



**BUBT** | BANGLADESH UNIVERSITY OF  
BUSINESS AND TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

LAB REPORT  
INTRODUCTION TO JAVA

---

# Advanced Programming Language (CSE-342)

---

*Submitted By:*  
Kaniz Fatema  
20245103154  
Intake 53 (01)

*Submitted To:*  
Most. Jannatul Ferdous  
Assistant Professor  
Dept. of CSE

January 21, 2026

# Contents

<b>1</b>	<b>Create a bank account system</b>	<b>3</b>
1.0.1	Question . . . . .	3
1.0.2	Solution . . . . .	4
<b>2</b>	<b>Represent a Book in a library</b>	<b>6</b>
2.0.1	Question . . . . .	6
2.0.2	Solution . . . . .	6
<b>3</b>	<b>A Metro Ticket booking system</b>	<b>7</b>
3.0.1	Question . . . . .	7
3.0.2	Solution . . . . .	7
<b>4</b>	<b>Array of Student objects</b>	<b>11</b>
4.0.1	Question . . . . .	11
4.0.2	Solution . . . . .	11

# Scenario

You are the system administrator of a Linux server used by a small development team. Recently, some team members reported some issues and you need to troubleshoot and perform user management tasks.

## 1 Create a bank account system

### 1.0.1 Question

Consider a simple bank account system where you have customers and their bank accounts. Each customer has a name, a unique identifier, and can have one or more bank accounts. Each bank account has an account number, a balance, and belongs to a specific customer. Design a set of Java classes to model this bank account system. Your implementation should include the following features.

### Customer Class

- **Attributes:** name, customerID, bankAccount
- **Methods:** a method to add a bank account, a method to display customer information, methods to deposit and withdraw money

### BankAccount Class

- **Attributes:** accountNumber, balance
- **Methods:** a method to deposit money, a method to withdraw money, a method to display account information

## 1.0.2 Solution

---

```
1 class BankAccount {
2     int accountNumber;
3     double balance;
4
5     void set(int accountNumber, double balance) {
6         this.accountNumber = accountNumber;
7         this.balance = balance;
8     }
9
10    void deposit(double amount) {
11        balance = balance + amount;
12    }
13
14    void withdraw(double amount) {
15        if (amount < 500) {
16            balance = balance - amount;
17        } else {
18            System.out.println("Transaction Not valid");
19        }
20    }
21
22    void displayInfo() {
23        System.out.println("Account Number: " + accountNumber + " Balance: " +
24            ↪ balance);
25    }
26 }
27 class Customer {
28     String name;
29     int customerID;
30     BankAccount account;
31
32     void set(String name, int customerID, BankAccount account) {
33         this.name = name;
34         this.customerID = customerID;
35         this.account = account;
36     }
37
38     void deposit(double amount) {
39         account.deposit(amount);
40     }
41
42     void withdraw(double amount) {
43         account.withdraw(amount);
44     }
45
46     void displayInfo() {
47         System.out.println("Name: " + name + " Customer Id: " + customerID );
```

```
48         account.displayInfo();
49     }
50 }
51
52 public class task1 {
53     public static void main(String args[]) {
54
55         BankAccount B1 = new BankAccount();
56         B1.set(12, 500.00);
57
58         Customer C1 = new Customer();
59         C1.set("Kaniz", 154, B1);
60
61         C1.deposit(300);
62         C1.withdraw(200);
63
64         C1.displayInfo();
65     }
66 }
```

---

Listing 1: Task 1: A bank account system

## Output

---

```
1 Name: Kaniz Customer Id: 154
2 Account Number: 12 Balance: 600.0
```

---

## 2 Represent a Book in a library

### 2.0.1 Question

Write a Java program to create a class named **Book** that represents basic information about a book in a library. The class should have three attributes: **title** (String), **author** (String), and **yearPublished** (int). Include a constructor to initialize these fields and a method called **displayInfo()** that prints the book's details. In the main method, create at least two objects of the Book class with different data and call the **displayInfo()** method for each to display their information.

### 2.0.2 Solution

---

```
1 class Book {
2     String title, author;
3     int yearPublished;
4     Book()
5     {
6         this.title = "Java: The Complete Reference";
7         this.author = "Herbert Schildt";
8         this.yearPublished = 2018;
9     }
10    Book(String title, String author, int yearPublished)
11    {
12        this.title = title;
13        this.author = author;
14        this.yearPublished = yearPublished;
15    }
16
17    void displayInfo()
18    {
19        System.out.println("Title: " + title + " Author: " + author + " Year Published:
20        ↪ " + yearPublished);
21    }
22 }
23 public class task2 {
24     public static void main (String args[])
25     {
26         Book B1 = new Book();
27         Book B2 = new Book("Atomic Habits", "James Clear", 2018);
28
29         B1.displayInfo();
30         B2.displayInfo();
31     }
32 }
```

---

Listing 2: Task 2: Represent a Book in a library

# Output

---

1 Title: Java: The Complete Reference Author: Herbert Schildt Year Published: 2018  
2 Title: Atomic Habits Author: James Clear Year Published: 2018

---

## 3 A Metro Ticket booking system

### 3.0.1 Question

3.Design a **MetroTicket** class to simulate a metro ticket booking system using **constructor overloading**. Passengers can book tickets with different levels of information: some may provide only their name, source, and destination; others may include whether it's a round trip, choose between "Regular" or "Premium" seat classes, and even apply discount codes. The class should have four overloaded constructors to handle these scenarios. Use constructor chaining to avoid code repetition. Include validations—for instance, seat class must be either "Regular" or "Premium", and if an invalid value is given, default to "Regular". Implement methods like **calculateFare()** to compute the fare based on trip type and seat class (e.g., one-way Regular = 100, Premium = 150; round-trip Regular = 180, Premium = 270), with a 10% discount if the code "DIS10" is applied. Also, write a **displayTicketDetails()** method to show all ticket information. Create at least four objects using different constructors to demonstrate constructor overloading effectively.

### 3.0.2 Solution

---

```
1 class MetroTicket {
2     String name;
3     String source;
4     String destination;
5     boolean roundTrip;
6     String seatClass;
7     String discountCode;
8
9     //constructor 1: name, source, destination
10    MetroTicket(String name, String source, String destination) {
11        this.name = name;
12        this.source = source;
13        this.destination = destination;
14        this.roundTrip = false;
15        this.seatClass = "Regular";
16        this.discountCode = "";
17    }
18
19    //constructor 2: name, source, destination, roundTrip
20    MetroTicket(String name, String source, String destination, boolean roundTrip) {
21        this(name, source, destination); // call constructor 1
22        this.roundTrip = roundTrip;
23    }
24}
```

```

25 //constructor 3: name, source, destination, roundTrip, seatClass
26 MetroTicket(String name, String source, String destination, boolean roundTrip,
    ↪ String seatClass) {
27     this(name, source, destination, roundTrip); // call constructor 2
28     if (seatClass != null && seatClass.equalsIgnoreCase("Premium")) {
29         this.seatClass = "Premium";
30     } else {
31         this.seatClass = "Regular";
32     }
33 }
34
35 //constructor 4: name, source, destination, roundTrip, seatClass, discountCode
36 MetroTicket(String name, String source, String destination, boolean roundTrip,
    ↪ String seatClass, String discountCode) {
37     this(name, source, destination, roundTrip, seatClass); // call constructor 3
38     if (discountCode != null) {
39         this.discountCode = discountCode;
40     } else {
41         this.discountCode = "";
42     }
43 }
44
45 //constructor 5: name, source, destination, seatClass, discountCode (no roundTrip)
46 MetroTicket(String name, String source, String destination, String seatClass,
    ↪ String discountCode) {
47     this(name, source, destination); // call constructor 1
48     if (seatClass != null && seatClass.equalsIgnoreCase("Premium")) {
49         this.seatClass = "Premium";
50     } else {
51         this.seatClass = "Regular";
52     }
53     if (discountCode != null) {
54         this.discountCode = discountCode;
55     } else {
56         this.discountCode = "";
57     }
58     this.roundTrip = false;
59 }
60
61 // Calculate fare
62 double calculateFare() {
63     double fare;
64     if (this.roundTrip) {
65         if (this.seatClass.equals("Premium")) {
66             fare = 270;
67         } else {
68             fare = 180;
69         }
70     } else {
71         if (this.seatClass.equals("Premium")) {

```

```

72         fare = 150;
73     } else {
74         fare = 100;
75     }
76 }
77
78 if ("DIS10".equalsIgnoreCase(this.discountCode)) {
79     fare = fare * 0.9; // 10% discount
80 }
81
82 return fare;
83 }
84
85 // Display ticket details
86 void displayTicketDetails() {
87     System.out.println("Passenger Name: " + this.name);
88     System.out.println("Source: " + this.source);
89     System.out.println("Destination: " + this.destination);
90     System.out.println("Trip Type: " + (this.roundTrip ? "Round Trip" : "One
    ↳ Way"));
91     System.out.println("Seat Class: " + this.seatClass);
92     System.out.println("Discount Code: " + (this.discountCode.isEmpty() ? "None" :
    ↳ this.discountCode));
93     System.out.println("Fare: " + this.calculateFare());
94     System.out.println("-----");
95 }
96 }
97
98 public class task3 {
99     public static void main(String[] args) {
100
101         MetroTicket t1 = new MetroTicket("Kaniz", "Station A", "Station B");
102         ↳ //constructor 1
103         MetroTicket t2 = new MetroTicket("Fatema", "Station C", "Station D", true);
104         ↳ //constructor 2
105         MetroTicket t3 = new MetroTicket("Spider Man", "Station G", "Station H", true,
106         ↳ "Premium", "DIS10"); //constructor 4
107         MetroTicket t4 = new MetroTicket("Mr. Bean", "Station E", "Station F", true,
108         ↳ "Premium"); //constructor 3
109         MetroTicket t5 = new MetroTicket("Hiro", "Station I", "Station J", "Premium",
110         ↳ "DIS10"); //constructor 5
111
112         t1.displayTicketDetails();
113         t2.displayTicketDetails();
114         t3.displayTicketDetails();
115         t4.displayTicketDetails();
116         t5.displayTicketDetails();
117     }
118 }

```

## Output

---

```
1 Passenger Name: Kaniz
2 Source: Station A
3 Destination: Station B
4 Trip Type: One Way
5 Seat Class: Regular
6 Discount Code: None
7 Fare: 100.0
8 -----
9 Passenger Name: Fatema
10 Source: Station C
11 Destination: Station D
12 Trip Type: Round Trip
13 Seat Class: Regular
14 Discount Code: None
15 Fare: 180.0
16 -----
17 Passenger Name: Spider Man
18 Source: Station G
19 Destination: Station H
20 Trip Type: Round Trip
21 Seat Class: Premium
22 Discount Code: DIS10
23 Fare: 243.0
24 -----
25 Passenger Name: Mr. Bean
26 Source: Station E
27 Destination: Station F
28 Trip Type: Round Trip
29 Seat Class: Premium
30 Discount Code: None
31 Fare: 270.0
32 -----
33 Passenger Name: Hiro
34 Source: Station I
35 Destination: Station J
36 Trip Type: One Way
37 Seat Class: Premium
38 Discount Code: DIS10
39 Fare: 135.0
40 -----
```

---

## 4 Array of Student objects

### 4.0.1 Question

A department registers students for a course using a fixed-size system. Each student has a student ID, name, and CGPA. The administrator can:

- Register a new student
- Search a student by ID
- Display all registered students

Implement the system using **an array of Student objects**.

### 4.0.2 Solution

---

```
1 import java.util.Scanner;
2
3 class Student {
4     int id;
5     String name;
6     double cgpa;
7
8     Student(int id, String name, double cgpa) {
9         this.id = id;
10        this.name = name;
11        this.cgpa = cgpa;
12    }
13
14    void display() {
15        System.out.println("ID: " + id + ", Name: " + name + ", CGPA: " + cgpa);
16    }
17 }
18
19 public class task4 {
20     public static void main(String[] args) {
21         Scanner sc = new Scanner(System.in);
22
23         System.out.print("Enter number of students: ");
24         int n = sc.nextInt();
25         sc.nextLine();
26
27         Student[] students = new Student[n];
28         int count = 0;
29
30         while (true) {
31             System.out.println("\n1.Register  2.Search  3.Display  4.Exit");
32             int x = sc.nextInt();
33             sc.nextLine();
34
```

```

35 // Register one student
36 if (x == 1) {
37     if (count == n) {
38         System.out.println("System full!");
39         continue;
40     }
41
42     System.out.print("Enter student ID: ");
43     int id = sc.nextInt();
44     sc.nextLine();
45
46     System.out.print("Enter student name: ");
47     String name = sc.nextLine();
48
49     System.out.print("Enter student CGPA: ");
50     double cgpa = sc.nextDouble();
51     sc.nextLine();
52
53     students[count] = new Student(id, name, cgpa);
54     count++;
55
56     System.out.println("Student registered successfully!");
57
58 // Search by ID
59 } else if (x == 2) {
60     System.out.print("Enter ID to search: ");
61     int id = sc.nextInt();
62     sc.nextLine();
63
64     int idx = -1;
65     for (int i = 0; i < count; i++) {
66         if (students[i].id == id) {
67             idx = i;
68             break;
69         }
70     }
71
72     if (idx == -1) System.out.println("Not found");
73     else students[idx].display();
74
75 // Display all registered students
76 } else if (x == 3) {
77     if (count == 0) {
78         System.out.println("No students registered yet.");
79         continue;
80     }
81
82     System.out.println("\nAll registered students:");
83     for (int i = 0; i < count; i++) {
84         students[i].display();

```

```

85         }
86
87         // Exit
88     } else if (x == 4) {
89         break;
90
91     } else {
92         System.out.println("Invalid choice!");
93     }
94 }
95
96 sc.close();
97 }
98 }

```

---

Listing 4: Task 4: Array of Student object

## Output

---

```

1 Enter number of students: 4
2
3 1.Register  2.Search  3.Display  4.Exit
4 1
5 Enter student ID: 154
6 Enter student name: Kaniz
7 Enter student CGPA: 4.00
8 Student registered successfully!
9
10 1.Register  2.Search  3.Display  4.Exit
11 2
12 Enter ID to search: 154
13 ID: 154, Name: Kaniz, CGPA: 4.0
14
15 1.Register  2.Search  3.Display  4.Exit
16 3
17
18 All registered students:
19 ID: 154, Name: Kaniz, CGPA: 4.0
20
21 1.Register  2.Search  3.Display  4.Exit
22 4

```

---

1 2

---

<sup>1</sup>Kaniz Fatema

<sup>2</sup>ID: 20245103154