



Southeast University

Department of Computer Science and Engineering (CSE)

School of Sciences and Engineering

Semester: (Summer, Year: 2025)

LAB REPORT NO: 02

Course Title: Introduction to Programming Language II (Java) Lab

Course Code: CSE282.2

Batch: 65

**Lab Experiment Name: Declaration of methods and constructors
in JAVA.**

Student Details

	Name	ID
1.	Md. Mahmud Hossain	2023200000799

Submission Date : 30-07-25

Course Teacher's Name : Dr. Mohammed Ashikur Rahman

Lab Report Status

Marks:

Comments:.....

Signature:.....

Date:.....

Lab Task 2: Declaration of methods and constructors in JAVA.

OBJECTIVES:

- To be familiar with the static keyword
- To be familiar with parameterized constructor
- To be familiar with the difference between class properties/methods and object properties and methods

PROBLEM:

1. Create an Employee class with properties such as name, age, designation, salary (object variables) and company name, company address (class variables). Implement a parameterized constructor to initialize 3 objects. Include an object method to display the employee details and a class method to display the total number of employees.
2. Create a Book class with properties such as title, author, year (object variables) and genre (class variable). Implement a parameterized constructor to initialize 3 objects. Include an object method to display the book details and a class method to display the total number of books.
3. Create a Student class with properties such as id, name, department, cgpa (object variables) and university (class variable). Implement a parameterized constructor to initialize 3 objects. Include an object method to display the student details and a class method to display the total number of students.

Solution:

1.

Problem Analysis:

The purpose of this program is to define an Employee class that showcases the use of object variables, class variables, constructors, object methods, and class methods. Employee details such as name, ID, and salary are initialized using a parameterized constructor. An object method is used

to display each employee's information. A class variable keeps track of how many employee objects have been created, and a class method is used to return and display the total count of employees.

Background Theory:

- **Object Variables:** These are unique to each object (e.g., name, age, salary) and store individual employee information.
- **Class Variables (static):** Shared among all objects of the class. For instance, companyName or employeeCount remains common for all employees.
- **Parameterized Constructor:** A special method used to initialize object variables with specific values when an object is created.
- **Object Methods:** These are regular methods that work with object-specific data, such as displaying an employee's details.
- **Class Methods (static):** These operate on static data (like a total employee count) and can be called without creating an object.
- **main() Method:** Acts as the entry point of the program, where objects are created, methods are called, and execution begins.

Algorithm Design:

1. Create an Employee class.
2. Declare class variables such as companyName, companyAddress, and count.
3. Declare object variables such as name, age, designation, and salary.
4. Implement a parameterized constructor to set the employee properties and increment count.
5. Define an object method, such as displayDetails(), to display the employee properties.
6. Implement a class method, such as count(), to return the total number of employees.
7. In the main() function, assign values to class variables.
8. Create employee objects using the constructor and set their properties.
9. Use object methods to display the employee details.
10. Use the class method to display the total number of employees.

Code:

```
public class Employee {  
  
    public static String companyName = "Infotech PLC";
```

```

public static String companyAddress = "Uttara, Dhaka";
public static int totalEmployee = 0;

private String name;
private String designation;
private int salary;
private int age;

public Employee(String name,String designation,int salary){
    this.name = name;
    this.designation = designation;
    this.salary = salary;
    totalEmployee++;
}

public void displayEmployeeDetails(){
    System.out.println("Name: "+name);
    System.out.println("Designation: "+designation);
    System.out.println("Salary: "+salary);
    System.out.println("Compnay Name: "+companyName);
    System.out.println("Company Address: "+companyAddress);
}

public static int getTotalEmployee() {
    return totalEmployee;
}

public static void main(String[] args) {

    Employee employee1 = new Employee("Mahmud","Manager",10000);
    Employee employee2 = new Employee("Habib","Businesss",50000);

    System.out.println("Employee 1 details:");
    employee1.displayEmployeeDetails();
    System.out.println("");
    System.out.println("Employee 2 details:");
    employee2.displayEmployeeDetails();
    System.out.println("");

    System.out.println("Total number of Employee: "+Employee.getTotalEmployee());
}
}

```

Output:

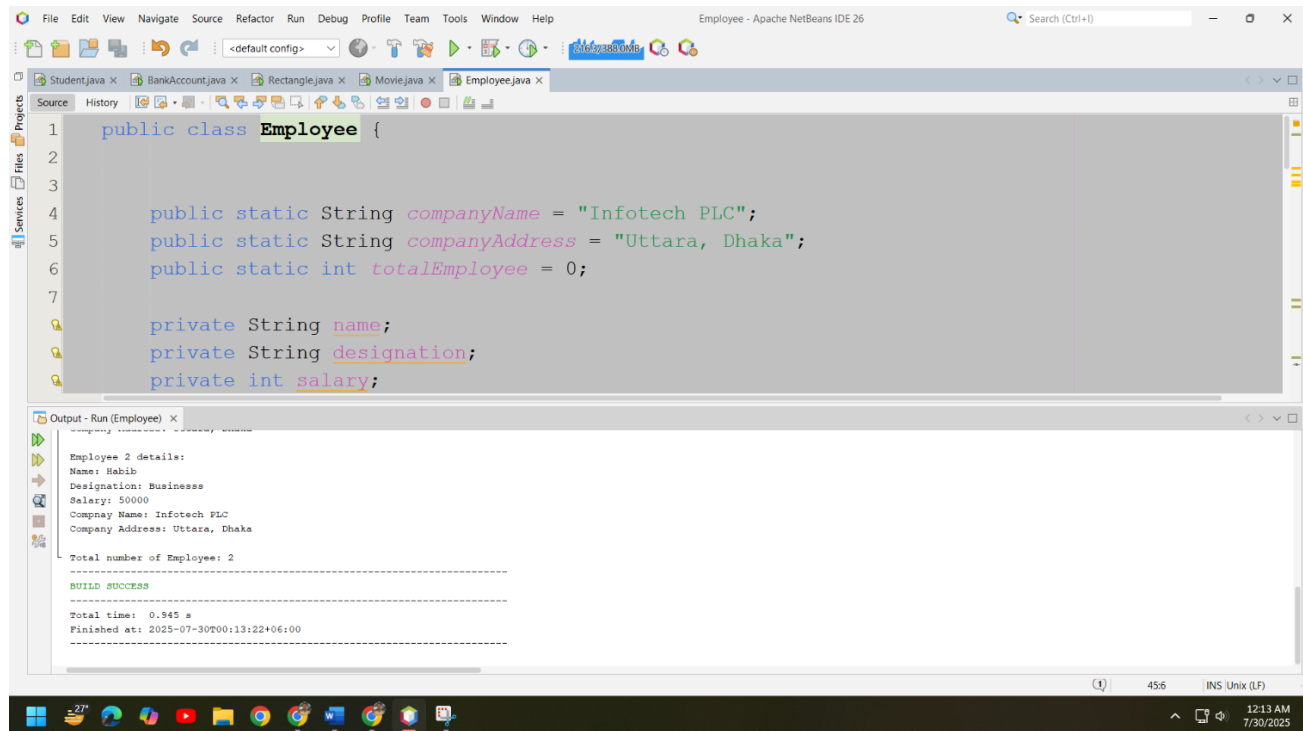


Figure 1: Output

2.

Problem Analysis:

The goal of this program is to define a Book class that illustrates the use of object-oriented programming concepts such as class variables, object variables, constructors, object methods, and class methods. Each Book instance will contain its own specific details—such as title, author, and publication year—stored as object variables. A common genre, shared by all books, will be implemented as a static class variable. The book properties will be initialized using a parameterized constructor. An object method will be used to display individual book information, while a static class method will be responsible for showing the total number of book objects created.

Background Theory:

- **Object Variables:** These hold data that's different for each object, such as title, author, or year in a book — each object gets its own values.

- **Class Variables (static):** These are shared by all objects of the class. For example, a genre that is the same for every book would be declared as a static variable.
- **Parameterized Constructor:** A constructor with parameters that sets the initial values of object variables when a new object is created.
- **Object Methods:** Functions that use and display data from individual objects, like showing details of a specific book.
- **Class Methods (static):** These work with static data and can be accessed using the class name—like showing how many books have been created.
- **main() Method:** This is the entry point of the program—where object creation and method calls begin.

Algorithm Design:

1. Create a Book class.
2. Declare a class variable: genre.
3. Declare object variables: title, author, year.
4. Declare a static variable count to track the total number of books.
5. Implement a parameterized constructor to initialize the object variables and increment count.
6. Define an object method, e.g., displayDetails(), to show book information.
7. Implement a class method, e.g., getTotalBooks(), to return the total number of books.
8. In the main() method, set the value of the class variable genre.
9. Create 3 book objects using the constructor.
10. Use object methods to display details of each book.
11. Use the class method to display the total number of books created. Use the display() method to output the details of each rectangle.

Code:

```
public class Book {

    static String genre;
    public static int totalBook = 0;

    private String title;
    private String author;
    private int year;
```

```

public Book(String title,String author,int year){
    this.title = title;
    this.author = author;
    this.year = year;
    totalBook++;
}

public void displayBookDetails(){
    System.out.println("Title: "+title);
    System.out.println("author: "+author);
    System.out.println("Year: "+year);

}

public static int getTotalBook() {
    return totalBook;

}

public static void main(String[] args) {

    Book.genre = "Information_System";
    Book Book1 = new Book("Introduction to Java","Robert Bruce",2005);
    Book Book2 = new Book("Introduction to Networking","Josheph Howard",2003);

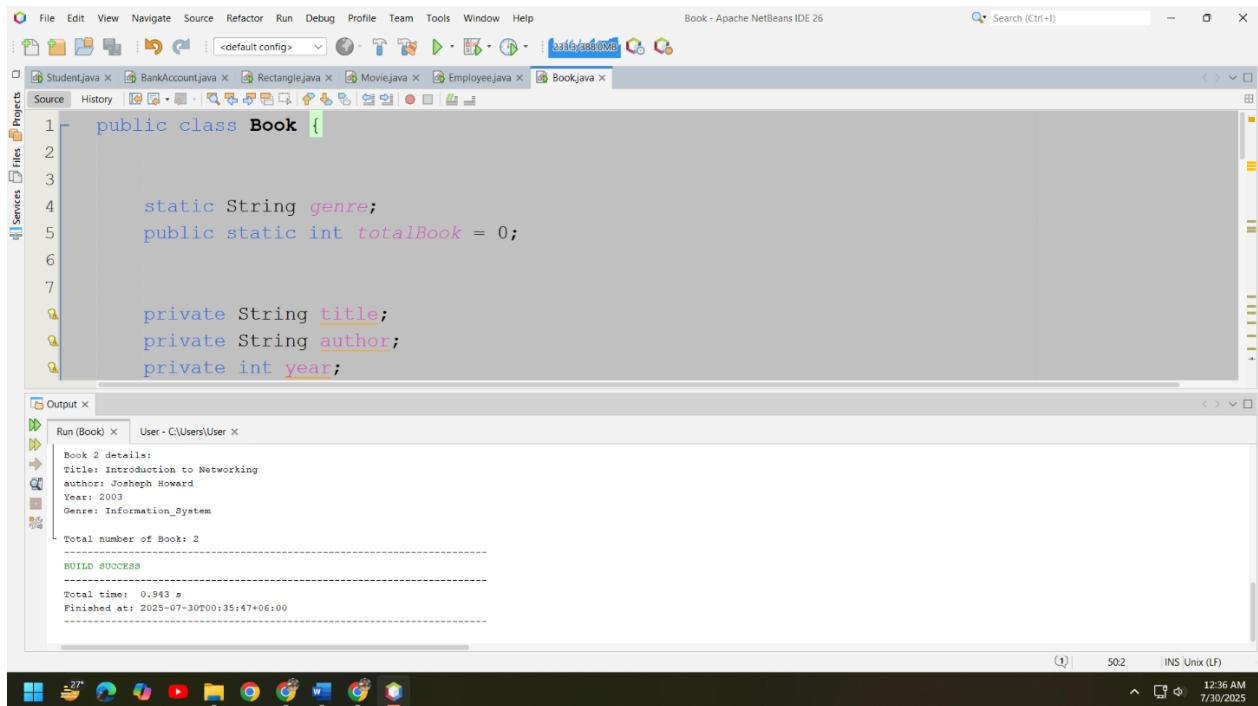
    System.out.println("Book 1 details:");
    Book1.displayBookDetails();
    System.out.println("Genre: "+genre);
    System.out.println("");

    System.out.println("Book 2 details:");
    Book2.displayBookDetails();
    System.out.println("Genre: "+genre);
    System.out.println("");

    System.out.println("Total number of Book: "+Book.getTotalBook());
}
}

```

Output:



The screenshot displays the Apache NetBeans IDE interface. The main editor window shows the source code for a Java class named `Book`. The code includes static variables `genre` and `totalBook`, and private instance variables `title`, `author`, and `year`. Below the editor, the 'Output' window is open, showing the results of running the program. It displays the details of two books and the total count.

```
1 public class Book {
2
3
4     static String genre;
5     public static int totalBook = 0;
6
7     private String title;
8     private String author;
9     private int year;
```

Run (Book) x User - C:\Users\User x

Book 2 details:
Title: Introduction to Networking
author: Joseph Howard
Year: 2003
Genre: Information_System

Total number of Book: 2

BUILD SUCCESS

Total time: 0.943 s
Finished at: 2025-07-30T00:35:47+06:00

Figure 2: Output

3.

Problem Analysis:

This program defines a Student class that showcases the difference between instance-level and class-level data. Each Student object contains personal attributes such as id, name, department, and cgpa, represented as object variables. The name of the university is shared across all students and is therefore declared as a static class variable. A parameterized constructor is used to assign values to each student upon creation. An instance method is responsible for displaying the details of an individual student, while a static class method keeps track of and displays the total number of student objects created.

Background Theory:

- **Object Variables:** These are unique for every object (like name, age, etc.)
- **Class Variables (static):** Shared across all objects - for example, the company name should be the same for all employees.

- **Parameterized Constructor:** A special method that helps us initialize values when we create an object.
- **Object Methods:** These use object data (like showing details of a specific employee).
- **Class Methods (static):** These work with static variables (like counting all employees), and can be called without creating an object.
- The main() method is where everything starts running.

Algorithm Design:

1. Define a class named Student.
2. Inside the class, declare a static variable university to represent the shared university name.
3. Add instance variables: id, name, department, and cgpa to store individual student data.
4. Declare a static counter variable count to keep track of how many student objects are created.
5. Write a parameterized constructor that sets the values of the object variables and increases the student count.
6. Implement an instance method displayDetails() to print each student's information.
7. Add a static method getTotalStudents() to return the current value of the student counter.
8. In the main() method:
 - a) Set the value of the class-level university variable.
 - b) Instantiate 3 Student objects using the constructor.
 - c) Call the displayDetails() method for each object to print their info.
 - d) Call getTotalStudents() to show the total number of students created.

Code:

```
public class Student{
    int id;
    String name;
    String department;
    double cgpa;

    static String university = "Southeast University";
    static int count = 0;

    Student(int id,String name, String department, double cgpa){
        this.id = id;
        this.name = name;
        this.department = department;
    }
}
```

```

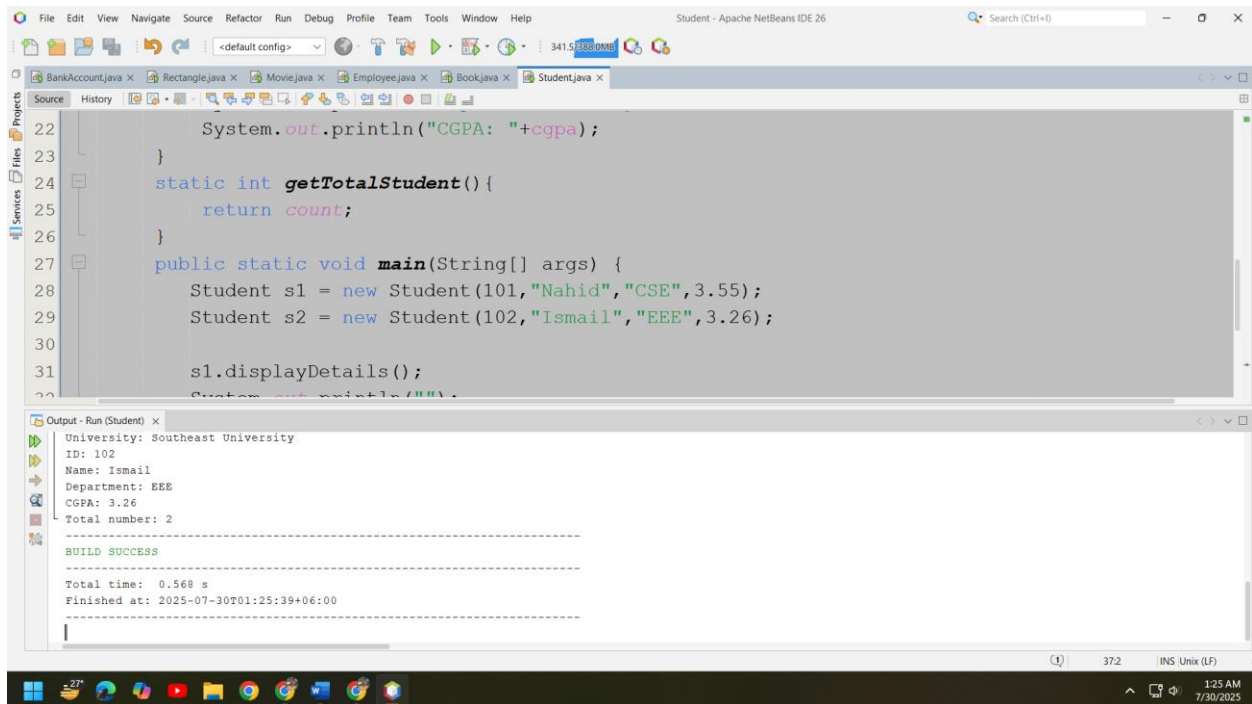
        this.cgpa = cgpa;
        count++;
    }
    void displayDetails(){
        System.out.println("University: "+university);
        System.out.println("ID: "+id);
        System.out.println("Name: "+name);
        System.out.println("Department: "+department);
        System.out.println("CGPA: "+cgpa);
    }
    static int getTotalStudent(){
        return count;
    }
    public static void main(String[] args) {
        Student s1 = new Student(101,"Nahid","CSE",3.55);
        Student s2 = new Student(102,"Ismail","EEE",3.26);

        s1.displayDetails();
        System.out.println("");
        s2.displayDetails();

        System.out.println("Total number: "+Student.getTotalStudent());
    }
}

```

Output:



The screenshot displays the Apache NetBeans IDE interface. The top pane shows the source code of a Java program. The bottom pane shows the output of the program's execution.

```
22     System.out.println("CGPA: "+cgpa);
23 }
24 static int getTotalStudent() {
25     return count;
26 }
27 public static void main(String[] args) {
28     Student s1 = new Student(101, "Nahid", "CSE", 3.55);
29     Student s2 = new Student(102, "Ismail", "EEE", 3.26);
30
31     s1.displayDetails();
32     System.out.println("====");
33 }
```

Output - Run (Student) x

```
University: Southeast University
ID: 102
Name: Ismail
Department: EEE
CGPA: 3.26
Total number: 2
=====
BUILD SUCCESS
=====
Total time: 0.568 s
Finished at: 2025-07-30T01:25:39+06:00
=====
```

Figure 3: Output