DAY 6

---

1. What is a Variable?

Think of a variable as a labeled container or a box in your computer's memory. This container holds a value, and its label (the **variable name**) allows you to easily find and use that value throughout your program. Before you can use a variable, you must **declare** it by specifying its data type and name.

Java

```
// Declaring a variable named 'age' of type 'int'

int age;

// Initializing the variable with a value

age = 20;

// Or you can declare and initialize in one line

String userName = "Alice";
```

In this example, age and userName are the containers, and 20 and "Alice" are the values they hold.

---

2. Different Types of Variables in Java

Java has three main types of variables, each with a different scope and purpose.

a) Local Variables

These are variables declared inside a **method**, **constructor**, or **block**. They only exist within that specific block of code and cannot be accessed from outside. You must explicitly initialize local variables before you can use them.

- **Real-time use case**: In a method that calculates a tax, the taxAmount variable would be a local variable, as it's only needed for that specific calculation.

Java

```
public void calculateTax(double price) {

   // taxAmount is a local variable; it only exists within this method

   double taxAmount = price * 0.08;

   System.out.println("Tax: " + taxAmount);

}
```

b) Instance Variables (Non-Static Fields)

These variables are declared inside a **class** but outside of any method or block. Each **object (or instance)** of the class gets its own copy of these variables. They are automatically initialized to a default value (e.g., 0 for numbers, null for objects, false for booleans) if you don't assign one.

- **Real-time use case**: In a Car class, color, model, and speed would be instance variables. Each individual car object (e.g., a "red Ford" and a "blue Toyota") would have its own unique set of these values.

Java

```
public class Car {
  // These are instance variables
  String color;
  int speed;

  public Car(String carColor, int carSpeed) {
    this.color = carColor;
    this.speed = carSpeed;
  }
}
```

c) Static Variables (Class Variables)

These variables are declared using the **static** keyword inside a class but outside any method. Unlike instance variables, there is only **one copy** of a static variable for the entire class, shared among all objects.

- **Real-time use case**: In a banking application, a variable like bankName would be static because all accounts at that bank share the same name. Similarly, a counter that tracks the number of Car objects created would be static.

Java

```
public class BankAccount {
  // This is a static variable, shared by all BankAccount objects
  public static String bankName = "MyBank Inc.";

  // This is an instance variable
  public String accountHolderName;
}
```

DAY 6

3. How to Access Variable Data/Value

Accessing a variable's data depends on its type and scope.

- **Local variables**: You can access them directly by their name within the method or block where they are defined.
- **Instance variables**: You must create an object of the class and then access the variable using the **dot operator (.)**.
- **Static variables**: You can access them using the **class name** (e.g., BankAccount.bankName) or through an object's reference. It is best practice to use the class name to emphasize that it's a static variable.

Java

```java
public class AccessExample {
   public static void main(String[] args) {
      // Accessing a local variable
      int myNumber = 100;
      System.out.println("My number is: " + myNumber);

      // Accessing an instance variable (requires an object)
      Car myCar = new Car("Blue", 60);
      System.out.println("My car's color is: " + myCar.color);

      // Accessing a static variable (using the class name)
      System.out.println("Bank name is: " + BankAccount.bankName);
   }
}
```

---

4. Naming Conventions for Variables

Using clear and consistent naming conventions is critical for writing professional, readable code.

- Variable names should be descriptive and meaningful.
- They must start with a letter, dollar sign ($), or underscore (_).
- They cannot contain spaces or special characters (except $ and _).
- Follow **camelCase** for local and instance variables (e.g., firstName, accountBalance).
- Use all **uppercase** and underscores for static constants (e.g., final int MAX_SPEED = 200;).

5. Interview Questions on Variables

1. **What is a variable in Java?**
   - *Expected Answer:* A named memory location that stores data.
2. **Explain the difference between local, instance, and static variables.**
   - *Expected Answer:* Explain scope (method vs. class), lifecycle (temporary vs. object's life vs. class's life), and memory (stack vs. heap vs. shared memory).
3. **When would you use an instance variable versus a static variable?**
   - *Expected Answer:* Use instance variables when each object needs its own unique copy of the data. Use static variables when a piece of data is shared by all objects of the class.
4. **Do local variables have a default value? What about instance variables?**
   - *Expected Answer:* Local variables **do not** have a default value and must be initialized. Instance variables **do** have a default value (e.g., 0, null, false).
5. **What is the correct naming convention for a variable that represents a user's first name?**
   - *Expected Answer:* firstName (following camelCase).