CodInClub

Powered By BridgeLabz

Best Practices in Constructors

1. **Use `this` Keyword**:
   - Avoid ambiguity when parameter names are the same as attribute names.
   - Example: `this.customerName = customerName;`
2. **Keep Logic Simple**:
   - Avoid heavy computations or database calls inside constructors.
3. **Provide Multiple Constructors**:
   - Support various initialization scenarios by overloading constructors.
4. **Encapsulate Logic**:
   - Use private methods (like `calculatePrice()`) to keep constructors clean.

Best Practices in Access Modifiers

Use the Least Privilege:

- Start with the most restrictive modifier (`private`) and relax it as needed (`protected` or `public`).

Encapsulation:

- Always make attributes `private` and use getters/setters for controlled access.

Protected Usage:

- Use `protected` only when inheritance is required and controlled access is necessary.

Avoid Overexposure:

- Limit the use of `public` to methods or classes that are meant to be accessed by external code.

Package Access:

- Use the default (package-private) modifier to restrict access to the same package unless explicitly needed elsewhere.

Avoid Leaks:

- Be cautious with exposing mutable objects, like collections, via getters. Return a copy or an unmodifiable view when possible.

Level 1 Practice Programs

1. Create a `Book` class with attributes `title`, `author`, and `price`. Provide both default and parameterized constructors.

```java
public class Book {

   private String title;
   private String author;
   private double price;

   //default Constructor
   public Book() {
       this.title = "Ek Kitaab";
       this.author = "Unknown";
       this.price = 0;
   }


   //parameterized Constructor
   public Book(String title, String author, double price) {
       this.title = title;
       this.author = author;
       this.price = price;
   }


   public void displayDetails() {
       System.out.println("Title: " + title);
       System.out.println("Author: " + author);
       System.out.println("Price: " + price);
   }
```

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

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

    public String getAuthor() {
        return this.author;
    }

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

    public double getPrice() {
        return this.price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

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

        //Object for Book Class
        Book b1 = new Book();
        b1.displayDetails();

        Book b2 = new Book("Hello Hii", "Naman", 1000);
        b2.displayDetails();

    }
}
```

2. Write a `Circle` class with a `radius` attribute. Use constructor chaining to initialize `radius` with default and user-provided values.

```java
public class Circle {

    private double radius;


    //default consturctor
    public Circle() {
        this(1.1);
    }

    //parameterized constructor
    public Circle(double radius) {
        this.radius = radius;
    }

    public void printArea(double radius) {
        System.out.println("Area: " + (3.14 * radius * radius));
    }

    public double getRadius() {
        return radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }

}
```

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


        //Object for Circle Class
        Circle c1 = new Circle();
        c1.printArea();

        Circle c2 = new Circle(2.2);
        c2.printArea();

    }
}
```

3. Create a `Person` class with a copy constructor that clones another person's attributes.

```
public class Person {

    private String name;
    private int age;

    //default Constructor
    public Person() {
        this("Unknown", 22);
    }

    //parameterized Constructor
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    //Copy Constructor
    public Person(Person p2) {
```

```java
        this.name = p2.name;
        this.age = p2.age;
    }

    public void displayDetails() {
        System.out.println("Name : " + this.name + "\nAge: " + this.age);

    }

    public String getName() {
        return this.name;
    }

    public void setName(String name){
        this.name = name;
    }

    public int getAge() {
        return this.age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

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


        //Object for Person Class
        Person p1 = new Person();
        p1.displayDetails();

        Person p2 = new Person("Hello", 20);
        p2.displayDetails();

        Person p3 = new Person(p2);
        p3.displayDetails();
 }}
```

4. **Hotel Booking System**: Create a `HotelBooking` class with attributes `guestName`, `roomType`, and `nights`. Use default, parameterized, and copy constructors to initialize bookings.

```
public class HotelBooking {


    private String guestName;
    private String roomType;
    private int nights;

    //default constructor
    public HotelBooking() {
        this("ABC", "Supreme", 4);
    }

    //parameterized constructor
    public HotelBooking(String guestName, String roomType, int nights) {
        this.guestName = guestName;
        this.roomType = roomType;
        this.nights = nights;
    }

    //Copy Constructor
    public HotelBooking(HotelBooking h2) {
        this.guestName = h2.guestName;
        this.roomType = h2.roomType;
        this.nights = h2.nights;
    }

    public void displayDetails() {
        System.out.println("Guest Name: " + this.guestName + "\nRoom Type: " +
this.roomType + "\nNights: " + this.nights);
    }

    public String getGuestName() {
        return this.guestName;
    }
```

```java
    public void setGuestName(String name) {
        this.guestName = name;
    }

    public String getRoomType() {
        return this.roomType;
    }

    public void setRoomType(String roomType) {
        this.roomType = roomType;
    }

    public int getNights() {
        return this.nights;
    }

    public void setNights(int nights) {
        this.nights = nights;
    }

}
```

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

        //Object for HotelBooking Class
        HotelBooking h1 = new HotelBooking();
        h1.displayDetails();

        HotelBooking h2 = new HotelBooking("Hiii", "Normal", 2);
        h2.displayDetails();

        HotelBooking h3 = new HotelBooking(h2);
        h3.displayDetails();

    }
}
```

5. **Library Book System**: Create a `Book` class with attributes `title`, `author`, `price`, and `availability`. Implement a method to borrow a book.

```java
public class Library {

    private String title;
    private String author;
    private int price;
    private boolean availability;

    //defaut Constructor
    public Library() {
        this("Meri Book", "Nahi Bataaunga", 10000, true);
    }

    //parameterized Constructor
    public Library(String title, String author, int price, boolean
availability) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.availability = availability;
    }

    public void displayDetails() {
        System.out.println("Title: " + this.title);
        System.out.println("Author: " + this.author);
        System.out.println("Price: " + this.price);
        System.out.println("Availability: " + this.availability);
    }

    public void borrowBook() {
        if (this.availability) {
            switchAvailability();
            System.out.println("Book Borrowed!");
        } else {
            System.out.println("This book is not available!");
        }
    }
    public String getTitle() {
        return this.title;
    }

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

    public String getAuthor() {
        return this.author;
    }

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

    public int getPrice() {
        return this.price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public boolean showAvailability() {
        return this.availability;
    }

    public void switchAvailability() {
        this.availability = !this.availability;
    }

}
```

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

        //Object for Library Class
        Library l1 = new Library();
        l1.displayDetails();

        Library l2 = new Library("ABC", "abc", 10, true);
        l2.displayDetails();

        l2.borrowBook();

    }
}
```

6. **Car Rental System**: Create a `CarRental` class with attributes `customerName`, `carModel`, and `rentalDays`. Add constructors to initialize the rental details and calculate total cost.

```java
public class CarRental {

    private String customerName;
    private String carModel;
    private int rentalDays;
    private int cost;

    // parameterized Constructor
    public CarRental(String customerName, String carModel, int rentalDays) {
        this.customerName = customerName;
        this.carModel = carModel;
        this.rentalDays = rentalDays;
        this.cost = 500;
    }

    public void displayDetails() {
        System.out.println("Customer Name: " + customerName + "\nCar Model: " +
carModel + "\nRental Days: "
                + rentalDays + "\nTotal Cost: " + (rentalDays * this.cost));
    }

    public void totalCost() {
        System.out.println("Total Cost: " + (this.rentalDays * this.cost));
    }

    public String getCustomerName() {
        return this.customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getCarModel() {
        return this.carModel;
    }
    public void setCarModel(String carModel) {
        this.carModel = carModel;
    }

    public int getRentalDays() {
        return this.rentalDays;
    }

    public void setRentalDays(int rentalDays) {
        this.rentalDays = rentalDays;
    }

}
```

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

        //Object for CarRental Class
        CarRental car1 = new CarRental("Me", "Car", 2);
        car1.displayDetails();
        car1.totalCost();

    }
}
```

1. Instance vs. Class Variables and Methods

Problem 1: Product Inventory

Create a `Product` class with:

- Instance Variables: `productName`, `price`.
- Class Variable: `totalProducts` (shared among all products).
- Methods:
  - An instance method `displayProductDetails()` to display the details of a product.
  - A class method `displayTotalProducts()` to show the total number of products created.

```
public class Product {

    private String productName;
    private int productPrice;

    static int totalProducts = 0;

    //parameterized Constructor
    public Product(String productName, int productPrice) {
        this.productName = productName;
        this.productPrice = productPrice;
        Product.totalProducts++;
    }

    public void displayProductDetails() {
        System.out.println("Product Name: " + this.productName + "\nProduct
Price: " + this.productPrice);
    }

    public static void displayTotalProducts() {
        System.out.println("Total Products: " + Product.totalProducts);
    }

    public String getProductName(){
        return this.productName;
    }
```

```java
    public void setProductName(String productName) {
        this.productName = productName;
    }

    public int getProductPrice() {
        return this.productPrice;
    }

    public void setProductPrice(int productPrice) {
        this.productPrice = productPrice;
    }

}
```

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

        //Object for Product Class
        Product p1 = new Product("null", 0);

        p1.displayProductDetails();
        Product.displayTotalProducts();

    }
}
```

Problem 2: Online Course Management

Design a `Course` class with:

- Instance Variables: `courseName`, `duration`, `fee`.
- Class Variable: `instituteName` (common for all courses).
- Methods:
    - An instance method `displayCourseDetails()` to display the course details.
    - A class method `updateInstituteName()` to modify the institute name for all courses.

```java
public class Course {

    private String courseName;
    private int duration;
    private int fee;

    static String instituteName = "ABC Institute";

    public Course(String courseName, int duration, int fee) {
        this.courseName = courseName;
        this.duration = duration;
        this.fee = fee;
    }

    public void displayCourseDetails() {
        System.out.println("Course Name: " + this.courseName + "\nDuration: " +
this.duration + "\nFee: " + this.fee);
    }

    public static void updateInstituteName(String instituteName) {
        Course.instituteName = instituteName;
    }

    public String getCourseName() {
        return this.courseName;
    }

    public void setCourseName(String courseName) {
        this.courseName = courseName;
    }
    public int getDuration() {
        return this.duration;
    }

    public void setDuration(int duration) {
        this.duration = duration;
    }

    public int getFee() {
        return this.fee;
    }

    public void setFee(int fee) {
        this.fee = fee;
    }
}
```

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

        //Object for Course Class
        Course c1 = new Course("ABC", 3, 2000);
        c1.displayCourseDetails();

        Course.updateInstituteName("New Name");
        System.out.println(Course.instituteName);

    }
}
```

Problem 3: Vehicle Registration

Create a Vehicle class to manage the details of vehicles:

- Instance Variables: ownerName, vehicleType.
- Class Variable: registrationFee (fixed for all vehicles).
- Methods:
    - An instance method displayVehicleDetails() to display owner and vehicle details.
    - A class method updateRegistrationFee() to change the registration fee.

```
public class Vehicle {

    private String ownerName;
    private String vehicleType;

    static int registrationFee = 1000;

    //parameterized constructor
    public Vehicle(String ownerName, String vehicleType) {
        this.ownerName = ownerName;
        this.vehicleType = vehicleType;
    }


    //method for displaying details
    public void displayDetails() {
        System.out.println("Owner Name: " + this.ownerName + "\nVehicle Type: "
+ this.vehicleType);
    }


    //method for updating registration fee
    public static void updateRegistrationFee(int registrationFee) {
        Vehicle.registrationFee = registrationFee;
    }

    //getters and setters for instance variables
    public String getOwnerName() {
        return this.ownerName;
    }
```

```java
    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }

    public String getVehicleType() {
        return this.vehicleType;
    }

    public void setVehicleType(String vehicleType) {
        this.vehicleType = vehicleType;
    }

}
```

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


        //Object for Vehicle Class
        Vehicle v1 = new Vehicle("Name", "4 Wheeler");
        v1.displayDetails();

        Vehicle.updateRegistrationFee(2000);

    }
}
```

2. Access Modifiers

Problem 1: University Management System

Create a `Student` class with:

- `rollNumber` (public).
- `name` (protected).
- `CGPA` (private).

Write methods to:

- Access and modify `CGPA` using public methods.
- Create a subclass `PostgraduateStudent` to demonstrate the use of protected members.

```java
public class Student {

    public int rollNumber;
    protected String name;
    private double cgpa;

    //parameterized constructor
    public Student(int rollNumber, String name, double cgpa) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.cgpa = cgpa;
    }

    public void displayDetails() {
        System.out.println("Roll Number: " + this.rollNumber);
        System.out.println("Name: " + this.name);
        System.out.println("CGPA: " + this.cgpa);
    }

    //getter and setter for private instances
    public double getCgpa() {
        return this.cgpa;
    }
    public void setCgpa(double cgpa) {
        this.cgpa = cgpa;
    }
}
```

```
public class PostgraduateStudent extends Student {
    private String specialization;

    public PostgraduateStudent(int rollNumber, String name, double cgpa, String
specialization) {
        super(rollNumber, name, cgpa);
        this.specialization = specialization;
    }

    public void displayPGDetails() {
        System.out.println("Roll Number: " + rollNumber);

        System.out.println("Name: " + name); // Accessing protected member

        System.out.println("Specialization: " + specialization);

        System.out.println("CGPA: " + getCgpa()); // Using getter for private
member
    }}
```

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

        //Object for Student and PostgraduateStudent Classes
        Student s1 = new Student(1, "Name", 8.96);
        s1.displayDetails();

        PostgraduateStudent pg1 = new PostgraduateStudent(2, "Name2", 8.6,
"Core CSE");
        pg1.displayPGDetails();

    }}
```

Problem 2: Book Library System

Design a Book class with:

- ISBN (public).
- title (protected).
- author (private).

Write methods to:

- Set and get the author name.
- Create a subclass EBook to access ISBN and title and demonstrate access modifiers.

```
public class Book {

    public int ISBN;
    protected String title;
    private String author;

    public Book(int ISBN, String title, String author) {
        this.ISBN = ISBN;
        this.title = title;
        this.author = author;
    }

    public void displayDetails() {
        System.out.println("ISBN: " + this.ISBN + "\nTitle: " + this.title
+ "\nAuthor: " + this.author);
    }

    // getter and setter for author
    public String getAuthor() {
        return this.author;
    }

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

```java
public class EBook extends Book {

    public int edition;

    public EBook(int ISBN, String title, String author, int edition) {
        super(ISBN, title, author);
        this.edition = edition;
    }

    public void displayEDetails() {

        System.out.println("ISBN: " + this.ISBN); //public

        System.out.println("Title: " + this.title); // accessing protected
instance

        System.out.println("Author: " + getAuthor());// accesing private
through getter

        System.out.println("Edition: " + this.edition); //public

    }

}
```

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

        //Object for Book and EBook
        Book b1 = new Book(1234, "Title", "Author");
        b1.displayDetails();

        EBook eb1 = new EBook(9876, "Title2", "Author2", 1);
        eb1.displayEDetails();
    }
}
```

Problem 3: Bank Account Management

Create a `BankAccount` class with:

- `accountNumber` (public).
- `accountHolder` (protected).
- `balance` (private).

Write methods to:

- Access and modify `balance` using public methods.
- Create a subclass `SavingsAccount` to demonstrate access to `accountNumber` and `accountHolder`.

```
public class BankAccount {

    public int accountNumber;
    protected String accountHolder;
    private double balance;

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

    public void displayDetails() {
        System.out.println("Account Number: " + this.accountNumber);
        System.out.println("Account Holder: " + this.accountHolder);
        System.out.println("Balance: " + this.balance);
    }

    // getter and setter
    public double getBalance() {
        return this.balance;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }

}
```

```
public class SavingsAccount extends BankAccount {

    String accountType;

    public SavingsAccount(int accountNumber, String accountHolder, double
balance, String accountType) {
        super(accountNumber, accountHolder, balance);
        this.accountType = accountType;
    }

    public void displaySavingsDetails() {

        System.out.println("Account Number: " + this.accountNumber);

        System.out.println("Account Holder: " + this.accountHolder); //
accessing protected instance

        System.out.println("Balance: " + this.getBalance()); // accessing
through getter

        System.out.println("Account Type: " + this.accountType);

    }
}
```

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

        //Object for BankAccount and SavingsAccount
        BankAccount bank1 = new BankAccount(10011001, "Me", 20000);
        bank1.displayDetails();

        SavingsAccount savings1 = new SavingsAccount(10011002, "You", 0,
"Savings");
        savings1.displaySavingsDetails();

    }
}
```

Problem 4: Employee Records

Develop an `Employee` class with:

- `employeeID` (public).
- `department` (protected).
- `salary` (private).

Write methods to:

- Modify `salary` using a public method.
- Create a subclass `Manager` to access `employeeID` and `department`.

```java
public class Employee {

    public int employeeId;
    protected String department;
    private int salary;

    public Employee(int employeeId, String department, int salary) {
        this.employeeId = employeeId;
        this.department = department;
        this.salary = salary;
    }

    public void displayDetails() {
        System.out.println("Employee Id: " + this.employeeId);
        System.out.println("Department: " + this.department);
        System.out.println("Salary: " + this.salary);
    }

    // getter and setter
    public int getSalary() {
        return this.salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }
}
```

```java
public class Manager extends Employee {

    public String managerName;

    public Manager(int employeeId, String department, int salary, String
managerName) {
        super(employeeId, department, salary);
        this.managerName = managerName;
    }

    public void displayDetailsWithManager() {

        System.out.println("Employee Id: " + this.employeeId); //public

        System.out.println("Department: " + this.department); // accessing
protected instance

        System.out.println("Salary: " + getSalary()); // accessing salary
through getter method

        System.out.println("Manager Name: " + this.managerName); // public
    }
}
```

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

        //Object for Employee and Manager Class
        Employee e1 = new Employee(1, "Cashier", 50000);
        e1.displayDetails();

        Manager m1 = new Manager(2, "Central", 100000, "Manager");
        m1.displayDetailsWithManager();

    }
}
```