**Kamran Gul | 023-25-0161 | Section C**

# LAB–3: Objects, Instance Methods & Encapsulation (Java)

## Objectives

1. Understand the difference between static and non-static members
2. Define classes with instance variables
3. Create and use objects in Java
4. Call non-static methods using objects
5. Understand reference variables and object aliasing
6. Use constructors to initialize objects
7. Apply basic encapsulation using private
8. Implement getters and setters
9. Use the this keyword correctly

## Prerequisite

Students should have completed Lab–2 and must be familiar with static methods, control statements, arrays, and basic Java syntax.

## 1. Transition from Static to Object-Oriented Programming

Static methods belong to the class, whereas non-static methods belong to objects. Object-oriented programming focuses on methods operating on object data.

### Example

```
// Static
static int add(int a, int b) { return a + b; }

// Non-static
int add(int a, int b) { return a + b; }
```

## 2. Class and Object Basics

A class is a blueprint, and an object is an instance created from that class.

### Example

```
class Student {
   String name;
   int age;
}

Student s1 = new Student();
```

## 3. Instance Variables vs Local Variables

Instance variables belong to objects, while local variables exist only within methods.

#### Example

```
class Student {
   String name;
   void show() {
      int x = 10;
   }
}
```

### 4. Object Creation and Method Calling

Non-static members are accessed using objects and the dot operator.

#### Example

```
Student s1 = new Student();
s1.name = "Ali";
s1.age = 20;
s1.display();
```

### 5. Reference Variables and Object Aliasing

Multiple references can point to the same object.

#### Example

```
Student s1 = new Student();
Student s2 = s1;
s1.name = "Ali";
System.out.println(s2.name);
```

### 6. Constructors

Constructors initialize objects and are called automatically.

#### Example

```
class Student {
   String name;
   int age;
   Student(String n, int a) {
      name = n;
      age = a;
   }
}
```

### 7. Encapsulation

Encapsulation hides data using private and provides controlled access.

#### Example

```
class Student {
   private int age;
}
```

## 8. Getters and Setters

Getters and setters access private data safely.

#### Example

```
class Student {
   private int age;
   void setAge(int a) { age = a; }
   int getAge() { return age; }
}
```

## 9. this Keyword

this refers to the current object and resolves naming conflicts.

#### Example

```
class Student {
   private int age;
   void setAge(int age) {
      this.age = age;
   }
}
```

## Practice Tasks

Instructions:
• All methods must be non-static (except main).
• Use private data members wherever applicable.
• Access data using constructors, getters, and setters.
• Do not write logic directly inside main.

### Task 1: Student Profile Management

Create a class named Student with the following private fields:
• id (int)
• name (String)
• age (int)
• cgpa (double)

Requirements:
1. Create setters and getters for all fields.
2. Validate age (15–60) and cgpa (0.0–4.0).
3. If invalid data is provided, do not update the value and display a message.
4. Create an instance method display() to print complete student information.

In main:
• Create two Student objects.
• Set values using setters and display both objects.

# Solution:

```
class Student{
	private int id;
	private String name;
	private int age;
	private double cgpa;

	void setId(int id){
		this.id = id;
	}
	int getId(){
		return id;
	}
	void setName(String name){
		this.name = name;
	}
	String getName(){
		return name;
}
```

```
	void setAge(int age){
		if(age >=15 && age <=60){
			this.age = age;
		}
		else
			System.out.print("Invalid age entered");
	}
	int getAge(){
		return age;
	}
	boolean isOkay = false;
	void setCgpa(double cgpa){
		if(cgpa >=0.0 && cgpa <=4.0){
			this.cgpa = cgpa;
			isOkay = true;
		}
		else
			System.out.println("Invalid cgpa entered");
		}
	double getCgpa(){
		return cgpa;
	}
	void display(){
		System.out.println(id);
		System.out.println(name);
		System.out.println(age);
		if(isOkay==true){
			System.out.println(cgpa);
	}}
	public static void main(String[] args){
		Student std = new Student();
		std.setId(123);
		std.setName("Kamran");
		std.setAge(20);
		std.setCgpa(7.0);
		std.display();
	}}
```

**Output**

```
Student id: 123
Name: Kamran
Age: 20
CGPA: 3.0
```

### Task 2: Bank Account System

Create a class named BankAccount with the following private fields:

- accountNumber (String)
- accountHolder (String)
- balance (double)

Requirements:

1. Use a parameterized constructor to initialize all fields.
2. Create methods deposit(amount) and withdraw(amount).
3. Deposit amount must be greater than 0.
4. Withdrawal amount must be greater than 0 and less than or equal to balance.
5. Create a method getBalance() that returns current balance.
6. Create a method printStatement() to display account details.

In main:

- Create one account object.
- Perform multiple deposits and withdrawals.
- Display account details after each operation.

# Solution:

```
class bankAccount{
        private String accountNumber;
        private String accountHolder;
        private double balance;

        bankAccount(String accountNumber, String accountHolder, double balance){
                this.accountNumber = accountNumber;
                this.accountHolder = accountHolder;
                this.balance = balance;
        }
        void depositAmount(double balance){
                if(balance>=0){
                        this.balance = this.balance + balance;
                }}
        void withdrawAmount(double amount){
                if(amount>0 && amount <=this.balance){
                        this.balance = this.balance - amount;
                }}
        double getBalance(){
                return this.balance;
        }
        void printStatement(){
```

```
                System.out.println("Account Number: " + this.accountNumber);
                System.out.println("Account Holder: " + this.accountHolder);
                System.out.println("Current Balance: " + this.balance);
        }

        public static void main(String[] args){
                bankAccount account = new bankAccount("PK25004567AB339","Kamran
Gul",3489.98);
                account.depositAmount(5000);
                account.withdrawAmount(3400);
                account.printStatement();
                System.out.print("\n");
                account.depositAmount(300.76);
                account.withdrawAmount(10000);
                account.printStatement();
                System.out.print("\n");
                account.depositAmount(45360.76);
                account.withdrawAmount(13349.59);
                account.printStatement();
                }}
```

**Output**

```
Account Number: PK25004567AB339
Account Holder: Kamran Gul
Current Balance: 5089.98

Account Number: PK25004567AB339
Account Holder: Kamran Gul
Current Balance: 5390.74

Account Number: PK25004567AB339
Account Holder: Kamran Gul
Current Balance: 37401.91
```

### Task 3: Car Showroom Inventory

Create a class named Car with the following private fields:

- brand (String)
- model (String)
- year (int)
- price (double)

Requirements:

1. Create a parameterized constructor using the this keyword.
2. Create a display() method to print car details.
3. Create a getter for price only.

In main:

- Create three Car objects.
- Display all car details.
- Identify and display the most expensive car.

# Solution:

```
class car{
	private String brand;
	private String model;
	private int year;
	private double price;

	car(String brand, String model, int year, double price){
		this.brand = brand;
		this.model = model;
		this.year = year;
		this.price = price;
	}
	double getPrice(){
		return this.price;
	}
	void display(){
		System.out.println("Brand: " + brand);
		System.out.println("Model: " + model);
		System.out.println("Year: " + year);
		System.out.println("Price: " + price);
	}
	public static void main(String[] args){
		car c1 = new car("toyota","A123",2023,45000);
		car c2 = new car("mehran","B456",1999,346000);
		car c3 = new car("bmw","C789",2019,57000);

		c1.display();
		System.out.print("\n");
```

```
                c2.display();
                System.out.print("\n");
                c3.display();

                double[] array = {c1.getPrice(), c2.getPrice(), c3.getPrice()};

                double exp = array[0];
                int ind = 0;
                for(int i=1; i<array.length;i++){
                        if(array[i]>exp){
                                exp = array[i];
                                ind = i;
                        }
                }

                System.out.print("\nThe most expensive car: " + exp);
        }
}
```

**Output**

```
Brand: toyota
Model: A123
Year: 2023
Price: 45000.0

Brand: mehran
Model: B456
Year: 1999
Price: 346000.0

Brand: bmw
Model: C789
Year: 2019
Price: 57000.0

The most expensive car: 346000.0
```

### Task 4: Reference and Object Aliasing Demonstration

Create a class named Box with the following fields:

- length (double)
- width (double)

Requirements:

1. Create an instance method area() that returns the area.
2. In main:
   - Create one Box object and assign values.
   - Create another reference pointing to the same object.
   - Modify dimensions using the second reference.
   - Print area using both references.

Observation:

- Both references should reflect the same updated area.

# Solution:

```
class box{
	double length;
	double width;

	double area(){
		return length*width;
	}

	public static void main(String[] args){
		box box1 = new box();
		box box2 = box1;
		box2.length= 5;
		box2.width =4.3;

		double res1 = box1.area();
		double res2 = box2.area();
		System.out.println("\nArea of box1 before referencing: "+res1);
		System.out.println("Area of box2 before referencing: "+res2);

		box2.width= 2;
		box2.length= 6.48;
		res1 = box1.area();
		res2 = box2.area();

		System.out.println("\nArea of box1 after referencing: "+ res1);
		System.out.println("Area of box2 before referencing: "+res2+"\n");

	}
}
```

```
Output

Area of box1 before referencing: 21.5
Area of box2 before referencing: 21.5

Area of box1 after referencing: 12.96
Area of box2 before referencing: 12.96
```

### Task 5: Employee Salary Calculator

Create a class named Employee with the following private fields:

- employeeId (int)
- name (String)
- basicSalary (double)

Requirements:

1. Create setters and getters with validation (basicSalary ≥ 30000).
2. Create methods:
   - hra() → returns 20% of basic salary
   - tax() → returns 5% of basic salary
   - netSalary() → returns basic + hra − tax
   - printSlip() → prints formatted salary slip

In main:

- Create two Employee objects.
- Display salary slips for both employees.

# Solution:

```
class employeeSalaryCalculator {
   private int employeeId;
   private String name;
   private double basicSalary;

   void setId(int employeeId){
    this.employeeId = employeeId;
   }
   void setName(String name){
    this.name = name;
   }
   void setBasicSalary(double basicSalary){
      if(basicSalary>=30000){
      this.basicSalary = basicSalary;
```

```
    }
    }

  double hra(){
    double hra=(basicSalary * (20.0/100));
    return hra;
  }

  double tax(){
    double tax = (basicSalary * (5.0/100));
    return tax;
  }
  double netSalary(){
    double netSalary = (basicSalary + hra() - tax());
    return netSalary;
  }

  void printSlip(){
    System.out.println("Employee id: " + employeeId);
    System.out.println("Employee Name: " + name);
    System.out.println("Basic Salary: " + basicSalary);
    System.out.println("House Rent Allowance: " + hra());
    System.out.println("Tax: " + tax());
    System.out.println("Net Salary: " + netSalary());
  }
  public static void main(String[] args) {
    employeeSalaryCalculator e1 = new employeeSalaryCalculator();
    e1.setId(123);
    e1.setName("Kamran");
    e1.setBasicSalary(100000);
    e1.printSlip();
    System.out.println("==================================");
    employeeSalaryCalculator e2 = new employeeSalaryCalculator();
    e2.setId(786);
    e2.setName("Sarmad");
    e2.setBasicSalary(123489.55);
    e2.printSlip();
  }
}
```

**Output**

```
Employee id: 123
Employee Name: Kamran
Basic Salary: 100000.0
House Rent Allowance: 20000.0
Tax: 5000.0
Net Salary: 115000.0
==================================
Employee id: 786
Employee Name: Sarmad
Basic Salary: 123489.55
House Rent Allowance: 24697.910000000003
Tax: 6174.477500000001
Net Salary: 142012.9825
```