

Q.1 Java program demonstrating class declaration and object initialization:

Code:

```
// Class declaration
class Car {

    // Instance variables (attributes)
    String make;
    String model;
    int year;

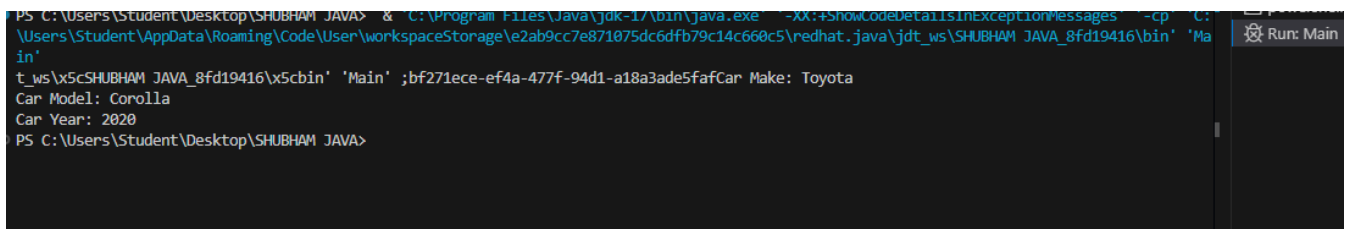
    // Constructor to initialize the object
    Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    // Method to display the car details
    void displayDetails() {
        System.out.println("Car Make: " + make);
        System.out.println("Car Model: " + model);
        System.out.println("Car Year: " + year);
    }
}
```

```
}
```

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // Object initialization using the constructor  
  
        Car myCar = new Car("Toyota", "Corolla", 2020);  
  
        // Calling the method to display details of the car  
  
        myCar.displayDetails();  
  
    }  
}
```

Output:



```
PS C:\Users\Student\Desktop\SHUBHAM JAVA> & "C:\Program Files\Java\jdk-17\bin\java.exe" -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e871075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin' 'Main'  
in'  
t_ws\5cSHUBHAM JAVA_8fd19416\5cbin' 'Main' ;bf271ece-ef4a-477f-94d1-a18a3ade5fafCar Make: Toyota  
Car Model: Corolla  
Car Year: 2020  
PS C:\Users\Student\Desktop\SHUBHAM JAVA>
```

Q.2 prog to demonstartae concept of encapsulation using getters and setters

code:

```
// Main.java
```

```
public class Main {  
  
    // Inner class Person  
    static class Person {  
        // Private instance variables  
        private String name;  
        private int age;  
  
        // Getter method for 'name'  
        public String getName() {  
            return name;  
        }  
  
        // Setter method for 'name'  
        public void setName(String name) {  
            this.name = name;  
        }  
  
        // Getter method for 'age'  
        public int getAge() {  
            return age;  
        }  
    }  
}
```

```
// Setter method for 'age'  
  
public void setAge(int age) {  
  
    // Ensure the age is positive before setting it  
  
    if (age > 0) {  
  
        this.age = age;  
  
    } else {  
  
        System.out.println("Age must be positive.");  
  
    }  
  
}  
  
}
```

```
public static void main(String[] args) {  
  
    // Create an object of the Person class  
  
    Person person = new Person();  
  
  
  
  
    // Using setter methods to set values  
  
    person.setName("Shubham Asawale");  
  
    person.setAge(25);  
  
  
  
  
    // Using getter methods to get values  
  
    System.out.println("Person's Name: " + person.getName());  
  
    System.out.println("Person's Age: " + person.getAge());  
  
}
```

```

// Trying to set an invalid age
person.setAge(-5); // This will print: Age must be positive.

// Display the name and age after the invalid attempt to change age
System.out.println("Person's Name: " + person.getName());
System.out.println("Person's Age: " + person.getAge());
}
}

```

Output:

```

PS C:\Users\Student\Desktop\SHUBHAM JAVA> c.; cd
'c:\Users\Student\Desktop\SHUBHAM JAVA'; & 'C:\Program Files\Java\jdk-
17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e87
1075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin'
'Main'

```

Person's Name: Shubham Asawale

Person's Age: 25

Age must be positive.

Person's Name: Shubham Asawale

Person's Age: 25

```
PS C:\Users\Student\Desktop\SHUBHAM JAVA>
```

Q3.prog to demonstrate of different types of constructor in java

// Class with different types of constructors

class Car {

// Instance variables

String make;

String model;

int year;

// Default constructor (no parameters)

public Car() {

this.make = "Unknown";

this.model = "Unknown";

this.year = 0;

System.out.println("Default constructor called!");

}

// Parameterized constructor (with parameters)

public Car(String make, String model, int year) {

this.make = make;

this.model = model;

this.year = year;

System.out.println("Parameterized constructor called!");

}

// Constructor to demonstrate constructor overloading

```
public Car(String make) {  
    this.make = make;  
    this.model = "Unknown";  
    this.year = 0;  
    System.out.println("Constructor with one parameter called!");  
}
```

// Method to display car details

```
public void displayDetails() {  
    System.out.println("Car Make: " + make);  
    System.out.println("Car Model: " + model);  
    System.out.println("Car Year: " + year);  
}  
}
```

public class Main {

public static void main(String[] args) {

// Creating objects using different constructors

// Default constructor

Car car1 = new Car();

car1.displayDetails();

```

// Parameterized constructor
Car car2 = new Car("Toyota", "Corolla", 2020);
car2.displayDetails();

// Constructor with one parameter
Car car3 = new Car("Honda");
car3.displayDetails();
}
}

```

Output:

```

PS C:\Users\Student\Desktop\SHUBHAM JAVA> c.; cd
'c:\Users\Student\Desktop\SHUBHAM JAVA'; & 'C:\Program Files\Java\jdk-
17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e87
1075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin'
'Main'

```

Default constructor called!

Car Make: Unknown

Car Model: Unknown

Car Year: 0

Parameterized constructor called!

Car Make: Toyota

Car Model: Corolla

Car Year: 2020

Constructor with one parameter called!

Car Make: Honda

Car Model: Unknown

Car Year: 0

PS C:\Users\Student\Desktop\SHUBHAM JAVA>

Q.4 Java Program to Demonstrate Static Variables, Static Methods, and Static Blocks

Code :

```
public class Example {
```

```
// Static variable (shared among all instances of the class)
```

```
static int count = 0;
```

```
// Instance variable
```

```
int instanceVar;
```

// Static block (called when the class is loaded)

static {

System.out.println("Static block executed.");

count = 10; // Initializing static variable in static block

}

// Constructor

public Example(int instanceVar) {

this.instanceVar = instanceVar;

count++; // Increment static variable count for every object created

}

// Static method

public static void displayCount() {

System.out.println("Count (Static Variable): " + count);

}

// Instance method

public void displayInstanceVar() {

System.out.println("Instance Variable: " + instanceVar);

}

public static void main(String[] args) {

// Static method can be called without creating an object

Example.displayCount(); // Output: Count (Static Variable): 10

// Creating objects of the Example class

Example obj1 = new Example(100);

Example obj2 = new Example(200);

// Displaying instance variables and static variables

obj1.displayInstanceVar(); // Output: Instance Variable: 100

obj2.displayInstanceVar(); // Output: Instance Variable: 200

// Static variable is shared among all instances

Example.displayCount(); // Output: Count (Static Variable): 12

}

}

Output:

```
PS C:\Users\Student\Desktop\SHUBHAM JAVA> c::; cd
'c:\Users\Student\Desktop\SHUBHAM JAVA'; & 'C:\Program Files\Java\jdk-
17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e87
1075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin'
'Example'
```

ceStorage\%ce2ab9cc7e871075dc6dfb79c14c660c5\credhat.java\cjd_t_ws\%cSHUBHAM JAVA_8fd19416\%cbin' 'Example';645f4900-007e-4588-a2a4-905704a10020Static block executed.

Count (Static Variable): 10

Instance Variable: 100

Instance Variable: 200

Count (Static Variable): 12

PS C:\Users\Student\Desktop\SHUBHAM JAVA>

Q.5 Design a Java class that represents a library book, applying the principles of encapsulation, constructors, and static members. Include methods to borrow and return books with relevant logic.

Code:

```
public class LibraryBook {
```

```
    // Private instance variables (Encapsulation)
```

```
    private String title;
```

```
    private String author;
```

```
    private boolean isBorrowed;
```

```
    // Static variable to keep track of total borrowed books
```

```
    private static int totalBorrowedBooks = 0;
```

// Default constructor

```
public LibraryBook() {  
    this.title = "Unknown";  
    this.author = "Unknown";  
    this.isBorrowed = false;  
}
```

// Parameterized constructor

```
public LibraryBook(String title, String author) {  
    this.title = title;  
    this.author = author;  
    this.isBorrowed = false;  
}
```

// Getter and Setter methods (Encapsulation)

```
public String getTitle() {  
    return title;  
}
```

```
public void setTitle(String title) {  
    this.title = title;  
}
```

```
public String getAuthor() {  
    return author;  
}
```

```
public void setAuthor(String author) {  
    this.author = author;  
}
```

```
public boolean isBorrowed() {  
    return isBorrowed;  
}
```

```
// Static method to get total borrowed books  
public static int getTotalBorrowedBooks() {  
    return totalBorrowedBooks;  
}
```

```
// Method to borrow the book  
public void borrowBook() {  
    if (!isBorrowed) {  
        isBorrowed = true;  
        totalBorrowedBooks++;  
    }  
}
```

```
        System.out.println("Book borrowed: " + title);
    } else {
        System.out.println("Book is already borrowed: " + title);
    }
}
```

// Method to return the book

```
public void returnBook() {
    if (isBorrowed) {
        isBorrowed = false;
        totalBorrowedBooks--;
        System.out.println("Book returned: " + title);
    } else {
        System.out.println("Book was not borrowed: " + title);
    }
}
```

// Method to display book info

```
public void displayInfo() {
    System.out.println("Title: " + title + " | Author: " + author + " | Borrowed: " +
isBorrowed);
}
```

```
// Main method to test the LibraryBook class

public static void main(String[] args) {

    LibraryBook book1 = new LibraryBook("The Hobbit", "J.R.R. Tolkien");

    LibraryBook book2 = new LibraryBook("1984", "George Orwell");


    book1.displayInfo();

    book2.displayInfo();


    book1.borrowBook();

    book1.borrowBook(); // Should show already borrowed


    book2.borrowBook();


    System.out.println("Total borrowed books: " +
LibraryBook.getTotalBorrowedBooks());


    book1.returnBook();

    book1.returnBook(); // Should show not borrowed


    System.out.println("Total borrowed books: " +
LibraryBook.getTotalBorrowedBooks());

}

}
```

Output:

```
PS C:\Users\Student\Desktop\SHUBHAM JAVA> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e871075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin' 'LibraryBook'
```

Title: The Hobbit | Author: J.R.R. Tolkien | Borrowed: false

Title: 1984 | Author: George Orwell | Borrowed: false

Book borrowed: The Hobbit

Book is already borrowed: The Hobbit

Book borrowed: 1984

Total borrowed books: 2

Book returned: The Hobbit

Book was not borrowed: The Hobbit

Total borrowed books: 1

```
PS C:\Users\Student\Desktop\SHUBHAM JAVA>
```

Q.6 Design a class hierarchy using single inheritance for a simple organizational structure (e.g., Employee → Manager). Include relevant attributes and methods for each class.

Code:

// Base class

class Employee {

protected String name;

protected int id;

protected double salary;

// Constructor

public Employee(String name, int id, double salary) {

this.name = name;

this.id = id;

this.salary = salary;

}

// Method to display employee details

public void displayDetails() {

System.out.println("Employee ID : " + id);

System.out.println("Name : " + name);

System.out.println("Salary : \$" + salary);

}

}

// Derived class using single inheritance

class Manager extends Employee {

```
private String department;
```

```
private int teamSize;
```

```
// Constructor
```

```
public Manager(String name, int id, double salary, String department, int  
teamSize) {
```

```
    super(name, id, salary); // Call parent class constructor
```

```
    this.department = department;
```

```
    this.teamSize = teamSize;
```

```
}
```

```
// Additional method to display manager-specific details
```

```
public void displayManagerDetails() {
```

```
    displayDetails(); // Call base class method
```

```
    System.out.println("Department   : " + department);
```

```
    System.out.println("Team Size    : " + teamSize);
```

```
}
```

```
}
```

```
// Main class to test the hierarchy
```

```
public class EmployeeManagerDemo {
```

```
    public static void main(String[] args) {
```

```
        // Create an Employee object
```

```

Employee emp = new Employee("SHUBHAM", 101, 500000);

System.out.println("=== Employee Details ===");

emp.displayDetails();


System.out.println();


// Create a Manager object
Manager mgr = new Manager("shubham", 201, 80000, "IT", 5);

System.out.println("=== Manager Details ===");

mgr.displayManagerDetails();

}

}

```

Output:

```

PS C:\Users\Student\Desktop\SHUBHAM JAVA> c:: cd
'c:\Users\Student\Desktop\SHUBHAM JAVA'; & 'C:\Program Files\Java\jdk-
17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e87
1075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin'
'EmployeeManagerDemo'

ceStorage\x5ce2ab9cc7e871075dc6dfb79c14c660c5\x5credhat.java\x5cjdt_ws\x5
cSHUBHAM JAVA_8fd19416\x5cbin' 'EmployeeManagerDemo' ;a9751ee9-86d4-
4eb4-b6e7-b7caa430a4ff=== Employee Details ===

Employee ID : 101

```

Name : SHUBHAM

Salary : \$500000.0

=== Manager Details ===

Employee ID : 201

Name : shubham

Salary : \$80000.0

Department : IT

Team Size : 5

PS C:\Users\Student\Desktop\SHUBHAM JAVA>

7. Create a simple Java program that demonstrates the concept of multilevel inheritance in the context of an

- a. Educational Institution**
- a. Banking**
- a. Library**
- a. Hospitals**

Code:

// Base class

class Institution {

```
protected String institutionName;
```

```
public Institution(String institutionName) {  
    this.institutionName = institutionName;  
}
```

```
public void showInstitutionInfo() {  
    System.out.println("Institution Name: " + institutionName);  
}  
}
```

```
// Derived from Institution
```

```
class Department extends Institution {  
    protected String departmentName;
```

```
public Department(String institutionName, String departmentName) {  
    super(institutionName); // Call to Institution constructor  
    this.departmentName = departmentName;  
}
```

```
public void showDepartmentInfo() {  
    showInstitutionInfo();  
    System.out.println("Department Name: " + departmentName);
```

```
}  
  
}  
  
// Derived from Department (multilevel)  
class Student extends Department {  
    private String studentName;  
    private int rollNo;  
  
    public Student(String institutionName, String departmentName, String  
studentName, int rollNo) {  
        super(institutionName, departmentName); // Call to Department constructor  
        this.studentName = studentName;  
        this.rollNo = rollNo;  
    }  
  
    public void showStudentInfo() {  
        showDepartmentInfo();  
        System.out.println("Student Name: " + studentName);  
        System.out.println("Roll Number : " + rollNo);  
    }  
}  
  
// Main class to test
```

```

public class EducationalInstitutionDemo {

    public static void main(String[] args) {

        Student student = new Student("CSIBER", "Computer Science", "Shubham
Asawale", 1);

        System.out.println("=== Student Information ===");

        student.showStudentInfo();

    }

}

```

Output:

```

PS C:\Users\Student\Desktop\SHUBHAM JAVA> c:: cd
'c:\Users\Student\Desktop\SHUBHAM JAVA'; & 'C:\Program Files\Java\jdk-
17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\Student\AppData\Roaming\Code\User\workspaceStorage\e2ab9cc7e87
1075dc6dfb79c14c660c5\redhat.java\jdt_ws\SHUBHAM JAVA_8fd19416\bin'
'EducationalInstitutionDemo'

ceStorage\x5ce2ab9cc7e871075dc6dfb79c14c660c5\x5credhat.java\x5cjdt_ws\x5
cSHUBHAM JAVA_8fd19416\x5cbin' 'EducationalInstitutionDemo' ;39b0189a-
9fd4-4499-b71d-7df3bd89e473=== Student Information ===

```

Institution Name: CSIBER

Department Name: Computer Science

Student Name: Shubham Asawale

Roll Number : 1

```

PS C:\Users\Student\Desktop\SHUBHAM JAVA>

```

Q.8 Two Way communication using java Socket

Code:

Client.java

import java.io.;*

import java.net.;*

public class Client {

public static void main(String[] args) throws IOException {

Socket socket = new Socket("localhost", 1234);

System.out.println("Connected to server.");

*BufferedReader in = new BufferedReader(new
 InputStreamReader(socket.getInputStream()));*

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

*BufferedReader consoleInput = new BufferedReader(new
 InputStreamReader(System.in));*

String messageFromServer;

while (true) {

System.out.print("You: ");

```
String userInput = consoleInput.readLine();  
out.println(userInput);  
  
messageFromServer = in.readLine();  
System.out.println("Server: " + messageFromServer);  
}  
}  
}
```

Server.java

```
import java.io.*;  
import java.net.*;  
  
public class Server {  
    public static void main(String[] args) throws IOException {  
        ServerSocket serverSocket = new ServerSocket(1234);  
        System.out.println("Server is waiting for a client...");  
  
        Socket socket = serverSocket.accept();  
        System.out.println("Client connected.");  
    }  
}
```

```
    BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

    PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

    BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in));

    String messageFromClient;
    while ((messageFromClient = in.readLine()) != null) {
        System.out.println("Client: " + messageFromClient);
        System.out.print("You: ");
        String response = consoleInput.readLine();
        out.println(response);
    }

    socket.close();
    serverSocket.close();
}
```

Output:

```

PS C:\Users\Student\Desktop\Shubham\Server> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Student\Desktop\Shubham\Server\bin' 'Server'
Server is waiting for a client...
Client connected.
Client: HI my name is shubham
You: hello from jamnik
Client: 1234
You: hii shubham
PS C:\Users\Student\Desktop\Shubham\Server>

```

```

PS C:\Users\Student\Desktop\Shubham\Server> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Student\Desktop\Shubham\Server\bin' 'Client'
Connected to server.
You: HI my name is shubham
Server: hello from jamnik
You: 1234

```

Q.9 Employee CRUD Operation using java and MYSQL

Code:

```
import java.sql.*;
```

```
import java.util.Scanner;
```

```
public class EmployeeCRUD {
```

```
    private static final String URL =
    "jdbc:mysql://localhost:3306/employee_db";
```

```
    private static final String USER = "root";
```

```
    private static final String PASS = "root";
```

```
private static Connection connect() throws SQLException {  
    return DriverManager.getConnection(URL, USER, PASS);  
}
```

```
public static void addEmployee(String name, String department,  
double salary) {  
    String sql = "INSERT INTO employees (name, department, salary)  
VALUES (?, ?, ?)";  
    try (Connection conn = connect(); PreparedStatement pstmt =  
conn.prepareStatement(sql)) {  
        pstmt.setString(1, name);  
        pstmt.setString(2, department);  
        pstmt.setDouble(3, salary);  
        pstmt.executeUpdate();  
        System.out.println("Employee added successfully.");  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```
public static void viewEmployees() {
```

```

String sql = "SELECT * FROM employees";

try (Connection conn = connect(); Statement stmt =
conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {

    System.out.println("ID | Name | Department | Salary");

    while (rs.next()) {

        System.out.printf("%d | %s | %s | %.2f%n",
            rs.getInt("id"), rs.getString("name"),
            rs.getString("department"), rs.getDouble("salary"));

    }

} catch (SQLException e) {

    e.printStackTrace();

}
}

```

```

public static void updateEmployee(int id, String name, String
department, double salary) {

    String sql = "UPDATE employees SET name = ?, department = ?,
salary = ? WHERE id = ?";

    try (Connection conn = connect(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {

        pstmt.setString(1, name);

        pstmt.setString(2, department);
    }
}

```

```
pstmt.setDouble(3, salary);

pstmt.setInt(4, id);

int rows = pstmt.executeUpdate();

if (rows > 0) {

    System.out.println("Employee updated successfully.");

} else {

    System.out.println("Employee not found.");

}

} catch (SQLException e) {

    e.printStackTrace();

}

}
```

```
public static void deleteEmployee(int id) {

    String sql = "DELETE FROM employees WHERE id = ?";

    try (Connection conn = connect(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {

        pstmt.setInt(1, id);

        int rows = pstmt.executeUpdate();

        if (rows > 0) {

            System.out.println("Employee deleted successfully.");

        }

    }

}
```

```
    } else {  
        System.out.println("Employee not found.");  
    }  
} catch (SQLException e) {  
    e.printStackTrace();  
}  
}  
  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    while (true) {  
        System.out.println("\n--- Employee Management System ---");  
        System.out.println("1. Add Employee");  
        System.out.println("2. View Employees");  
        System.out.println("3. Update Employee");  
        System.out.println("4. Delete Employee");  
        System.out.println("5. Exit");  
        System.out.print("Choose an option: ");  
        int choice = Integer.parseInt(scanner.nextLine());  
  
        switch (choice) {
```

case 1:

```
System.out.print("Name: ");  
String name = scanner.nextLine();  
System.out.print("Department: ");  
String dept = scanner.nextLine();  
System.out.print("Salary: ");  
double salary = Double.parseDouble(scanner.nextLine());  
addEmployee(name, dept, salary);  
break;
```

case 2:

```
viewEmployees();  
break;
```

case 3:

```
System.out.print("Enter ID to update: ");  
int uid = Integer.parseInt(scanner.nextLine());  
System.out.print("New Name: ");  
name = scanner.nextLine();  
System.out.print("New Department: ");  
dept = scanner.nextLine();  
System.out.print("New Salary: ");  
salary = Double.parseDouble(scanner.nextLine());
```

```
        updateEmployee(uid, name, dept, salary);
```

```
        break;
```

```
    case 4:
```

```
        System.out.print("Enter ID to delete: ");
```

```
        int did = Integer.parseInt(scanner.nextLine());
```

```
        deleteEmployee(did);
```

```
        break;
```

```
    case 5:
```

```
        System.out.println("Goodbye!");
```

```
        return;
```

```
    default:
```

```
        System.out.println("Invalid option.");
```

```
    }
```

```
}
```

```
}
```

```
}
```

Output:

```
argfile' 'EmployeeCRUD'
```

```
--- Employee Management System ---
```

1. Add Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Exit

Choose an option: 1

Name: Shubham

Department: IT

Salary: 100000

Employee added successfully.

```
--- Employee Management System ---
```

1. Add Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Exit

Choose an option: 2

ID	Name	Department	Salary
----	------	------------	--------

1	Shubham	IT	100000.00
---	---------	----	-----------

```
--- Employee Management System ---
```

1. Add Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Exit

Choose an option: 3

Enter ID to update: 1

New Name: Shubham Asawale

New Department: CS

New Salary: 500000

Employee updated successfully.

```
--- Employee Management System ---
```

1. Add Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Exit

Choose an option: