JAVA BOOTCAMP - I

Classes

- In Java, a **class** is a blueprint (or template) used to create objects.
- In Java, every piece of code must be inside a class
- In Java, if a class is declared as **`public`**, the **filename** must be the same as the **class name**.
- The BankAccount class below must be defined within BankAccount.java

```
public class BankAccount{

}
```

```
public class Shape{

}
```

Objects

- In Java, an **object** is an instance of a class.

```
BankAccount acc1 = new BankAccount(....);

Shape shape1 = new Shape(....);
```

- The method used above is a constructor.
- A **constructor** in Java is a special method that is used to initialize objects of a class.
- Has the same name as the class.
- Has no return type (not even `void`).

Attributes

- **Attributes** are the variables (fields) defined inside a class.
- They represent the state or properties of objects created from that class.

```
public class BankAccount {

private String accountHolderName;

private String accountNumber;

private double balance;

}
```

```
public class Shape {

private double area;

 private double perimeter;

 private ShapeType shapeType;

 }
```

Methods

- Methods are functions defined inside a class.
- They represent the behavior of an object.

```
public class BankAccount {

public void deposit(double amount);

public void withdraw(double amount);

public double getBalance();

}
```

```
public class Shape {

public double getArea();

public double getPerimeter();

public void displayShape();

}
```

Implementing Methods

```
public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Insufficient funds");
        }
    }
```

Implementing Methods

This is how we will be using OOPs practically— hiding complex implementations behind simple method calls.

```
public void displayShape(Graphics g) {
    Graphics2D g2d = (Graphics2D) g.create();
    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
    Random rand = new Random();
    Color color = new Color(rand.nextInt(255), rand.nextInt(255), rand.nextInt(255));
    g2d.setColor(color);
    Ellipse2D.Double circle = new Ellipse2D.Double(x, y, radius * 2, radius * 2);
    g2d.fill(circle);
    g2d.setStroke(new BasicStroke(3));
    g2d.setColor(Color.BLACK);
    g2d.draw(circle);
    g2d.dispose();
}
```

Packages and Subclasses

- A package in Java is a **collection of related classes and interfaces** grouped together under a single name (like a folder).

  Eg: `java.util`.

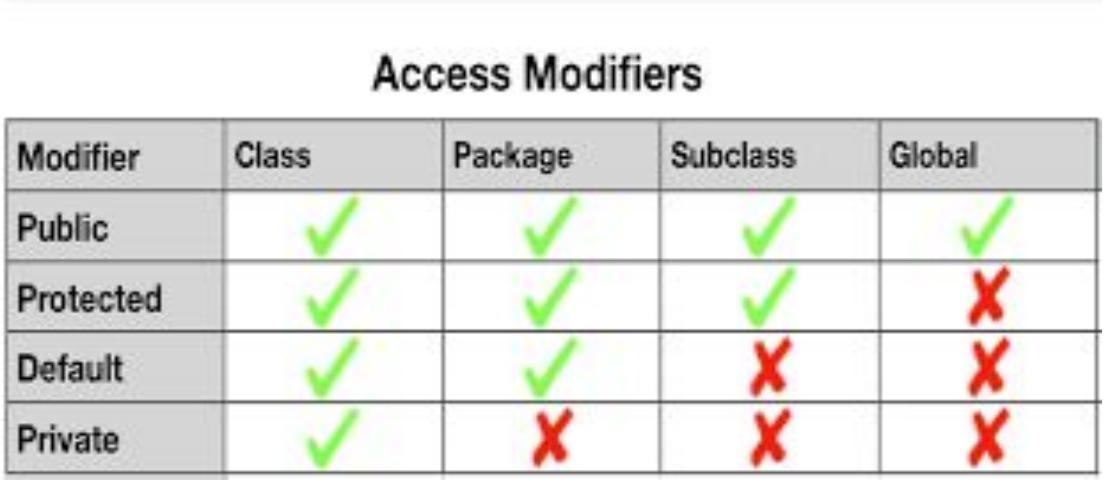
- A subclass is a **class that inherits from another class (the superclass)** and can use or override its methods and attributes.

  Eg: public class SavingsAccount extends BankAccount{

  }

Access Modifiers

- An **access modifier** in Java defines the **scope** and **accessibility** of attributes and methods.



Static Keyword

- The **`static`** keyword means a member belongs to the class itself, not to any specific object.

  Static variable : Shared by all objects of the class.
  Static method : Can be called without creating an object.

```
public class BankAccount {

private static double interestRate=3.5;

}
```

```
public class Shape {

private static int numShapes=0;

}
```

Final Keyword

- The **`final`** keyword in Java is used to declare **constant** i.e it cannot be changed once assigned**.**
- It also prevents prevent method overriding, or prevent inheritance of a class (will be covered later).
- Constants are typically declared as final and static.

```
public class BankAccount {
private final accountNumber;
}
```

```
public class Shape {
private final ShapeId;
}
```

Naming Conventions

- **Classes & Interfaces**(later) → Use **PascalCase** (capitalize each word).
  Eg: `BankAccount`

- **Methods** → Use **camelCase** (first word lowercase, subsequent words capitalized).
  Example: `depositMoney()`, `getArea()`.

- **Variables / Attributes** → Use **camelCase** as well.
  Example: `accountNumber`, `accountHolderName`.

- **Constants** → Use **UPPER_CASE with underscores**.
  Example: `MAX_BALANCE`, `INTEREST_RATE`.

- **Packages** → Use **lowercase** (often reversed domain name convention).
  Example: `com.bank.accounts`, `java.util`.

Enums

- An **enum** in Java is a special data type used to define a fixed set of **named constants**.
- It is useful when you know all possible values of a variable in advance (like days of the week, directions, account types, etc.).

```
public enum ShapeType {
    CIRCLE,
    SQUARE,
    TRIANGLE,
    RECTANGLE
} ShapeType type=ShapeType.CIRCLE;
```

```
public enum AccountType {
SAVINGS,
CURRENT,
FIXED_DEPOSIT
}
AccountType type=AccountType.SAVINGS;
```

USER INPUT

```
import java.util.Scanner;

public class UserInputExample {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

            String name = sc.next(); //nextLine()

        int age = sc.nextInt();

        sc.close();

    }

}
```

- Import `java.util.Scanner` to enable user input handling.
- Create a `Scanner` object linked to standard input (`System.in`).
- Call `nextLine()` or `nextInt()` to capture input into variables.
- Close the `Scanner` to release resources after use.

THANK YOU