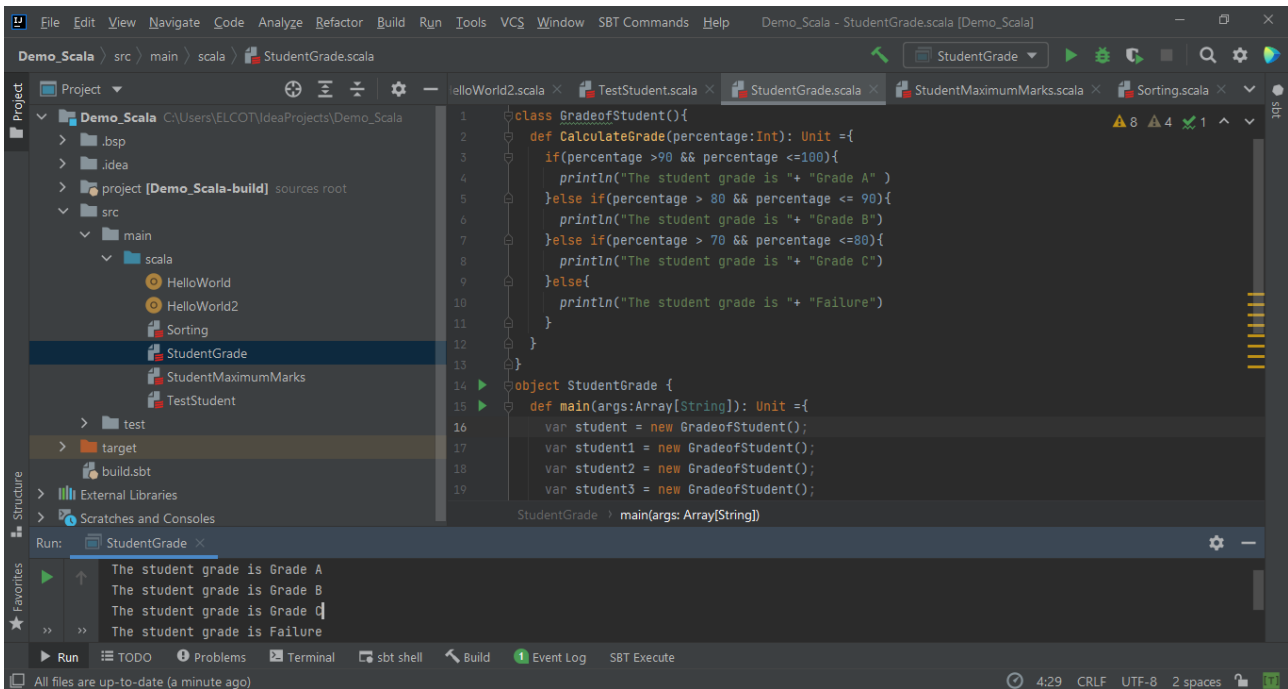


Scala

1. WAP to calculate student grade based on below rules

- a. 90 to 100 -> Grade A
- b. 80 to 90 -> Grade B
- c. 70 to 80 -> Grade C
- d. Failure.



The screenshot shows an IDE with a project named 'Demo_Scala'. The file explorer on the left shows the project structure, including 'src/main/scala'. The main editor displays the 'StudentGrade.scala' file. The code defines a class 'GradeofStudent()' with a method 'CalculateGrade(percentage: Int): Unit' that uses if-else statements to determine the grade based on the percentage. It also defines an object 'StudentGrade' with a 'main' method that creates three instances of 'GradeofStudent()' and prints their grades. The run console at the bottom shows the output: 'The student grade is Grade A', 'The student grade is Grade B', 'The student grade is Grade C', and 'The student grade is Failure'.

```
1 class GradeofStudent(){
2   def CalculateGrade(percentage: Int): Unit = {
3     if (percentage > 90 && percentage <= 100) {
4       println("The student grade is " + "Grade A")
5     } else if (percentage > 80 && percentage <= 90) {
6       println("The student grade is " + "Grade B")
7     } else if (percentage > 70 && percentage <= 80) {
8       println("The student grade is " + "Grade C")
9     } else {
10      println("The student grade is " + "Failure")
11    }
12  }
13 }
14 object StudentGrade {
15   def main(args: Array[String]): Unit = {
16     var student = new GradeofStudent()
17     var student1 = new GradeofStudent()
18     var student2 = new GradeofStudent()
19     var student3 = new GradeofStudent()
20   }
21 }
```

2. WAP to calculate maximum % scored student report from below data.

{id:101,name:raj,cmarks:45,pmarks:55,mmarks:67}

{id:102,name:rajesh,cmarks:65,pmarks:85,mmarks:77}

{id:103,name:suraj,cmarks:43,pmarks:55,mmarks:60}

{id:104,name:tom,cmarks:71,pmarks:65,mmarks:70}

```
1 class MaximumMarks(id:Int,name:String,cmarks:Int,pmarks:Int,mmarks:Int){
2   def PrintMarks(): Unit = {
3     var sum: Int = 0
4     var average: Int = 0
5     sum = cmarks+pmarks+mmarks
6     average = sum/3
7     println("My ID is " + this.id+" My name is " +this.name+" The maximum % scored mark is "+ average)
8   }
9 }
10
11 object StudentMaximumMarks {
12   def main(Args:Array[String]): Unit ={
13     var studmark = new MaximumMarks( id = 101, name = "raj", cmarks = 45, pmarks = 55, mmarks = 67)
14     studmark.PrintMarks()
15     var studmark1 = new MaximumMarks( id = 102, name = "rajesh", cmarks = 65, pmarks = 85, mmarks = 77)
16     studmark1.PrintMarks()
17     var studmark2 = new MaximumMarks( id = 103, name = "suraj", cmarks = 43, pmarks = 55, mmarks = 60)
18     studmark2.PrintMarks()
19     var studmark3 = new MaximumMarks( id = 104, name = "tom", cmarks = 71, pmarks = 65, mmarks = 70)
20     studmark3.PrintMarks()
21   }
22 }
```

Run: StudentMaximumMarks

```
My ID is 101 My name is raj The maximum % scored mark is 55%
My ID is 102 My name is rajesh The maximum % scored mark is 75%
My ID is 103 My name is suraj The maximum % scored mark is 52%
My ID is 104 My name is tom The maximum % scored mark is 68%
```

3. WAP to perform sorting of below data based on id and name(create class, object and a method for sorting in util class)

{id:101,name:raj}

{id:121,name:rajesh}

{id:130,name:suraj}

{id:114,name:tom}

```
1 case class Person(var id: Int, var name: String)
2
3 object Sorting {
4   def main(Args:Array[String]): Unit = {
5     var studentone = Person(101, "raj")
6     var studenttwo = Person(121, "rajesh")
7     var studentthree = Person(130, "suraj")
8     var studentfour = Person(114, "tom")
9     val merge = List(studentone,studenttwo,studentthree,studentfour)
10    //sortBy function ,sorted using given attribute
11    var resultid =merge.sortBy(_.id)
12    var resultName =merge.sortBy(_.name)
13    println("the orderwise id value is " + resultid)
14    println("the orderwise name value is " + resultName)
15    // SortWith function comparing with comparator function like java
16    println("The result of id compared to another id is " + merge.sortWith(_.id < _.id))
17    println("The result of name compared to another name is " + merge.sortWith(_.name < _.name))
18  }
19 }
```

Run: Sorting

```
the orderwise id value is List(Person(101,raj), Person(114,tom), Person(121,rajesh), Person(130,suraj))
the orderwise name value is List(Person(101,raj), Person(121,rajesh), Person(130,suraj), Person(114,tom))
The result of id compared to another id is List(Person(101,raj), Person(114,tom), Person(121,rajesh), Person(130,suraj))
The result of name compared to another name is List(Person(101,raj), Person(121,rajesh), Person(130,suraj), Person(114,tom))
```