

Practical: 1

AIM- Develop a Kotlin program for demonstrating various programming concepts.

Submitted By: Patel Vaishvi
Enrollment number:22012011164



**Ganpat
University**

॥ विद्यया समाजोत्कर्षः ॥

U.V. Patel
College of
Engineering

Department of Computer
Engineering/Information Technology

1. Store & Display Values in Different Variables: Create and display variables of different data types, including Integer, Double, Float, Long, Short, Byte, Char, Boolean, and String

Code:

```
fun main(){
    val a:Int=10
    println("Integer value is $a")
    val b:Float=3.14f
    println("Float value is $b")
    val c:Double=2.167
    println("Double value is $c")
    val d:String="hello"
    println("String value is $d")
    val e:Boolean=true
    println("Boolean value is $e")
    val f:Char='V'
    println("Char value is $f")
    val g:Long=1500000000L
    println("Long value is $g")
    val h:Short=1000
    println("Short value is $h")
    val i:Byte=127
    println("Byte value is $i")
}
```

output:

```
Integer value is 10
Float value is 3.14
Double value is 2.167
String value is hello
Boolean value is true
Char value is V
Long value is 1500000000
Short value is 1000
Byte value is 127
```

2. Type Conversion: Perform type conversions such as Integer to Double, String to Integer, and String to Double.

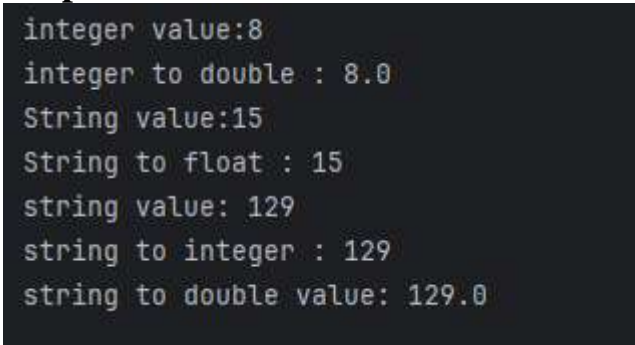
Code:

```
fun main(){
```

```
val x:Int=8
val y:Double=x.toDouble()
println("integer value:$x")
println("integer to double : $y")

val z:String="15"
val w=z.toInt()
println("String value:$z")
println("String to float : $w")
val i:String="129"
val j=Integer.valueOf(i)
println("string value: $i")
println("string to integer : $j")
val m:Double=i.toDouble()
println("string to double value: $m")
}
```

Output:

A screenshot of a terminal window with a dark background and light-colored text. It displays the output of the provided Kotlin code. The output consists of seven lines, each showing a variable name followed by its value and a colon. The values are: 8, 8.0, 15, 15, 129, 129, and 129.0.

```
integer value:8
integer to double : 8.0
String value:15
String to float : 15
string value: 129
string to integer : 129
string to double value: 129.0
```

3. Scan student's information and display all the data: Input and display data of students, including their name, enrolment no, branch,etc.

Code:

```
fun main(args: Array<String>) {
    println("Enter your enrollment number:")
    val enno = readLine()
    println("Enter your name:")
    val name= readLine()
    println("Enter your branch:")
    val branch = readLine()
    println("Enter your batch:")
    val batch = readLine()
    println("Enter your college name:")
    val college = readLine()
    println("Enter your university name:")
}
```

Practical: 1

```
val university = readLine()
println("Enter your age:")
var age: Int = Integer.valueOf(readLine())
println("*****")
println("student's data:")
println("enrollment number:$enno")
println("name:$name")
println("branch:$branch")
println("batch:$batch")
println("college:$college")
println("university:$university")
println("age:$age")
println("*****")
}
```

Output:

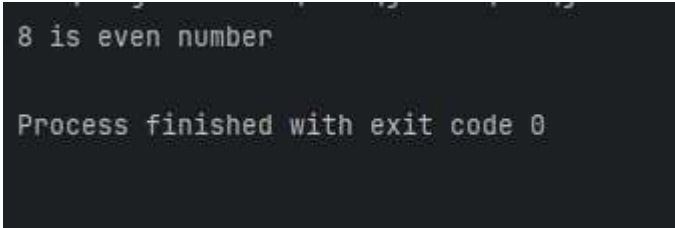
```
Enter your enrollment number:
22012011164
Enter your name:
vaishvi
Enter your branch:
ce
Enter your batch:
ceb3
Enter your college name:
uvpce
Enter your university name:
ganpat university
Enter your age:
20
*****
student's data:
enrollment number:22012011164
name:vaishvi
branch:ce
batch:ceb3
college:uvpce
university:ganpat university
age:20
*****
```

4. Check Odd or Even Numbers: Determine whether a number is odd or even using control flow within println() method.

Code:

```
fun main(){
    val n:Int=8
    println(
        if(n%2==0){
            "$n is even number"
        }
        else{
            "$n is odd number"
        }
    )
}
```

Output:



```
8 is even number

Process finished with exit code 0
```

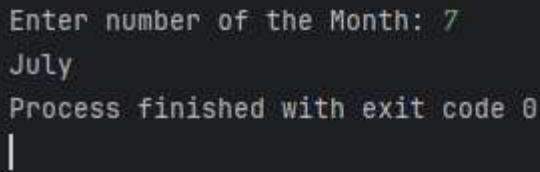
5. Display Month Name: Use a when expression to display the month name based on user input.

Code:

```
fun main(args : Array<String>) {
    print("Enter number of the Month: ")
    var monthOfYear = readLine()!!.toInt()
    var month= when(monthOfYear) {
        1->"January"
        2->"February"
        3->"March"
        4->"April"
        5->"May"
        6->"June"
        7->"July"
```

```
8->"August"
9->"September"
10->"October"
11->"November"
12->"December"
else-> "enter valid number"
}
print(month)
}
```

Output:



```
Enter number of the Month: 7
July
Process finished with exit code 0
|
```

6. User-Defined Function: Create a user-defined function to perform arithmetic operations (addition, subtraction, multiplication, division) on two numbers

Code:

```
fun main(){
    println("Enter number1")
    val number1=readLine()!!.toInt()
    println("Enter number2")
    val
    number2=readLine()!!.t
    oInt()var result:Int

    result = number1 + number2
    println("number1 + number2 = $result")

    result = number1 - number2
    println("number1 - number2 = $result")

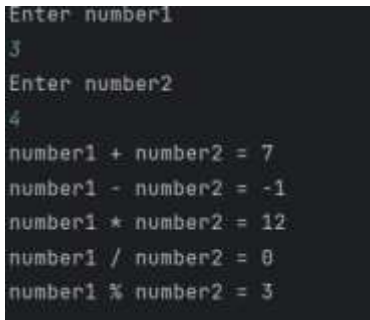
    result = number1 * number2
    println("number1 * number2 = $result")

    result = number1 / number2
    println("number1 / number2 = $result")
}
```

Practical: 1

```
result = number1 % number2
println("number1 % number2 = $result")
}
```

Output:



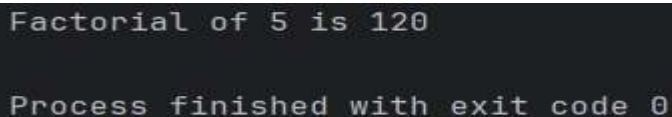
```
Enter number1
3
Enter number2
4
number1 + number2 = 7
number1 - number2 = -1
number1 * number2 = 12
number1 / number2 = 0
number1 % number2 = 3
```

7. Factorial Calculation with Recursion: Calculate the factorial of a number using recursion.

```
Code: fun factorial(n: Int): Int {
    return if (n == 0 || n == 1) {
        1
    } else {
        n * factorial(n - 1)
    }
}
```

```
fun main() {
    val number = 5
    val result = factorial(number)
    println("Factorial of $number is $result")
}
```

Output:



```
Factorial of 5 is 120

Process finished with exit code 0
```

8. Working with Arrays: Explore array operations such as

Practical: 1

Arrays.deepToString(), contentDeepToString(), IntArray.joinToString(), and use them to print arrays. Utilize various loop types like range, downTo, until, etc., to manipulate arrays. Sort an array of integers both without using built-in functions and with built-in functions.

Code:

```
fun main() {
    val intArray = intArrayOf(7, 3, 5, 2)
    println("Original array: ${intArray.joinToString(", ")}")

    println("Array with joinToString: ${intArray.joinToString(", ")}")
    println("Array with contentToString: ${intArray.contentToString()}")
    println("Array with Arrays.deepToString:
    ${java.util.Arrays.deepToString(arrayOf(intArray))}")

    println("Array elements using range:")
    intArray.indices.forEach { i -> println("Element at index $i: ${intArray[i]}") }

    println("Array elements using downTo:")
    (intArray.size - 1 downTo 0).forEach { i -> println("Element at index $i: ${intArray[i]}")
    }

    println("Array elements using until:")

    (0 until intArray.size).forEach { i -> println("Element at index $i: ${intArray[i]}") }

    val sortedArrayBuiltIn = intArray.sortedArray()
    println("Sorted array using built-in function: ${sortedArrayBuiltIn.joinToString(", ")}")

    val arrayToSort = intArray.copyOf()
    bubbleSort(arrayToSort)
    println("Sorted array without built-in function: ${arrayToSort.joinToString(", ")}")
}
```


Practical: 1

```
fun bubbleSort(array: IntArray) { for (i in array.indices) {  
for (j in 0 until array.size - i - 1) { if (array[j] > array[j + 1]) {  
array[j] = array[j].also { array[j] = array[j + 1]; array[j + 1] = it }  
}  
}  
}  
}
```

Output:

```
Original array: 7, 3, 5, 2  
Array with joinToString: 7, 3, 5, 2  
Array with contentToString: [7, 3, 5, 2]  
Array with Arrays.deepToString: [[7, 3, 5, 2]]  
Array elements using range:  
Element at index 0: 7  
Element at index 1: 3  
Element at index 2: 5  
Element at index 3: 2  
Array elements using downTo:  
Element at index 3: 2  
Element at index 2: 5  
Element at index 1: 3  
Element at index 0: 7  
Array elements using until:  
Element at index 0: 7  
Element at index 1: 3  
Element at index 2: 5  
Element at index 3: 2  
Sorted array using built-in function: 2, 3, 5, 7  
Sorted array without built-in function: 7, 7, 7, 7  
  
Process finished with exit code 0
```

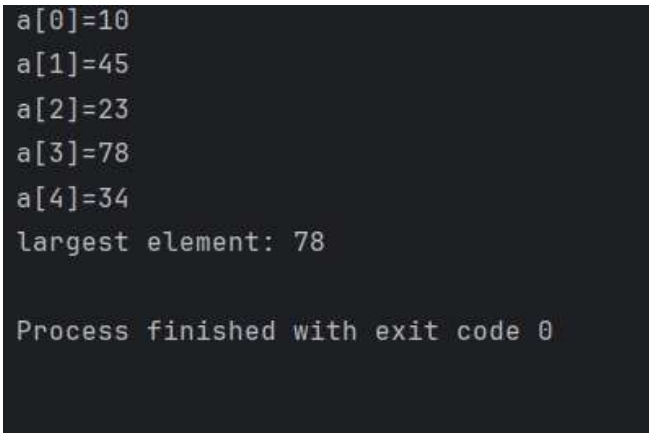
9. Find Maximum Number from ArrayList: Write a program to find the maximum number from an ArrayList of integers.

Code: fun main() {
val numbers = arrayListOf(10, 45, 23, 78, 34)
var a = numbers[0]
var b = numbers[1]
var c = numbers[2]

Practical: 1

```
var d = numbers[3]
var e = numbers[4]
println("a[0]=$a")
println("a[1]=$b")
println("a[2]=$c")
println("a[3]=$d")
println("a[4]=$e")
val maxNumber = numbers.max()
println("largest element: $maxNumber")
}
```

Output:



```
a[0]=10
a[1]=45
a[2]=23
a[3]=78
a[4]=34
largest element: 78

Process finished with exit code 0
```

10. Class and Constructor Creation: Define different classes and constructors. Create a "Car" class with properties like type, model, price, owner, and miles driven. Implement functions to get car information, original car price, current car price, and display car information

Code: Class Car(val type: String, val model: String, private val originalPrice: Double, var owner: String, var milesDriven: Int) {

```
fun getCarInfo(): String {
return "Type: $type, Model: $model, Owner: $owner, Miles Driven: $milesDriven"
}
```

```
fun getOriginalPrice(): Double { return originalPrice
}
```

Practical: 1

```
fun getCurrentPrice(): Double {  
  
    val depreciation = milesDriven / 10000 * 0.1 * originalPrice  
    return originalPrice - depreciation  
}  
  
fun displayCarInfo() {  
  
    println(getCarInfo())  
    println("Original Price: ${getOriginalPrice()}")  
    println("Current Price: ${getCurrentPrice()}")  
}  
  
fun main() {  
    val car1 = Car("BMW", "2018", 100000.0, "aarvi", 105)  
    car1.displayCarInfo()  
  
    val car2 = Car("BMW", "2019", 400000.0, "bindu", 20)  
    car2.displayCarInfo()  
  
    val car3 = Car("Toyota", "2017", 1080000.0, "priyanshi", 100) car3.displayCarInfo()  
  
    val car4 = Car("Maruti", "2020", 4000000.0, "drasti", 200) car4.displayCarInfo()  
}
```

Output:

```
Type: BMW, Model: 2018, Owner: aarvi, Miles Driven: 105
Original Price: $100000.0
Current Price: $100000.0
Type: BMW, Model: 2019, Owner: bindu, Miles Driven: 20
Original Price: $400000.0
Current Price: $400000.0
Type: Toyota, Model: 2017, Owner: priyanshi, Miles Driven: 100
Original Price: $1080000.0
Current Price: $1080000.0
Type: Maruti, Model: 2020, Owner: drasti, Miles Driven: 200
Original Price: $4000000.0
Current Price: $4000000.0

Process finished with exit code 0
```

11.Operator Overloading and Matrix Operations: Explain operator overloading and implement matrix addition, subtraction, and multiplication using a "Matrix" class. Overload the toString() function in the "Matrix" class for customized output.

Code:

```
class Matrix(private val data: List<List<Int>>){ val rows: Int = data.size
val cols: Int = data[0].size
```

```
operator fun plus(other: Matrix): Matrix {
require(rows == other.rows && cols == other.cols) { "Matrix dimensions must agree" }
val result = List(rows) { i ->
```

```
List(cols) { j ->
data[i][j] + other.data[i][j]
}
}
return Matrix(result)
}
```

```
operator fun minus(other: Matrix): Matrix {
require(rows == other.rows && cols == other.cols) { "Matrix dimensions must agree" }
val result = List(rows) { i ->
```

Practical: 1

```
List(cols) { j ->
data[i][j] - other.data[i][j]
}
}
return Matrix(result)
}
```

```
operator fun times(other: Matrix): Matrix {
require(cols == other.rows) { "Matrix A's columns must equal Matrix B's rows" }
val result = List(rows) { i ->
List(other.cols) { j ->
(0 until cols).sumOf { k -> data[i][k] * other.data[k][j] }
}
}
return Matrix(result)
}
```

```
override fun toString(): String {
return data.joinToString("\n") { row -> row.joinToString("\t") }
}
}
```

```
fun readMatrix(): Matrix {
println("Enter matrix dimensions (rows and columns):") val (rows, cols) =
readLine()!!.split(" ").map { it.toInt() }
println("Enter matrix values row by row, with each row on a new line and values separated
by spaces:")
}
```

```
val data = List(rows) {
readLine()!!.split(" ").map { it.toInt() }
}
```

Practical: 1

```
return Matrix(data)
}
```

```
fun main() {
println("Enter matrix A:")
```

```
val matrixA = readMatrix() println("Enter matrix B:") val matrixB = readMatrix()
```

```
println("Matrix A:")
println(matrixA)
```

```
println("Matrix B:")
println(matrixB)
```

```
println("Matrix A + Matrix B:")
println(matrixA + matrixB)
```

```
println("Matrix A - Matrix B:")
println(matrixA - matrixB)
```

```
println("Matrix A * Matrix B:")
println(matrixA * matrixB)
}
```

Output:

Practical: 1

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ J
Enter matrix A:
Enter matrix dimensions (rows and columns):
2 2
Enter matrix values row by row, with each row on a new line and values separated by spaces:
3 6
9 2
Enter matrix B:
Enter matrix dimensions (rows and columns):
2 2
Enter matrix values row by row, with each row on a new line and values separated by spaces:
4 8
2 4
Matrix A:
3   6
9   2
Matrix B:
4   8
2   4
Matrix A + Matrix B:
7   14
11   6
Matrix A - Matrix B:
-1  -2
7   -2
```

```
Matrix A * Matrix B:
24  48
40  80

Process finished with exit code 0
```

EXERCISE QUESTION

1. **Swap Value of two variables without using third variable and with using third variable.**

Code:

```
fun main() { var a = 5 var b = 10
```

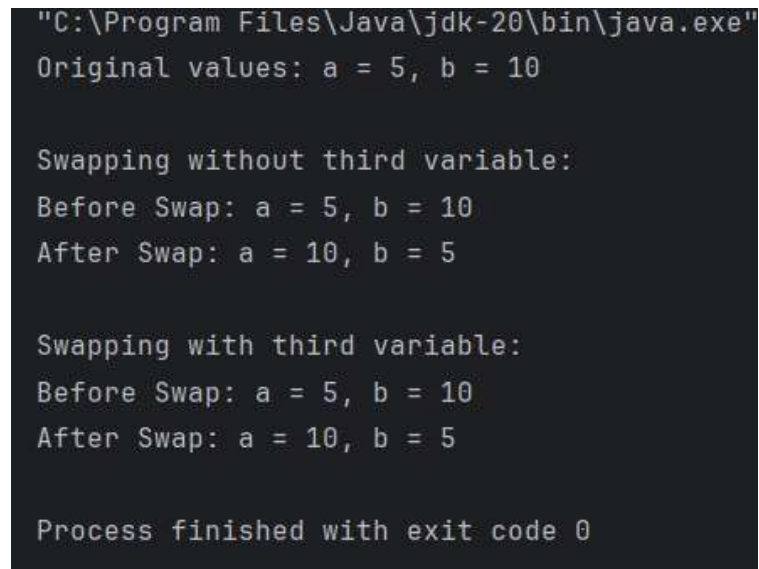
```
println("Original values: a = $a, b = $b") swapWithoutThirdVariable(a, b)
swapWithThirdVariable(a, b)
}
```

Practical: 1

```
fun swapWithoutThirdVariable(a: Int, b: Int) { var x = a
var y = b
println("\nSwapping without third variable:")
println("Before Swap: a = $x, b = $y") x = x + y
y = x - y x = x - y
println("After Swap: a = $x, b = $y")
}
```

```
fun swapWithThirdVariable(a: Int, b: Int) { var x = a
var y = b
println("\nSwapping with third variable:") println("Before Swap: a = $x, b = $y") val
temp = x
x = y
y = temp
println("After Swap: a = $x, b = $y")
}
```

Output:



```
"C:\Program Files\Java\jdk-20\bin\java.exe"
Original values: a = 5, b = 10

Swapping without third variable:
Before Swap: a = 5, b = 10
After Swap: a = 10, b = 5

Swapping with third variable:
Before Swap: a = 5, b = 10
After Swap: a = 10, b = 5

Process finished with exit code 0
```

2. Create two class named as Product and Laptop. Inherit with this information: Product class should be parent and child class should be Laptop class.

Add Product Name, Quantity, Amount per Quantity in Product class. In

Practical: 1

Laptop class add CPU name, RAM size, HDD Size, etc. of Laptop configuration.

Create primary and secondary Constructor of both class.

If Primary constructor is there then can we create secondary constructor in inheritance?

If we can create secondary and primary constructor both in child class then what is restriction if parent have more than two different secondary constructor?

Create List of 5 laptops in ArrayList and display all objects information

Code:

```
open class Product(val name: String, var quantity: Int, var amountPerQuantity: Double) {  
    constructor(name: String, quantity: Int) : this(name, quantity, 0.0) {  
    }  
}
```

```
override fun toString(): String {  
    return "Product(name='$name', quantity=$quantity,  
    amountPerQuantity=$amountPerQuantity)"  
}  
}
```

```
class Laptop(name: String, quantity: Int, amountPerQuantity: Double, val cpuName:  
String, val ramSize: Int, val hddSize: Int) : Product(name, quantity, amountPerQuantity) {  
    constructor(name: String, quantity: Int, cpuName: String, ramSize: Int, hddSize: Int) :  
    this(name, quantity, 0.0, cpuName, ramSize, hddSize) {  
    }  
}
```

```
override fun toString(): String {  
    return "${super.toString()}, Laptop(cpuName='$cpuName', ramSize=$ramSize,  
    hddSize=$hddSize)"  
}  
}
```

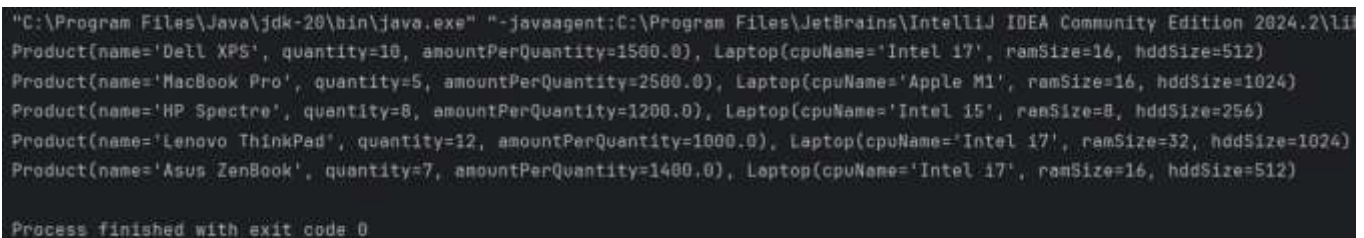
```
fun main() {  
    val laptops = arrayListOf(  

```

Practical: 1

```
Laptop("Dell XPS", 10, 1500.0, "Intel i7", 16, 512),  
Laptop("MacBook Pro", 5, 2500.0, "Apple M1", 16, 1024),  
Laptop("HP Spectre", 8, 1200.0, "Intel i5", 8, 256),  
Laptop("Lenovo ThinkPad", 12, 1000.0, "Intel i7", 32, 1024),  
Laptop("Asus ZenBook", 7, 1400.0, "Intel i7", 16, 512)  
)  
laptops.forEach { println(it) }  
}
```

Output:



```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2\lib\idea_rt.jar" 160122011164_Vaishvi  
Product(name='Dell XPS', quantity=10, amountPerQuantity=1500.0), Laptop(cpuName='Intel i7', ramSize=16, hddSize=512)  
Product(name='MacBook Pro', quantity=5, amountPerQuantity=2500.0), Laptop(cpuName='Apple M1', ramSize=16, hddSize=1024)  
Product(name='HP Spectre', quantity=8, amountPerQuantity=1200.0), Laptop(cpuName='Intel i5', ramSize=8, hddSize=256)  
Product(name='Lenovo ThinkPad', quantity=12, amountPerQuantity=1000.0), Laptop(cpuName='Intel i7', ramSize=32, hddSize=1024)  
Product(name='Asus ZenBook', quantity=7, amountPerQuantity=1400.0), Laptop(cpuName='Intel i7', ramSize=16, hddSize=512)  
  
Process finished with exit code 0
```

3. Create two class named as Person and Student. Inherit with this information: Person class should be parent and child class should be Student class.

Add first name, last name, age in Person class. In Laptop class add enrollment no, branch, class, lab batch, etc.

Create primary and secondary Constructor of both class.

Code:

```
open class Person(val firstName: String, val lastName: String, var age: Int) {  
    constructor(firstName: String, lastName: String) : this(firstName, lastName, 0)
```

```
    override fun toString(): String {  
        return "Person(firstName='$firstName', lastName='$lastName', age=$age)"  
    }  
}
```

```
class Student(firstName: String, lastName: String, age: Int, val enrollmentNo: String, val  
branch: String, val classLevel: String, val labBatch: String) : Person(firstName, lastName,  
age) {
```

Practical: 1

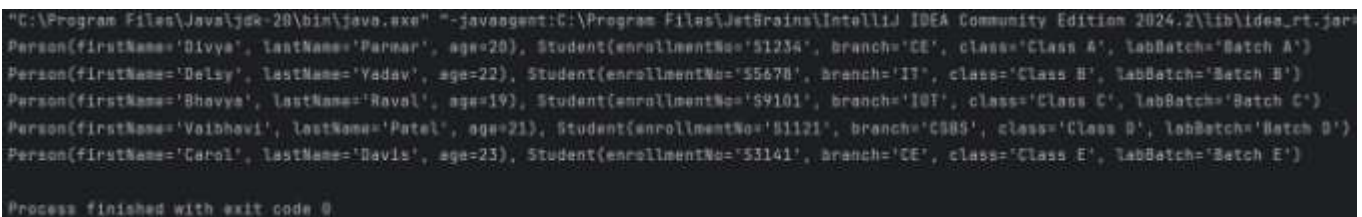
```
constructor(firstName: String, lastName: String, enrollmentNo: String, branch: String,
classLevel: String, labBatch: String) : this(firstName, lastName, 0, enrollmentNo, branch,
classLevel, labBatch)
```

```
override fun toString(): String {
return "${super.toString()}, Student(enrollmentNo='$enrollmentNo', branch='$branch',
```

```
class='$classLevel', labBatch='$labBatch')"
}
}
```

```
fun main() {
val students = arrayListOf(
Student("Divya", "Parmar", 20, "S1234", "CE", "Class A", "Batch A"),
Student("Delsy", "Yadav", 22, "S5678", "IT", "Class B", "Batch B"),
Student("Bhavya", "Raval", 19, "S9101", "IOT", "Class C", "Batch C"),
Student("Vaibhavi", "Patel", 21, "S1121", "CSBS", "Class D", "Batch D"),
Student("Carol", "Davis", 23, "S3141", "CE", "Class E", "Batch E")
)
students.forEach { println(it) }
}
```

Output:



```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2\lib\idea_rt.jar"
Person(firstName='Divya', lastName='Parmar', age=20), Student(enrollmentNo='S1234', branch='CE', class='Class A', labBatch='Batch A')
Person(firstName='Delsy', lastName='Yadav', age=22), Student(enrollmentNo='S5678', branch='IT', class='Class B', labBatch='Batch B')
Person(firstName='Bhavya', lastName='Raval', age=19), Student(enrollmentNo='S9101', branch='IOT', class='Class C', labBatch='Batch C')
Person(firstName='Vaibhavi', lastName='Patel', age=21), Student(enrollmentNo='S1121', branch='CSBS', class='Class D', labBatch='Batch D')
Person(firstName='Carol', lastName='Davis', age=23), Student(enrollmentNo='S3141', branch='CE', class='Class E', labBatch='Batch E')
Process finished with exit code 0
```