CodInClub

Powered By BridgeLabz

Best Programming Practice

1. Use **`static`** for shared values and utility methods to reduce memory usage and avoid redundancy.
2. Leverage **`this`** to avoid ambiguity when initializing attributes.
3. Declare **`final`** variables for identifiers or constants that should remain unchanged.
4. Use **`instanceof`** for safe type-checking and to prevent runtime errors during typecasting.

Sample Program 1: Bank Account System

Create a `BankAccount` class with the following features:

1. **Static**:
    - A static variable `bankName` is shared across all accounts.
    - A static method `getTotalAccounts()` to display the total number of accounts.
2. **This**:
    - Use `this` to resolve ambiguity in the constructor when initializing `accountHolderName` and `accountNumber`.
3. **Final**:
    - Use a `final` variable `accountNumber` to ensure it cannot be changed once assigned.
4. **Instanceof**:
    - Check if an account object is an instance of the `BankAccount` class before displaying its details.

```
class BankAccount {
```

```
    private static String bankName = "Global Bank";
    private static int totalAccounts = 0;
    private final int accountNumber;
    private String accountHolderName;
    private double balance;


    public BankAccount(int accountNumber, String accountHolderName, double
balance) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
        this.balance = balance;
        totalAccounts++;
    }

    public static void getTotalAccounts() {
        System.out.println("Total Accounts: " + totalAccounts);
    }

    public void displayDetails() {
        if (this instanceof BankAccount) {
            System.out.println("Bank: " + bankName + ", Account Number: " +
accountNumber +
                    ", Holder: " + accountHolderName + ", Balance: $" +
balance);
        }
    }

    public static void main(String[] args) {
        BankAccount acc1 = new BankAccount(101, "Alice", 5000);
        BankAccount acc2 = new BankAccount(102, "Bob", 3000);

        acc1.displayDetails();
        acc2.displayDetails();
        getTotalAccounts();
    }
}
```

Sample Program 2: Library Management System

Create a `Book` class to manage library books with the following features:

1. **Static**:
    - A static variable `libraryName` shared across all books.
    - A static method `displayLibraryName()` to print the library name.
2. **This**:
    - Use `this` to initialize `title`, `author`, and `isbn` in the constructor.
3. **Final**:
    - Use a `final` variable `isbn` to ensure the unique identifier of a book cannot be changed.
4. **Instanceof**:
    - Verify if an object is an instance of the `Book` class before displaying its details.

```
class Book {
   private static String libraryName = "Central Library";
   private final String isbn;
   private String title;
   private String author;


   public Book(String isbn, String title, String author) {
       this.isbn = isbn;
```

```
        this.title = title;
        this.author = author;
    }

    public static void displayLibraryName() {
        System.out.println("Library Name: " + libraryName);
    }

    public void displayDetails() {
        if (this instanceof Book) {
            System.out.println("ISBN: " + isbn + ", Title: " + title + ",
Author: " + author);
        }
    }

    public static void main(String[] args) {
        Book book1 = new Book("123-ABC", "Java Basics", "John Doe");
        Book book2 = new Book("456-DEF", "Python Guide", "Jane Smith");

        displayLibraryName();
        book1.displayDetails();
        book2.displayDetails();
    }
}
```

Sample Program 3: Employee Management System

Design an `Employee` class with the following features:

1. **Static**:
    - A static variable `companyName` shared by all employees.
    - A static method `displayTotalEmployees()` to show the total number of employees.
2. **This**:
    - Use `this` to initialize `name`, `id`, and `designation` in the constructor.
3. **Final**:
    - Use a `final` variable `id` for the employee ID, which cannot be modified after assignment.
4. **Instanceof**
    - Check if a given object is an instance of the `Employee` class before printing the employee details.

```
class Employee {
   private static String companyName = "Tech Corp";
   private static int totalEmployees = 0;
   private final int id;
   private String name;
   private String designation;

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

   public static void displayTotalEmployees() {
       System.out.println("Total Employees: " + totalEmployees);
   }

   public void displayDetails() {
       if (this instanceof Employee) {
```

```
            System.out.println("Company: " + companyName + ", ID: " + id +
", Name: " + name + ", Designation: " + designation);
        }
    }

    public static void main(String[] args) {
        Employee emp1 = new Employee(1, "Alice", "Developer");
        Employee emp2 = new Employee(2, "Bob", "Manager");

        emp1.displayDetails();
        emp2.displayDetails();
        displayTotalEmployees();
    }
}
```

Sample Program 4: Shopping Cart System

Create a `Product` class to manage shopping cart items with the following features:

1. **Static**:
    - A static variable `discount` shared by all products.
    - A static method `updateDiscount()` to modify the discount percentage.
2. **This**:
    - Use `this` to initialize `productName`, `price`, and `quantity` in the constructor.
3. **Final**:

- Use a `final` variable `productID` to ensure each product has a unique identifier that cannot be changed.

4. **Instanceof**:

   - Validate whether an object is an instance of the `Product` class before processing its details.

```
class Product {
    private static double discount = 10.0;
    private final int productID;
    private String productName;
    private double price;
    private int quantity;


    public Product(int productID, String productName, double price, int
quantity) {
        this.productID = productID;
        this.productName = productName;
        this.price = price;
        this.quantity = quantity;
    }

    public static void updateDiscount(double newDiscount) {
        discount = newDiscount;
    }

    public void displayDetails() {
        if (this instanceof Product) {
            System.out.println("Product ID: " + productID + ", Name: " +
productName +
                    ", Price: $" + price + ", Quantity: " + quantity + ",
Discount: " + discount + "%");
        }
    }

    public static void main(String[] args) {
        Product p1 = new Product(101, "Laptop", 1000, 2);
        Product p2 = new Product(102, "Phone", 500, 5);
```

```
        p1.displayDetails();
        p2.displayDetails();
        updateDiscount(15.0);
        p1.displayDetails();
    }
}
```

Sample Program 5: University Student Management

Create a `Student` class to manage student data with the following features:

1. **Static**:
    - A static variable `universityName` shared across all students.
    - A static method `displayTotalStudents()` to show the number of students enrolled.
2. **This**:
    - Use `this` in the constructor to initialize `name`, `rollNumber`, and `grade`.
3. **Final**:
    - Use a `final` variable `rollNumber` for each student that cannot be changed.
4. **Instanceof**:
    - Check if a given object is an instance of the `Student` class before performing operations like displaying or updating grades.

```
class Student {
    private static String universityName = "Global University";
    private static int totalStudents = 0;
    private final int rollNumber;
    private String name;
    private double grade;

    public Student(int rollNumber, String name, double grade) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.grade = grade;
        totalStudents++;
    }

    public static void displayTotalStudents() {
        System.out.println("Total Students: " + totalStudents);
    }

    public void displayDetails() {
        if (this instanceof Student) {
            System.out.println("University: " + universityName + ", Roll
Number: " + rollNumber + ", Name: " + name + ", Grade: " + grade);
        }
    }

    public static void main(String[] args) {
        Student s1 = new Student(1, "Alice", 9.2);
        Student s2 = new Student(2, "Bob", 8.7);

        s1.displayDetails();
        s2.displayDetails();
        displayTotalStudents();
    }
}
```

CodInClub

Powered By BridgeLabz

Sample Program 6: Vehicle Registration System

Create a `Vehicle` class with the following features:

1. **Static**:
    - A static variable `registrationFee` common for all vehicles.
    - A static method `updateRegistrationFee()` to modify the fee.
2. **This**:
    - Use `this` to initialize `ownerName`, `vehicleType`, and `registrationNumber` in the constructor.
3. **Final**:
    - Use a `final` variable `registrationNumber` to uniquely identify each vehicle.
4. **Instanceof**:
    - Check if an object belongs to the `Vehicle` class before displaying its registration
    - details.

```
class Vehicle {
    private static double registrationFee = 5000;
    private final String registrationNumber;
    private String ownerName;
    private String vehicleType;


    public Vehicle(String registrationNumber, String ownerName, String
vehicleType) {
        this.registrationNumber = registrationNumber;
        this.ownerName = ownerName;
        this.vehicleType = vehicleType;
    }
```

```
    public static void updateRegistrationFee(double newFee) {
        registrationFee = newFee;
    }

    public void displayDetails() {
        if (this instanceof Vehicle) {
            System.out.println("Registration Number: " + registrationNumber
+ ", Owner: " + ownerName +
                    ", Vehicle Type: " + vehicleType + ", Fee: $" +
registrationFee);
        }
    }

    public static void main(String[] args) {
        Vehicle v1 = new Vehicle("XYZ123", "Alice", "Car");
        Vehicle v2 = new Vehicle("ABC789", "Bob", "Bike");

        v1.displayDetails();
        v2.displayDetails();
        updateRegistrationFee(5500);
        v1.displayDetails();
    }
}
```

Sample Program 7: Hospital Management System

Create a `Patient` class with the following features:

1. **Static**:
    - A static variable `hospitalName` shared among all patients.
    - A static method `getTotalPatients()` to count the total patients admitted.
2. **This**:

- Use `this` to initialize `name`, `age`, and `ailment` in the constructor.

3. **Final**:
   - Use a `final` variable `patientID` to uniquely identify each patient.
4. **Instanceof**:
   - Check if an object is an instance of the `Patient` class before displaying its details.

```
class Patient {
    private static String hospitalName = "City Hospital";
    private static int totalPatients = 0;
    private final int patientID;
    private String name;
    private int age;
    private String ailment;


    public Patient(int patientID, String name, int age, String ailment) {
        this.patientID = patientID;
        this.name = name;
        this.age = age;
        this.ailment = ailment;
        totalPatients++;
    }

    public static void getTotalPatients() {
        System.out.println("Total Patients: " + totalPatients);
    }

    public void displayDetails() {
        if (this instanceof Patient) {
            System.out.println("Hospital: " + hospitalName + ", Patient ID:
" + patientID + ", Name: " + name + ", Age: " + age + ", Ailment: " +
ailment);
        }
    }
```

```
    public static void main(String[] args) {
        Patient p1 = new Patient(101, "Alice", 30, "Flu");
        Patient p2 = new Patient(102, "Bob", 40, "Diabetes");

        p1.displayDetails();
        p2.displayDetails();
        getTotalPatients();
    }
}
```