WEEK2
Introduction to OOP: Building blocks: class, object, attributes, methods; Class and objects in java; Variable: Types (local, instance, static); declaration, initialization; comments; 'Data types;

Variable

A variable is like a **storage box** in a program that holds data. Think of it as a container where we store information, and each container has a name and a type of content it can hold.

int number = 50; // the value 50 is stored in the variable 'number'

Data Types in Java

Data types define **what kind of data** a variable can store. In Java, there are different data types for different kinds of information.
There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include Boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Java Primitive Data Types

- In Java, **primitive data types** are the most basic data types that serve as the building blocks of data manipulation. They are predefined by the language and are not objects.
- There are 8 types of primitive data types:
  - Boolean data type
  - byte data type
  - char data type
  - short data type
  - int data type
  - long data type
  - float data type
  - double data type

Data Type	Default size
boolean	1 bit
char	2 byte
byte	1 byte
short	2 byte
int	4 byte
long	8 byte
float	4 byte
double	8 byte

1. **byte**: A small whole number.
   - Example: byte b = 100; (Stores numbers from -128 to 127)
2. **short: A slightly bigger whole number.**
   - Example: short s = 3000; (Stores numbers from -32,768 to 32,767)
3. **int: Regular whole numbers (most common).**
   - Example: int num = 500; (Stores numbers from -2 billion to 2 billion)
4. **long: Really big whole numbers.**
   - Example: long bigNum = 1000000000L; (Add L at the end for long)
5. **float: Decimal numbers with less precision.**
   - Example: float pi = 3.14f; (Add f at the end for float)
6. **double: Decimal numbers with more precision.**
   - Example: double pi = 3.141592;

7. **char: A single character.**
   - Example: char letter = 'A'; (Use single quotes ')
8. **boolean: True or false values.**
   - Example: boolean isJavaFun = true;

Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:
Syntax:

type variableName = value;

Example 1: Storing Age

```
int age = 16;
System.out.println ("Your age is: " + age);
```

- **Explanation**:
  - The **variable** is age, which stores the value 16.
  - The **data type** is int because age is a whole number.

Example 2: Storing Temperature

```
double temperature = 23.5;
System.out.println ("The temperature is: " + temperature + " degrees.");
```

- **Explanation**:
  - The **variable** is temperature, which stores the value 23.5.
  - The **data type** is double because temperature can have decimal points.

Example 3: Storing a Character

```
char grade = 'A';
System.out.println ("Your grade is: " + grade);
```

- **Explanation**:
  - The **variable** is grade, which stores the value 'A'.
  - The **data type** is char because it stores a single character.

Example 4: Storing a Yes/No Value

```
boolean isStudent = true;
System.out.println ("Are you a student?” + isStudent);
```

- **Explanation**:
    - The **variable** is isStudent, which stores the value true (yes).
    - The **data type** is boolean because it's a yes/no (true/false) value.

Example 5: Storing a Name

```
String name = "Alice";
System.out.println ("Hello,” + name);
```

- **Explanation**:
    - The **variable** is name, which stores the value "Alice".
    - The **data type** is String because it stores text (a sequence of characters).

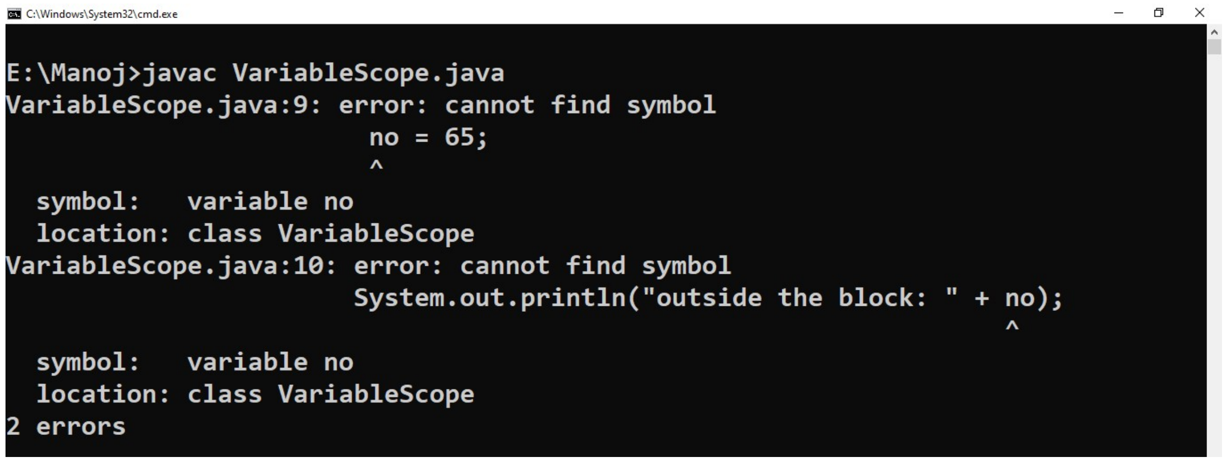
Variable Scope

Variables declared inside a block ({ }) are only accessible within that block. Variables are created and destroyed within the block. They are inaccessible outside the block where they are defined.

Demonstrate variable scope by declaring a variable inside a block and trying to access it outside of that block.

```
public class VariableScope
{
    public static void main(String[] args)
    {
        {
            int no = 42;
            System.out.println("Inside block: " + no);
        }
        no = 65;
        System.out.println("outside the block: " + no);
    }
}
```

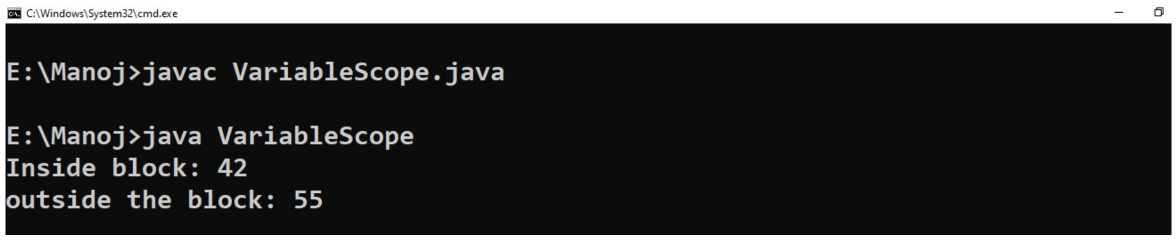
Observe the error.


C:\Windows\System32\cmd.exe

```
E:\Manoj>javac VariableScope.java
VariableScope.java:9: error: cannot find symbol
                        no = 65;
                        ^
  symbol:   variable no
  location: class VariableScope
VariableScope.java:10: error: cannot find symbol
                       System.out.println("outside the block: " + no);
                                                                   ^
  symbol:   variable no
  location: class VariableScope
2 errors
```

```
public class VariableScope
{
    public static void main(String[] args)
    {
        int no = 42;
        {
          System.out.println("Inside block: " + no);
        }
        no = 55;
        System.out.println("outside the block: " + no);

    }
}
```

OUTPUT:


C:\Windows\System32\cmd.exe

```
E:\Manoj>javac VariableScope.java

E:\Manoj>java VariableScope
Inside block: 42
outside the block: 55
```

Input Variables:

The **Scanner** class, available in the **java.util** package, is the most commonly used way to handle user input. It allows input of various data types such as integers, floats, strings, and more.

Key Methods in Scanner:

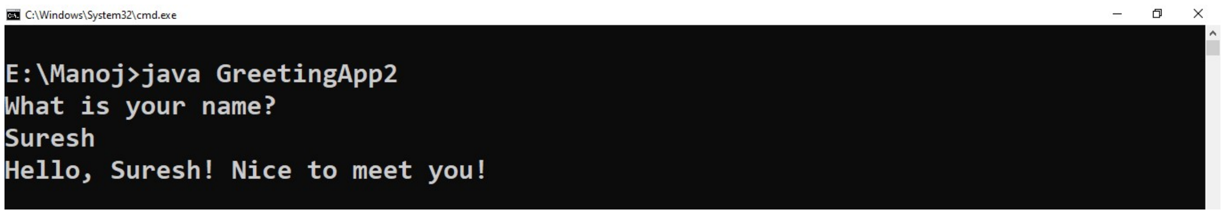
- nextInt(): Reads an integer.
- nextFloat(): Reads a float.

- nextBoolean(): Reads a boolean True/False.
- nextDouble(): Reads a double.
- nextLine(): Reads a full line of text or String.
- next(): Reads a single word or token.

Write a program that accepts user input for a name, then prints a greeting message.

```
import java.util.Scanner; // Allows us to take input from
the user

public class GreetingApp
{
     public static void main(String[] args)
     {
             // Step 1: Create a Scanner object to read input
             Scanner read = new Scanner(System.in);

             // Step 2: Ask the user for their name
             System.out.println("What is your name?");

            // Step 3: Store the user's input in a variable
             String name = read.nextLine();

             // Step 4: Print a personalized greeting
             System.out.println("Hello, " + name + "! Nice
     to meet you!");
     }
}
```

OUTPUT:



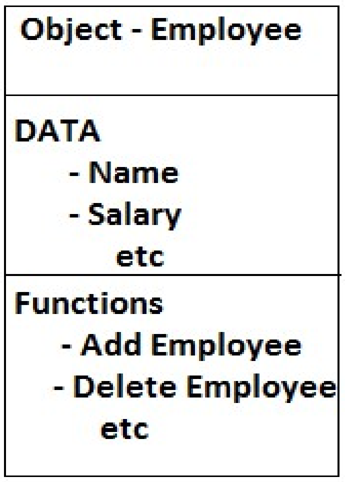
```
C:\Windows\System32\cmd.exe

E:\Manoj>java GreetingApp2
What is your name?
Suresh
Hello, Suresh! Nice to meet you!
```

Class:

- Class is a **user-defined data type**; it consists of both data members (Attributes) and member functions (Methods).
- In general, class is template for a group of objects.

Object:

- Object means it's a **real-time entity**. It may be person, place, item etc
- Object combines both data and functions



Attributes in a Class in Java

- Let us take a person as an example; the Person ***class*** has attributes and behaviors. The Person ***class*** attributes include name, gender, height, weight, and age.
- These attributes are characteristics of the Person ***class***. Behaviors are the tasks that the Person ***class*** can perform.
- For example, if the person can speak, eat, dance, sing and, sleep these are the person's behaviors. In other words, attributes are fields declared inside an object.
- These variables belong to an object and are represented with different data types.
- As seen in the code below, we have a ***Person*** class with member variables ***name*** and ***gender*** of String type, ***age*** of int type, ***height*** and ***weight*** of Do uble type.

```
class Person
{
    //Attribute Declaration
    String name;
    String Gender;
    int age;
    double height
    double weight;

    //Method Declaration
    public void speak();
    public void dance();
    public void eat();
    public void sleep();
}
```

Defining a class

- Class is a user defined data type, it contains variables and methods. These variables are termed as instances of classes, which are the actual objects.
- Usually, "**class**" is a keyword, it used to create the class definition.
- The general syntax is

```
class class_name [extends super_class_name]
{
    Variables or fields declaration;
    Methods Declaration;
}
```

- Here "**extends**" keywords extends the properties of super class into its class.
- Example -

```
class  SIMPLE
{
        String name;
        void  display()
         {
             System.out.println("My name is =" + name);
         }
}
```

Attributes Declaration

- A variable can be placed within a class. A class supports any number of variables within it.
- Example

```
class rectangle
{
    int length;
    int width;
}
```

- The "rectangle" class contains two integer type instance variable.

Methods of declaration

- The class definition contains any number of variables as well as any number of methods.
- The class definition contains at least one method called as "main() method.
- Methods can be declared inside the body of the class after the declaration of variables.

- The general syntax is

```
return_type  method_name([list of parameters])
{
    Body of the method
}
```

- **The method declaration have 4 basic parts**
  1. The type of values the method returns
  2. The name of the method
  3. List of parameters with its data type
  4. Body of the method
- Example

```
class rect
{
   int l,w;
   void getdata (int x, int y);
   {
       l=x;
      w=y;
   }
 int rectangle()
 {
    int area;
    area=l*w;
    return(area);
 }
}
```

Creating objects

- In java, the "**new**" operator is used to create object with help of class name.
- The objects are used to executes the properties of the class.
- There are two ways to create object. The General syntax is

```
classname object_name = new classname([Parameters]) ;
                              OR
Classname  objectname;
     Objectname = new Classname([Parameters]);
```

Accepting class members

- **object** and **dot (.)** operator is used to access the variables and methods of the class.
- There is different ways to access the variables and methods of the class.
- The general syntax for accessing variable is

```
Object_name.variable_name = value;
```

- The general syntax for accessing a method is

```
Object_name.Method_name([parameters]);
```

- Example

```
class  SIMPLE
{
        int a;
        void getdata(int x)
          {
              a=x;
          }
        void display()
        {
          System.out.println("a="+a);
        }
}
class M
    {
        public static void main(String arg[])
        {
              SIMPLE obj = new SIMPLE( );
              obj.a=10; //Accessing variable
              obj.getdata(10);   // Accessing Methods
              obj.display();
        }
}
```

- This program contains two classes namely "SIMPLE" and "M". The SIMPLE class contains one variable called "**a**" and two methods called "**getdata(int x)**" and "**display()".**
- The class M contains main() for executing the SIMPLE class properties.

Program 1:
Write a java program to add two numbers.

```
class Add
{
    int no1;
    int no2;

    void get_numbers(int n1, int n2)
    {
            no1 = n1;
            no2 = n2;
        }

        int add_numbers()
        {
            int sum;
            sum = no1 + no2;
            return(sum);
        }
}

class AddNos
{
    public static void main(String args[])
        {
            int result = 0;
            Add AddObj = new Add();
            AddObj.get_numbers(25, 35);
            result = AddObj.add_numbers();
            System.out.println("The sum of two no's: " + result);
        }
}
```

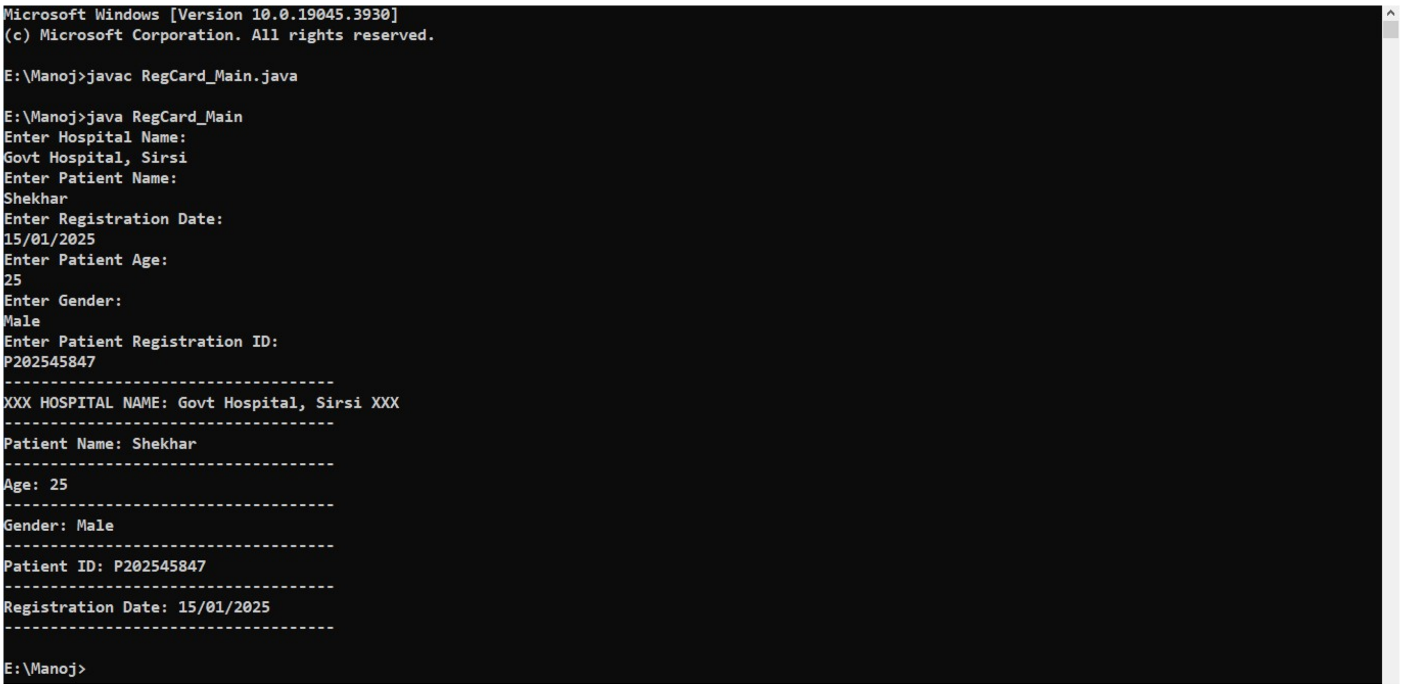
Program 2:
Write a program to read the information of two students and display their profile.

```
Import java.util.Scanner;
class Student
{
    String name;
    String regNo;
    int age;
    String address;
    int phone;

    void read()
    {
            Scanner read = new Scanner (System.in);
            System.out.println(“Enter the Name:”);
            name = read.nextLine();
            System.out.println(“Enter the Register Number:”);
            regNo = read.nextLine();
            System.out.println(“Enter the Address:”);
            address = read.nextLine();
            System.out.println(“Enter the Age:”);
            age = read.nextInt();
            System.out.println(“Enter the phone:”);
            phone = read.nextInt();
    }
    void display ()
    {
            System.out.println(“Name: “ + name);
            System.out.println(“Register No: “ + regNo);
            System.out.println(“Age: “ + age);
            System.out.println(“Address: “ + address);
            System.out.println(“Phone: “ + phone);
    }
}
```

```
public class StudentMain
{
    public static void main(String args[])
    {
            Student s1 = new Student();
            Student s2 = new Student();
            Student s3 = new Student();
            s1. read();
            s2.read();
            s3.read();

            s1.display();
            s2.display();
            s3.display();
    }
}
```

Program 3:
Write a program to display the patient registration card.

```
import java.util.Scanner;

class RegCard
{
    String hName;
    String pName;
    String regDate;
    int age;
    String pGender;
    String regId;

    void ReadValues()
    {
            Scanner read = new Scanner(System.in);
            System.out.println("Enter Hospital Name: ");
            hName = read.nextLine();

            System.out.println("Enter Patient Name: ");
            pName = read.nextLine();

            System.out.println("Enter Registration Date: ");
            regDate = read.nextLine();

            System.out.println("Enter Patient Age: ");
            age = read.nextInt();
            read.nextLine();

            System.out.println("Enter Gender: ");
            pGender = read.nextLine();

            System.out.println("Enter Patient Registration ID: ");
            regId = read.nextLine();
    }

    void DisplayCard()
    {
            System.out.println("------------------------------------");
            System.out.println("XXX HOSPITAL NAME: " + hName + " XXX");
            System.out.println("------------------------------------");
            System.out.println("Patient Name: " + pName);
            System.out.println("------------------------------------");
            System.out.println("Age: " + age);
            System.out.println("-----------------------------------");
            System.out.println("Gender: " + pGender);
            System.out.println("-----------------------------------");
            System.out.println("Patient ID: " + regId);
            System.out.println("-----------------------------------");
            System.out.println("Registration Date: " + regDate);
            System.out.println("-----------------------------------");
    }
}

class RegCard_Main
{
    public static void main(String[] args)
    {
            RegCard r = new RegCard();
            r.ReadValues();
            r.DisplayCard();
    }
}
```



```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

E:\Manoj>javac RegCard_Main.java

E:\Manoj>java RegCard_Main
Enter Hospital Name:
Govt Hospital, Sirsi
Enter Patient Name:
Shekhar
Enter Registration Date:
15/01/2025
Enter Patient Age:
25
Enter Gender:
Male
Enter Patient Registration ID:
P202545847
------------------------------------
XXX HOSPITAL NAME: Govt Hospital, Sirsi XXX
------------------------------------
Patient Name: Shekhar
------------------------------------
Age: 25
------------------------------------
Gender: Male
------------------------------------
Patient ID: P202545847
------------------------------------
Registration Date: 15/01/2025
------------------------------------

E:\Manoj>
```

Experiment 3:
Identify and resolve issues in the given code snippet

```
class Calculate
{
    void cube(int x)
    {
            int result
            result = x*x*x;
    }
    void display()
    {
            System.out.println("Cube of a no: " + result)
    }

    public static void main(String args[])
    {
            cube(5);
            display();
    }
}
```