Exercise1: Create a function to compute final salary using the following formula.

finalSalary = salary \* ( 1 + incrementPercent) + bonus

where salary, incrementPercent, bonus are input arguments

Solution 1:

def finalSalary(salary: Double, incrementPercent: Double, bonus: Double) : Double = {

val finalSalary = salary \* (1+ incrementPercent) + bonus

return finalSalary

}

finalSalary(20000, 2, 1000)

Exercise2: Modify the above function to apply default valued for below input parameters:

incrementPercent = 0.05

bonus = 5000

Solution 2:

def finalSalary(salary: Double, incrementPercent: Double = 0.05, bonus: Double = 5000) : Double = {

val finalSalary = salary \* (1+ incrementPercent) + bonus

return finalSalary

}

finalSalary(20000)

Exercise3: Store the following list of values using appropriate collections.

EmpName,Loc,Salary

Ramesh,BDC,55000

Janaki,CDC,45000

Kailash,BDC,60000

Lijo,CDC,51000

Create a function which accepts the collection as input and returns average salary

Solution: 3

def averageSalary(empList: List[(String, String, Int)]): Double = {

val length: Int = empList.length

var averageSalary: Double = 0

empList.foreach(emp => {

averageSalary = averageSalary + emp.\_3

})

averageSalary / length

}

val empDetails = List(("Ramesh", "BDC", 55000), ("Janki", "CDC", 45000), ("Kailash", "BDC", 60000), ("Lijo", "CDC", 51000))

val avgSalary: Double = averageSalary(empDetails)

print(avgSalary)

Exercise4: Create a employee class to create employee records using following class variables and methods:

Variables:

empID String

empName String

location String

salary Int

Methods:

getEmpName(empID)

incrementSalary(empID, salIncPct) ->> returns salary post increment

Create a default constructor to generate employee records with all 4 variables.

Solution: 4

class employee {

var empId: String = ""

var empName: String = ""

var location: String = ""

var salary: Int = 0

def getEmpName(empId: String): String = {

empName

}

def incrementSalary(empId: String, salIncPct: Int): Double = {

val finalSalary = salary + (salary \* salIncPct) / 100

finalSalary

}

}

val emp1 = new employee()

emp1.empId = "1"

emp1.empName = "Himanshu"

emp1.location = "BDC"

emp1.salary = 1000

val empName: String = emp1.getEmpName("1")

val finalSalary: Double = emp1.incrementSalary("1", 2)

Exercise5: In the above employee class, create an auxiliary constructor which allows creating employee instances with no location assigned.

Solution 5:

class employee {

var empId: String =""

var empName: String = ""

var salary: Int = 0

var location: String = ""

def this(empId: String, empName: String, salary: Int) {

this()

this.empId = empId

this.empName = empName

this.salary = salary

}

def getEmpName(empId: String): String = {

empName

}

def incrementSalary(empId: String, salIncPct: Int): Double = {

val finalSalary = salary + (salary \* salIncPct) / 100

finalSalary

}

}

val emp1 = new employee()

emp1.empId = "1"

emp1.empName = "Himanshu"

emp1.location = "BDC"

emp1.salary = 1000

val empName: String = emp1.getEmpName("1")

val finalSalary: Double = emp1.incrementSalary("1", 2)

Exercise6: Create a companion object for above employee class with following method

getEmployeeLocation(locID)

returns location using the below HashMap

location(BDC) = Bangalore

location(CDC) = Chennai

location(HDC) = Hyderabad

location(Others) = Unknown

Solution 6:

class employee {

var empId: String =""

var empName: String = ""

var salary: Int = 0

var location: String = ""

def this(empId: String, empName: String, salary: Int) {

this()

this.empId = empId

this.empName = empName

this.salary = salary

}

def getEmpName(empId: String): String = {

empName

}

def incrementSalary(empId: String, salIncPct: Int): Double = {

val finalSalary = salary + (salary \* salIncPct) / 100

finalSalary

}

}

object employee {

def getEmployeeLocation(locId: String): String = {

val hashMap = mutable.HashMap("BDC" -> "Bangalore", "CDC" -> "Chennai", "HDC" -> "Hyderabad", "Others" -> "Unknown")

val value = hashMap.get(locId)

value.get

}

}

val emp1 = new employee()

emp1.empId = "1"

emp1.empName = "Himanshu"

emp1.location = "BDC"

emp1.salary = 1000

val empName: String = emp1.getEmpName("1")

val finalSalary: Double = emp1.incrementSalary("1", 2)

val locationValue: String = employee.getEmployeeLocation("BDC")

print(locationValue)