Development Scenario 1: Personal Finance Tracker

Day 1: Introduction and Setup and Variables and Control Structures

Task 1: Install Kotlin and configure IntelliJ IDEA. Verify the setup by running a "Hello, World!" program.

Sol:

1.First we can go to the website then search IntelliJ IDEA Community Edition.

2.After that Select the Operating System

- Windows
- Mac
- Linux

3. Then after click on download then it navigate to download page.

4.Double click on the downloaded file then click on next to set the configuration

5.Then after select the folder which we can install these particular IntelliJ IDEA.

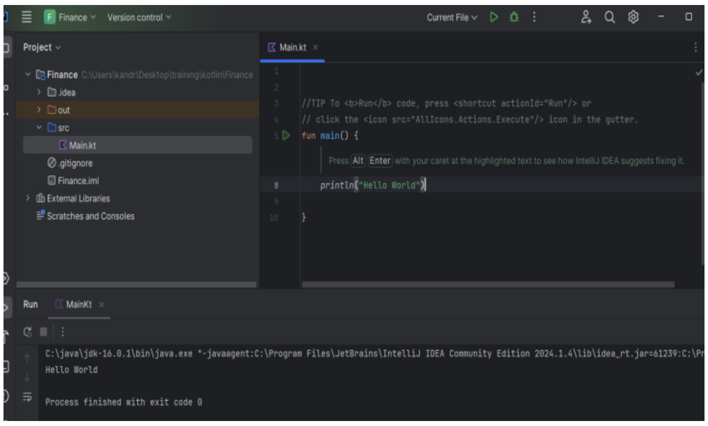
6.Next we have dialogue box we can select the languages like

- .java
- .kt
- .scala
- .groovy etc

7.then click the checkbox and click on next

8.After click on next to configure the files and after click on finish.

9.Then IntelliJ IDEA installed successfully then after set the sample code it will working properly or not



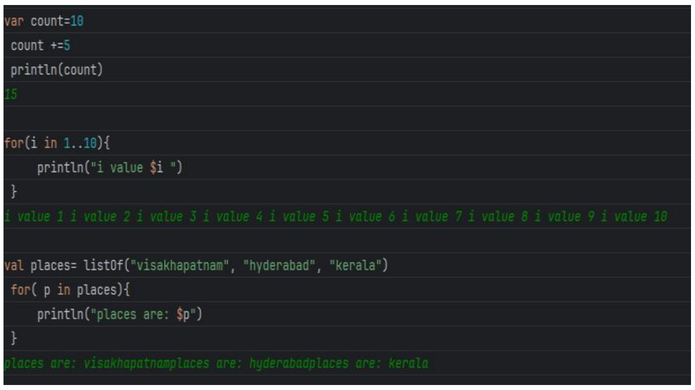
```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
fun main() {
    Press Alt Enter with your caret at the highlighted text to see how IntelliJ IDEA suggests fixing it.
    println("Hello World")

}
```

```
C:\java\jdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1.4\lib\idea_rt.jar=61239:C:\Pr
Hello World

Process finished with exit code 0
```

Task 2: Explore Kotlin REPL (Read-Eval-Print Loop) to familiarize with Kotlin syntax and basic operations.

1.open IntelliJ then menu bar click on tools then click on Kotlin REPL

2.after that we can run the programs



```
var count=10
 count +=5
 println(count)
15

for(i in 1..10){
    println("i value $i ")
 }
i value 1 i value 2 i value 3 i value 4 i value 5 i value 6 i value 7 i value 8 i value 9 i value 10

val places= listOf("visakhapatnam", "hyderabad", "kerala")
 for( p in places){
    println("places are: $p")
 }
places are: visakhapatnamplaces are: hyderabadplaces are: kerala
```

Task 3: Create a Transaction class with properties such as amount, date, and category.

Code:

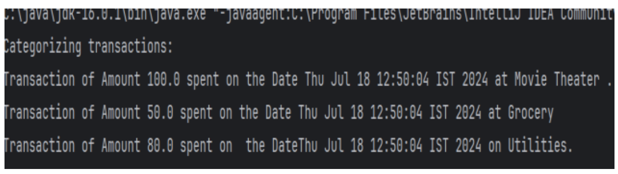
Transaction.kt

```
Import java.util.Date

Class Transaction(val amount: Double, val date: Date , val category: String){

 override fun  toString(): String {

return “Transaction $amount on $date spent on $category”

}

}
```

Main.kt

```
  Fun main(){

Val tan=Transaction(100.0,date=Date(),”Movie”)

Val tan1=Transaction(50.0,date=Date(),”Grocery”)

Val tan2=Transaction(80.0,date=Date(),”Utilities”)

Println(“Categorizing transactions:”)

cateTran(tan)

cateTran(tan1)

cateTran(tan2)

}

Fun cateTran(transa: Transaction){

When(transa.category){

“Movie” -> Println(“Transaction of Amount ${transa.amount} spent on the date ${transa.date} on Movie”

“Grocery” -> Println(“Transaction of Amount ${transa.amount} spent on the date ${transa.date} on Grocery”

“Utilities” -> Println(“Transaction of Amount ${transa.amount} spent on the date ${transa.date} on Utilities”

Else -> Println(“${transa.amount} spent on others”

}
```

Output:



```
C:\java\jdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Communit
Categorizing transactions:
Transaction of Amount 100.0 spent on the Date Thu Jul 18 12:50:04 IST 2024 at Movie Theater .
Transaction of Amount 50.0 spent on the Date Thu Jul 18 12:50:04 IST 2024 at Grocery
Transaction of Amount 80.0 spent on  the DateThu Jul 18 12:50:04 IST 2024 on Utilities.
```

```
class Controlflo(val amount: Double,val date: String,val category: String)
{
    override fun toString(): String {
        return "transaction $amount on $date spent on $category"
    }
}
fun main(){
    val tran1=Controlflo(100.0,"27-06-2024","Food")
    val tran2=Controlflo(50.0,"25-06-2024","Entertainment")
    val tran3=Controlflo(350.0,"22-06-2024","Utilities")
    cateroryupdate(tran1)
    cateroryupdate(tran2)
    cateroryupdate(tran3)

}

fun cateroryupdate(tran: Controlflo)
{
    when(tran.category){
        "Food" -> println("Transaction ${tran.amount} spent on Food on the
date ${tran.date} on ${tran.category}")
        "Entertainment" -> println("transaction ${tran.amount} spent on the
date ${tran.date} on ${tran.category}")
        "Utilities" -> println("transaction ${tran.amount} spent on the
date ${tran.date} on ${tran.category}")
        else -> println("enter valid category")
    }
}
```

Development Scenario 2: Event Management System

Day 1: Introduction and Setup

Task 1: Set up the Kotlin development environment and write a simple Kotlin script to validate the setup.

```
import java.time.LocalDateTime
import java.time.format.DateTimeFormatter

fun main() {
    val currentDateTime = LocalDateTime.now()
    println("Current Date and Time: $currentDateTime")
    val formattedDateTime =
currentDateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))
    println("Formatted Date and Time: $formattedDateTime")
}
```

task2: Experiment with Kotlin's string templates to create dynamic welcome messages.

```
 fun main() {
    val userName = "Swamy"
    fun getGreeting(name: String): String {
        return "Hello, $name! Welcome to our world"
    }
    val welcomeMessage = "Dear $userName,\n${getGreeting(userName)}"

    println(welcomeMessage)
}
```

task3:

Define data types to represent event details such as name, date, and attendee count.

```
import java.time.LocalDate

data class Event(
    val name: String,
    val date: LocalDate,
    val attendeeCount: Int
)

fun main() {
    val event = Event("Birth day", LocalDate.of(2024, 8, 15), 250)
    val event2=Event("Farewall Day",LocalDate.of(2024,6,23),450)
    println("Event Name: ${event.name} Event Date ${event.date} Attendance
count ${event.attendeeCount}")
    println("Event Name: ${event2.name} Event Date ${event2.date}
Attendance count ${event2.attendeeCount}")
}
```

```
task4: fun main()
{
    println("enter a value")
    val a= readLine()
    val a1=a?.toIntOrNull()
    if(a1==null){
        println("enter valid number")
        return
    }
    if(a1%2==0){
        println("a1 is even number $a1")
    }
    else if(a1%2==1){
        println("a1 is odd number $a1")
    }
    else{
        println("a1 is either zero or negative number $a1")
    }
}
```

```
fun main(){
    println("enter age")
    val a= readLine()
    val age=a?.toIntOrNull()
    if(age==null){
        println("enter valid age")
        return
    }
    val res = when(age){
        in 0 ..12 ->"child"
        in 13 .. 19 ->"teenager"
        in 20.. 60 ->"adult"
        in 61 .. 100 -> "senior citizen"
        else ->"Invalid age"
    }
    println(res)
}
```