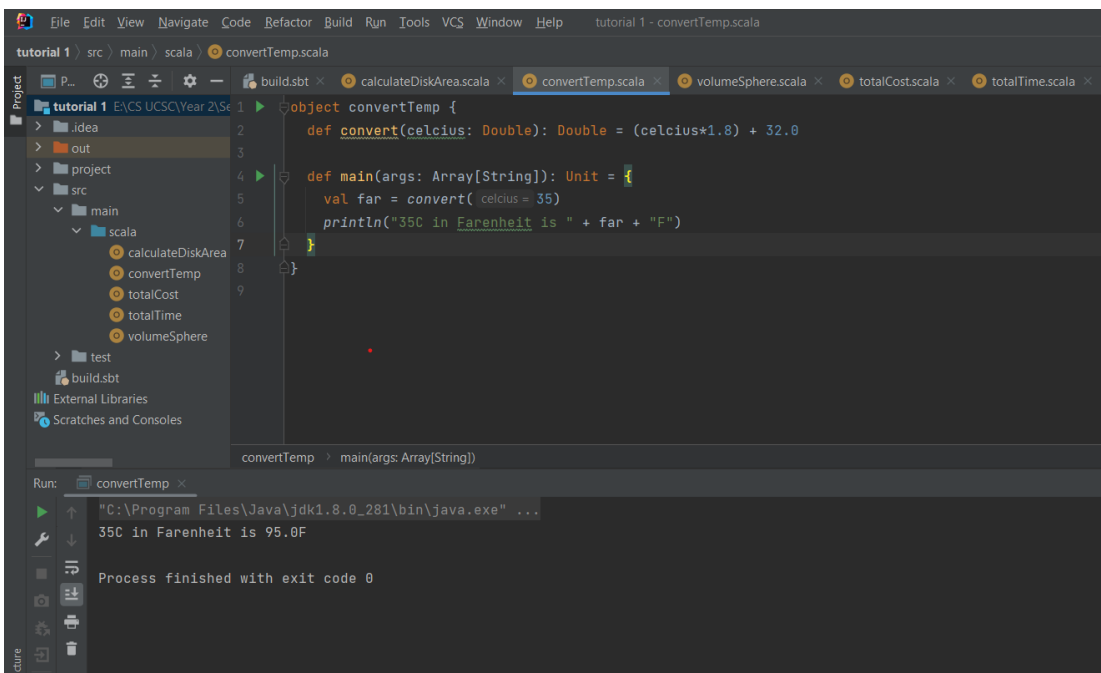
The screenshot shows the IntelliJ IDEA IDE with the file `calculateDiskArea.scala` open. The code defines an object `calculateDiskArea` with a method `calc_area` and a `main` function. The `main` function calls `calc_area` with a radius of 5 and prints the result.

```
1 object calculateDiskArea {  
2   def calc_area(radius: Double): Double = 3.14 * radius * radius  
3  
4   def main(args: Array[String]): Unit = {  
5     val area = calc_area(radius = 5)  
6     println("The area of the disk with radius is " + area)  
7   }  
8 }  
9  
10  
11
```

The Run tab at the bottom shows the execution output:

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...  
The area of the disk with radius is 78.5  
  
Process finished with exit code 0
```

Question 2



The screenshot shows the IntelliJ IDEA IDE with the file `convertTemp.scala` open. The code defines an object `convertTemp` with a method `convert` and a `main` function. The `main` function calls `convert` with a Celsius value of 35 and prints the result in Fahrenheit.

```
1 object convertTemp {  
2   def convert(celcius: Double): Double = (celcius*1.8) + 32.0  
3  
4   def main(args: Array[String]): Unit = {  
5     val far = convert(celcius = 35)  
6     println("35C in Farenheit is " + far + "F")  
7   }  
8 }  
9
```

The Run tab at the bottom shows the execution output:

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...  
35C in Farenheit is 95.0F  
  
Process finished with exit code 0
```

Question 3

The screenshot shows an IDE with the file `volumeSphere.scala` open. The code defines an object `volumeSphere` with a `volume` method and a `main` method. The `main` method calls `volume` with a radius of 5 and prints the result.

```

1 object volumeSphere {
2   def volume(radius: Double): Double = (4 * 3.14 * radius * radius * radius)/3
3 }
4
5 def main(args: Array[String]): Unit = {
6   val vol_sphere = volume( radius = 5)
7   println("The volume of the sphere is " + vol_sphere)
8 }
9

```

The Run console shows the output: "The volume of the sphere is 523.333333333334".

Question 4

The screenshot shows an IDE with the file `totalCost.scala` open. The code defines an object `totalCost` with a `tot_cost` method and a `main` method. The `main` method calls `tot_cost` with 60 books and prints the result.

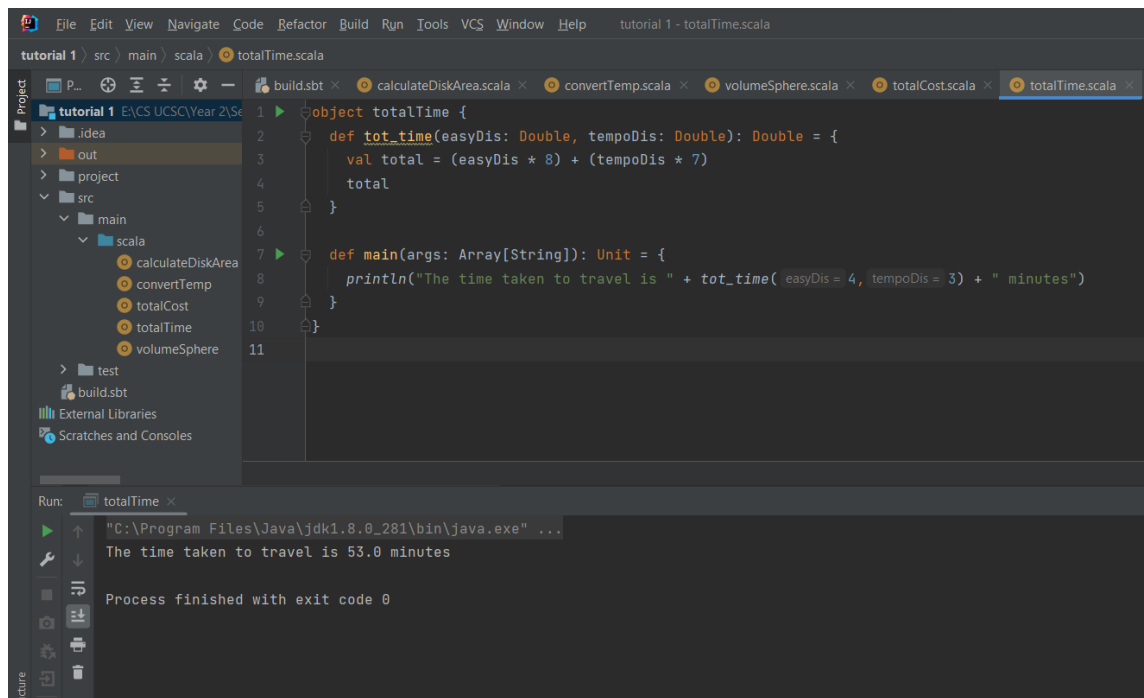
```

1 object totalCost {
2   def tot_cost(books: Int): Double = {
3     val shipping_cost = if(books >= 50) (50 * 3) + (books - 50) * 0.75 else books * 3
4     val tot_cost = (24.95 * 60 * books)/100 + shipping_cost
5     tot_cost
6   }
7 }
8
9 def main(args: Array[String]): Unit = {
10  println("The total wholesale cost for 60 books is Rs." + tot_cost( books = 60))
11 }
12

```

The Run console shows the output: "The total wholesale cost for 60 books is Rs.1055.7".

Question 5



The screenshot shows an IDE window titled "tutorial 1 - totalTime.scala". The code defines an object `totalTime` with a method `tot_time` and a `main` function. The `tot_time` method takes two arguments, `easyDis` and `tempoDis`, both of type `Double`, and returns a `Double`. It calculates the total time as `(easyDis * 8) + (tempoDis * 7)`. The `main` function prints the result of `tot_time(4, 3)`, which is 53.0 minutes. The Run window at the bottom shows the output: "The time taken to travel is 53.0 minutes" and "Process finished with exit code 0".

```
1 object totalTime {  
2   def tot_time(easyDis: Double, tempoDis: Double): Double = {  
3     val total = (easyDis * 8) + (tempoDis * 7)  
4     total  
5   }  
6  
7   def main(args: Array[String]): Unit = {  
8     println("The time taken to travel is " + tot_time(4, 3) + " minutes")  
9   }  
10 }  
11
```

Run: totalTime x
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
The time taken to travel is 53.0 minutes
Process finished with exit code 0