



## **Assignment -1**

<b>Subject</b>	OOP
<b>Name</b>	Palak Abhay Pardeshi
<b>Class</b>	CS-D
<b>Roll No.</b>	72

**Problem Statement:** A VIT Melange committee is conducting auditions to admit interested candidates. You need to implement a Participant class for the dance club based on the class diagram and description given below.

Implement Student class using following Concepts

- All types of Constructors
- Static variables and instance variables
- Static blocks and instance blocks
- Static methods and instance methods

<b>C-Participant</b>
<ul style="list-style-type: none"><li>• <u>counter</u>: int</li><li>• registrationId: String</li><li>• name: String</li><li>• contactNumber: long</li><li>• branch: String</li></ul>
<ul style="list-style-type: none"><li>○ <b>C</b>Participant(name:String,contactNumber:long,branch:String)</li><li>○ getRegistrationId(): String</li><li>○ <u>getCounter()</u>: int</li><li>○ <u>setCounter(counter:int):void</u></li></ul>

- getName():String
- setName(name:String):void
- getContactNumber():long
- setContactNumber(contactNumber:long):void
- getBranch():String
- setBranch(branch:String):void

## Method Description

### Participant(String name, long contactNumber, String branch)

- Initialize the name, contactNumber and branch instance variables appropriately with the values passed to the constructor.
- Generate the registrationId using the static variable counter.
- The value of registrationId should start from 'D1001' and the numerical part should be incremented by 1 for the subsequent values.
- Initialize the counter in static block.

Implement the appropriate getter and setter methods.

Test the functionalities using the provided Tester class. Create two or more Participant objects and validate that the values of the member variables are proper.

## Input:

```
package Assignment1; //Package

public class Melange { //Class
    //Declaring variables with access specifiers
    private String name;//Instance variables
    private String branch;
    private String registrationId;
    private long ContactNumber;
    private static int counter;//Static variable

    static //Initializer block for counter variable
    {
        counter=1001;
    }
    //Constructor with three parameters
    //Initializes variables with values passed as arguments
    public Melange(String name, long ContactNumber, String branch)
    {
        //this keyword is used to refer to the current instance of the
```

```

class
    this.name=name;
    this.branch= branch;
    this.ContactNumber=ContactNumber;
    this.registrationId="D"+ counter;
    Melange.counter++; //Incrementing counter variable after each object
of class in created
    //Calling counter variable with constructor
    }
//Getter and Setter methods for all variables
    public String getName() {
        return name;
    }
//Using this and return to accept and display results respectively
    public void setName(String name) {
        this.name = name;
    }

    public String getBranch() {
        return branch;
    }
//Return displays values received as arguments to constructors
    public void setBranch(String branch) {
        this.branch = branch;
    }

    public long getContactNumber() {
        return ContactNumber;
    }

    public void setContactNumber(long contactNumber) {
        this.ContactNumber = contactNumber;
    }

    public static int getCounter() {
        return counter;
    }

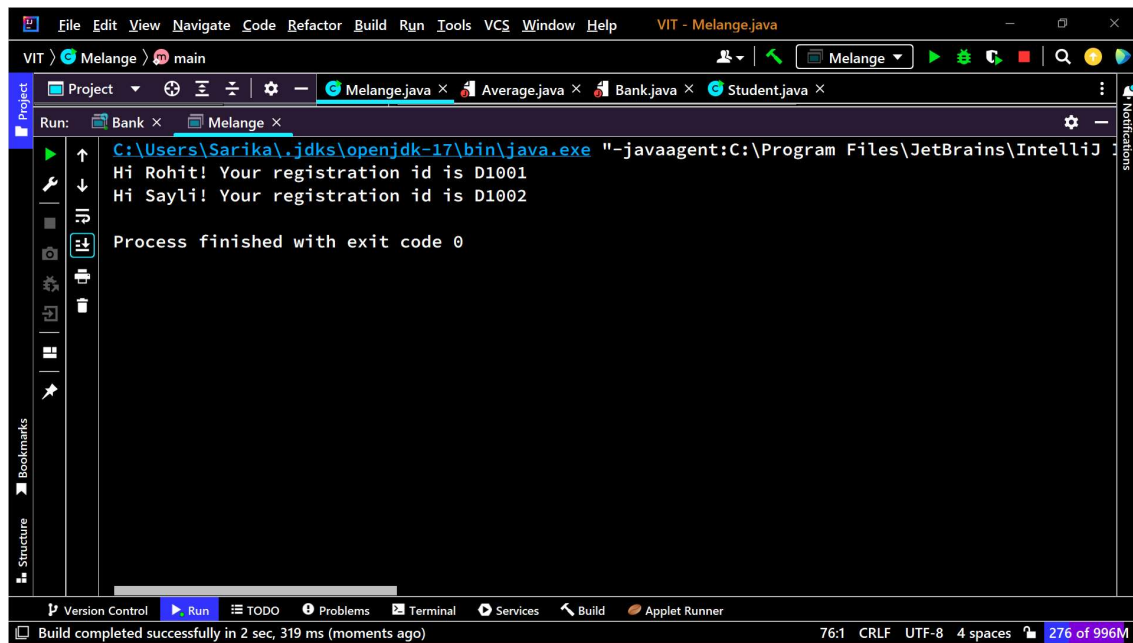
    public static void setCounter(int counter) {
        Melange.counter = counter;
    }

    public String getRegistrationId() {
        return registrationId;
    }
/*getName() and getRegistrationId() methods are used to retrieve
the values of the name and registrationId instance variables of the
student1 and student2 objects, respectively
*/
    public static void main(String[] args) //main class
    {
        //Objects of Melange class using new
        Melange student1 = new Melange("Rohit", 1234567889, "Computer" );
        Melange student2 = new Melange("Sayli", 1988612300, "Mechanical");
//Printing output
        System.out.println("Hi " +student1.getName()+"! Your registration
id is "+ student1.getRegistrationId());

        System.out.println("Hi " +student2.getName()+"! Your registration
id is "+ student2.getRegistrationId());

```

## Output:



```
Run: Bank × Melange ×
C:\Users\Sarika\.jdk\openjdk-17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ I...
Hi Rohit! Your registration id is D1001
Hi Sayli! Your registration id is D1002

Process finished with exit code 0
```

## Student Class

### Input:

```
class Student{
    static int cnt = 12220001;
    int PRN;
    String Name;
    String division;

    static {
        System.out.println("Static block");
    }

    {
        System.out.println("Instance block");
    }

    public Student() {
    }

    public Student(String name, String div) {
        this.Name = name;
        this.division = div;
        this.PRN = cnt;
        cnt++;
    }

    public Student(Student another) {
        this(another.Name, another.division);
    }
}
```

```

static int getCurrentPRN() {
    return cnt;
}

String getDivision() {
    return this.division;
}

String getName() {
    return this.Name;
}

int getPRN() {
    return this.PRN;
}

public static void main(String args[]) {
    Student student = new Student("Palak Pardeshi", "CS D");
    System.out.println("Current PRN: " + getCurrentPRN());
    System.out.println("Student Name: " + student.getName());
    System.out.println("Student Division: " + student.getDivision());
    System.out.println("Student PRN: " + student.getPRN());
}
}

```

## Output:

The screenshot shows an IDE window titled "VIT - Student.java". The "Run" tab is active, displaying the output of the program. The output is as follows:

```

Run: C:\Users\Sarika\.jdk\openjdk-17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ
Static block
Instance block
Current PRN: 12220002
Student Name: Palak Pardeshi
Student Division: CS D
Student PRN: 12220001

Process finished with exit code 0

```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help), a toolbar with icons for running and debugging, and a status bar at the bottom indicating "All files are up-to-date (moments ago)" and "634 of 996M".