INTRODUCTION TO JAVA

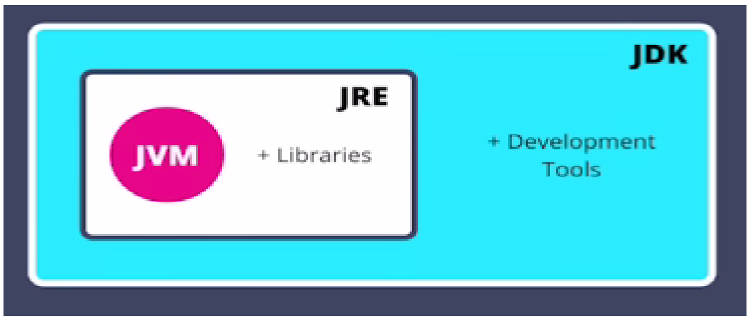
What is *Pseudo Code* ?

- Pseudocode is a description of an algorithm or program that uses natural language instead of a programming language.
- It's a blueprint for writing code, and it's often used as a rough draft.
- Basically converting programming language into English language that is understand by the user.

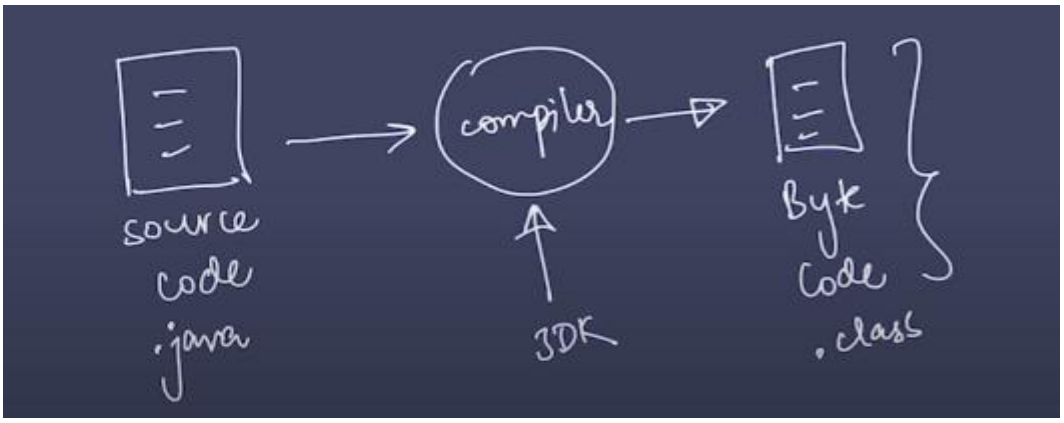
How my java Code is Running?

There are two steps from which my code is running

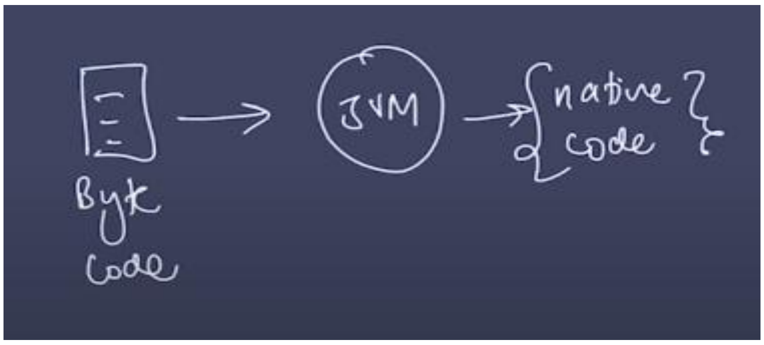
1. **Compilation**
2. **Execution**



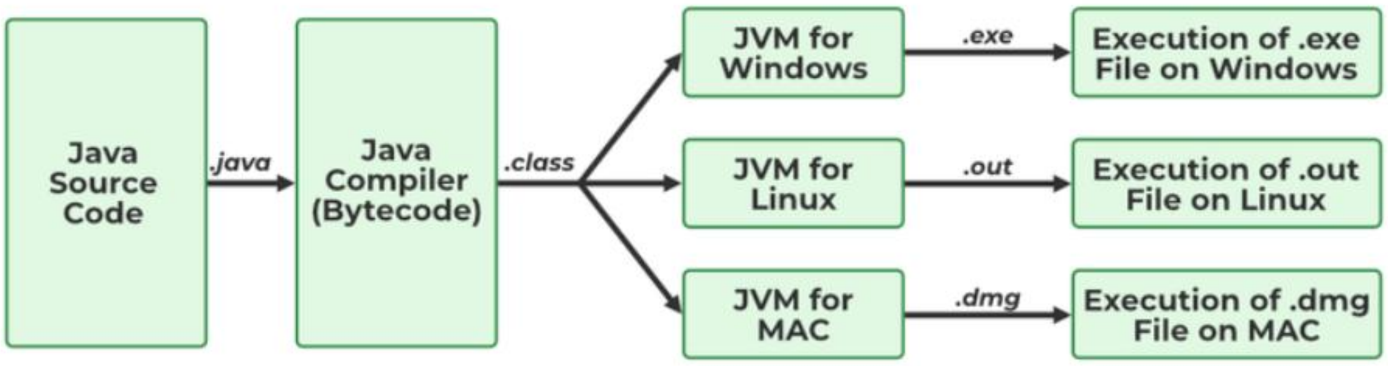
What happens in the compilation stage?



What happens in the Execution stage?



Together



What does JVM stands for what it is?

- JVM stands for Java Virtual Machine it is a Java interpreter.
- It is responsible for loading, verifying, and executing the bytecode created in Java.

What does JRE stand for and what it is?

- The Java Runtime Environment (JRE) is software that allows Java programs to run on a computer.
- It's a collection of tools that includes the Java Virtual Machine (JVM), Java class libraries, and the Java class loader.
- The JRE act as a translator between Java programs and the operating system, ensuring compatibility.

What does JDK stands for and what it does?

- JDK (Java Development Kit) stands for Java Development Kit.
- It is a software development kit used to develop Java applications.

Difference between JDK, JRE, and JVM

- JDK: Java Development Kit is a software development environment used for developing and Executing Java applications and applets.

- JRE: JRE stands for Java Runtime Environment and it provides an environment to run only the Java program onto the system. (not to deploy)

- JVM: JVM stands for Java Virtual Machine and is responsible for executing the Java program. (also called as interpreter)

BASIC FUNDAMENTALS IN JAVA

- **OUTPUT**
- **VARIABLES**
- **DATA TYPES**
- **INPUT**

Input/output

Syntax :- System.out.print(" ");

1. **print :- system.out.print("….");** the line continues
2. **println :- system.out.println("…");** the 2nd output goes to nextline
3. **"\n":-** goes to next line

VARIABLES

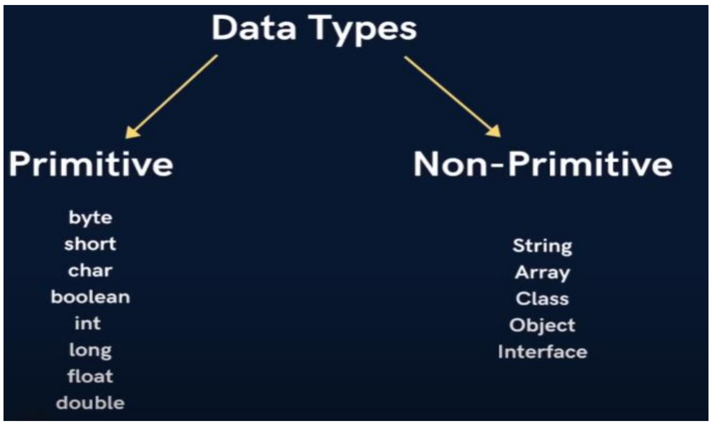
A variable is a named location in computer memory that stores data that can change.

How variables are stored in the computer memory?

1. Variables are stored in memory cells built from transistors and other components on an integrated circuit.
2. Variables are assigned a memory address when they are created.
3. The memory address is the location of the variable in the computer's memory.
4. The value of a variable is stored in the memory address.

DATATYPES

Since JAVA is a **Typed Language** we have to mention what is the type of data we have written to get him the proper space in the memory



- JAVA does not follow the BODMAS rule.
- In Java calculation happens from left to right.

INPUT

Eg code

package Input_03; → Declares that `Question.java` belongs to the `Input_03` package, which organizes classes and prevents naming conflicts; the file must be inside the `Input_03` folder for correct compilation.

import java.util.*; → Imports the `java.util` package, which includes the `Scanner` class used for taking user input.

public class Question { → Declares a public class `Question`, which must match the filename `Question.java`.

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

- This is the entry point of the Java program.
- `public` → Allows access from anywhere.
- `static` → Belongs to the class, doesn't need an object to be called.
- `void` → No return value.
- `String[] args` → Command-line arguments (not used here).

```
    Scanner sc = new Scanner(System.in);
```

- `Scanner sc` → Creates a `Scanner` object named `sc`.
- `new Scanner(System.in)` → Reads input from the keyboard.

```
    System.out.println("Enter two numbers : ");

    int a = sc.nextInt();

    int b = sc.nextInt();
```

- `sc.nextInt();` → Reads an integer input from the user and stores it in `a`.
- `sc.nextInt();` → Reads another integer input and stores it in `b`.

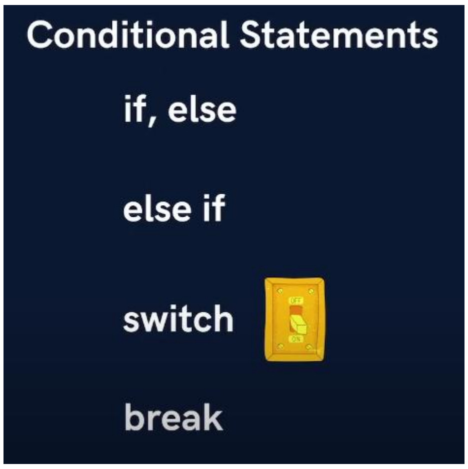
```
    int sum = a + b;

    System.out.println("Sum : " + sum);

    sc.close();
```
→ Closes the `Scanner` to free up system resources.

```
  }

}
```

CONDITIONAL STATEMENTS

Conditional statements control the flow of a program based on certain conditions (true or false).

The main Conditional Statements are



Conditional Statements

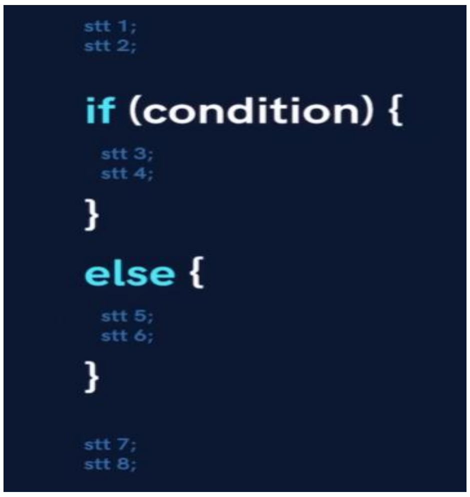
if, else

else if

switch

break

1. If, Else



```
stt 1;
stt 2;

if (condition) {
  stt 3;
  stt 4;
}
else {
  stt 5;
  stt 6;
}

stt 7;
stt 8;
```

- Executes one block if the condition is `true`, otherwise executes another block.

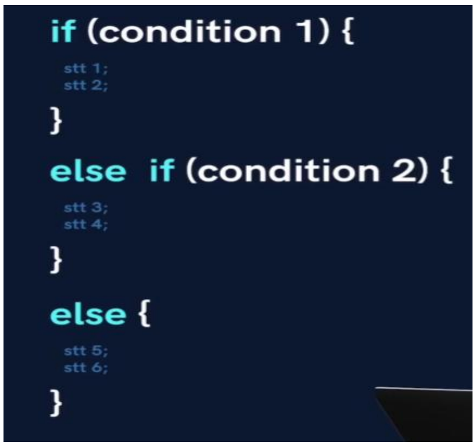
```
if (age >= 18) {
   System.out.println("You can vote.");
} else {
   System.out.println("You cannot vote.");
}
```

2. else-if

- Checks multiple conditions sequentially.

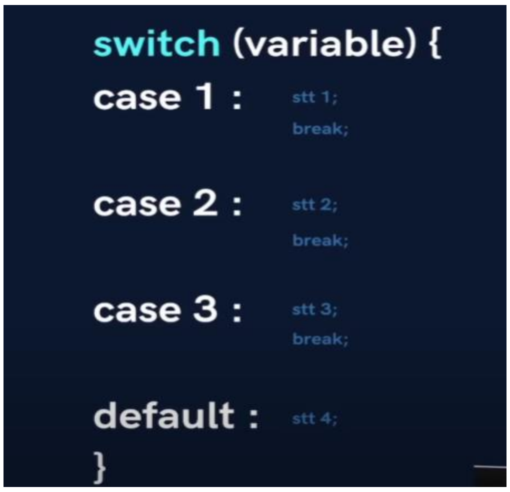
-
  ```
  if (marks >= 90) {
     System.out.println("Grade: A");
  } else if (marks >= 75) {
     System.out.println("Grade: B");
  } else {
  ```

```
    System.out.println("Grade: C");
  }
```

- 

```
if (condition 1) {
  stt 1;
  stt 2;
}
else  if (condition 2) {
  stt 3;
  stt 4;
}
else {
  stt 5;
  stt 6;
}
```

3. switch Statement



```
switch (variable) {
case 1 :      stt 1;
              break;

case 2 :      stt 2;
              break;

case 3 :      stt 3;
              break;

default :     stt 4;
}
```

- Used when multiple conditions depend on a single variable.

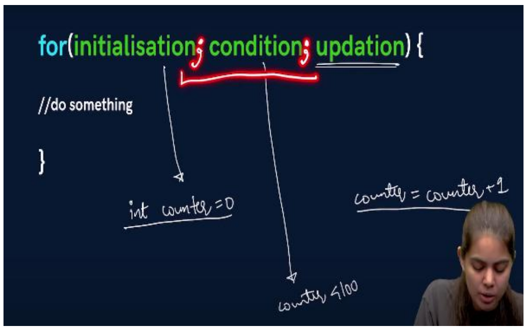
-
  ```
  int day = 3;
  switch (day) {
     case 1: System.out.println("Monday"); break;
     case 2: System.out.println("Tuesday"); break;
     case 3: System.out.println("Wednesday"); break;
     default: System.out.println("Invalid day");
  }
  ```

LOOPS

- A loop is used for executing a block of statements repeatedly until a particular condition is satisfied.
- A loop consists of an initialization statement, a test condition and an increment statement.)

<u>Types of Loops</u> :

1. For loop

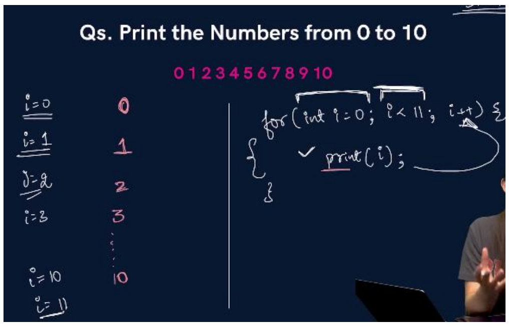


```
for(initialisation; condition; updation) {
//do something
}
```

int counter=0

counter <100

counter = counter+1



Qs. Print the Numbers from 0 to 10

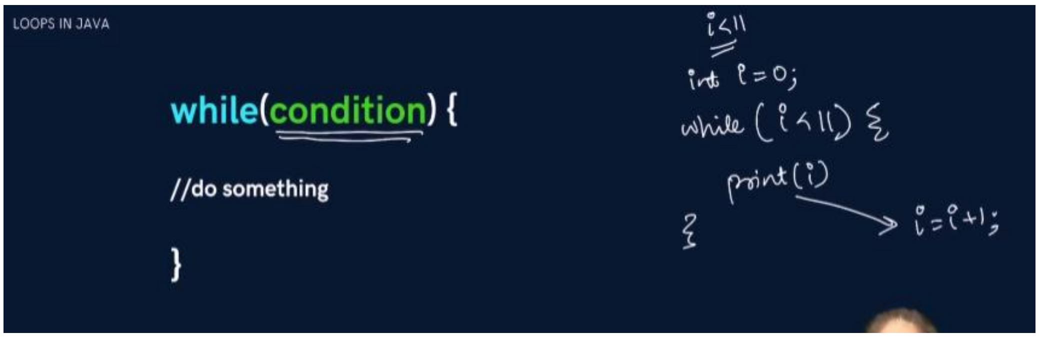
0 1 2 3 4 5 6 7 8 9 10

i=0	0
i=1	1
i=2	2
i=3	3
i=10	10
i=11	

```
for (int i=0; i<11; i++) {
    ✓ print(i);
}
```

- Runs a block of code a specific number of times.
- for (int i = 1; i <= 5; i++) {
    System.out.println("Iteration: " + i);
  }

2. While loop



LOOPS IN JAVA

```
while(condition) {

//do something

}
```

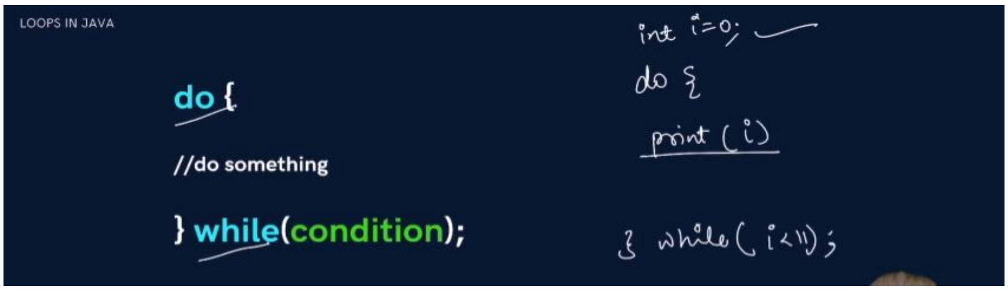
i<11

```
int i=0;
while (i<11) {
    print(i)
}
```

i=i+1;

- Runs a block of code **while** a condition is true.
  int i = 1;
  while (i <= 5) {
    System.out.println("Iteration: " + i);
    i++;
  }

3. Do while loop



LOOPS IN JAVA

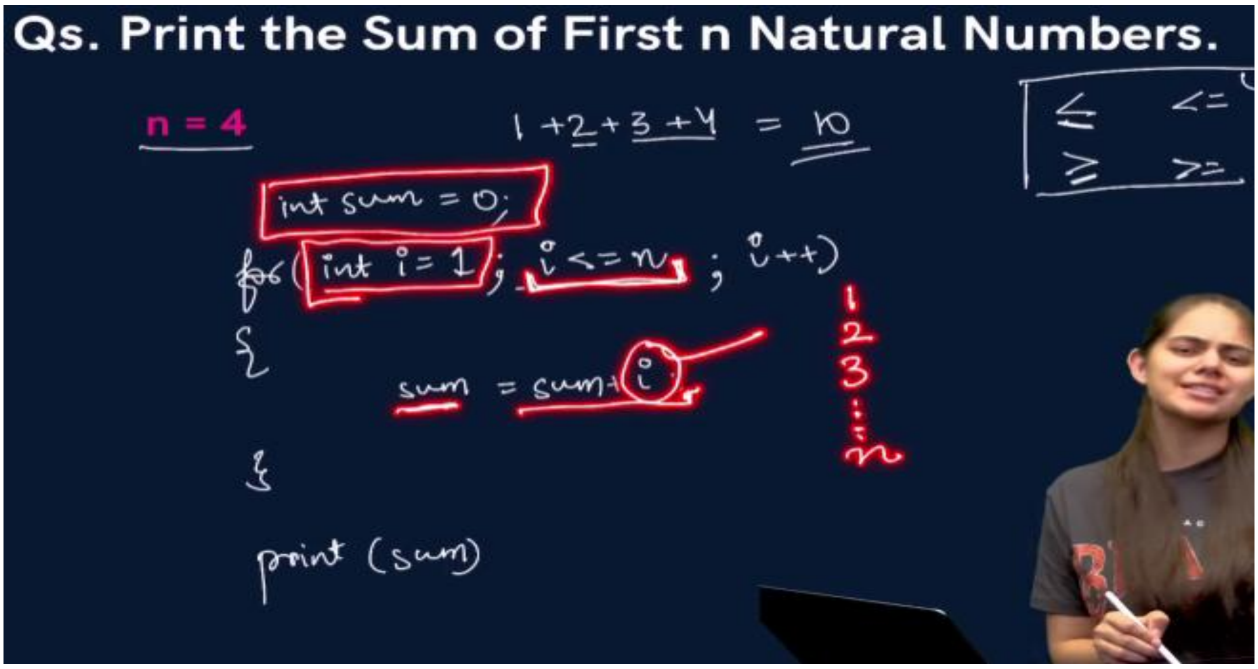
```
do {

//do something

} while(condition);
```

```
int i=0;
do {
    print (i)
} while(i<11);
```

- Runs the code **at least once**, then repeats **while** the condition is true.

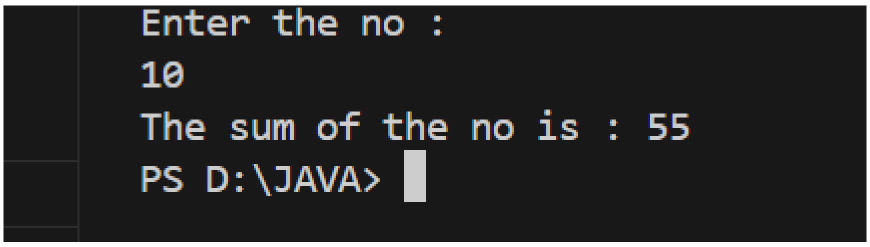
-
  ```
  int i = 1;
  do {
     System.out.println("Iteration: " + i);
     i++;
  } while (i <= 5);
  ```

<u>Egs on Loops</u>

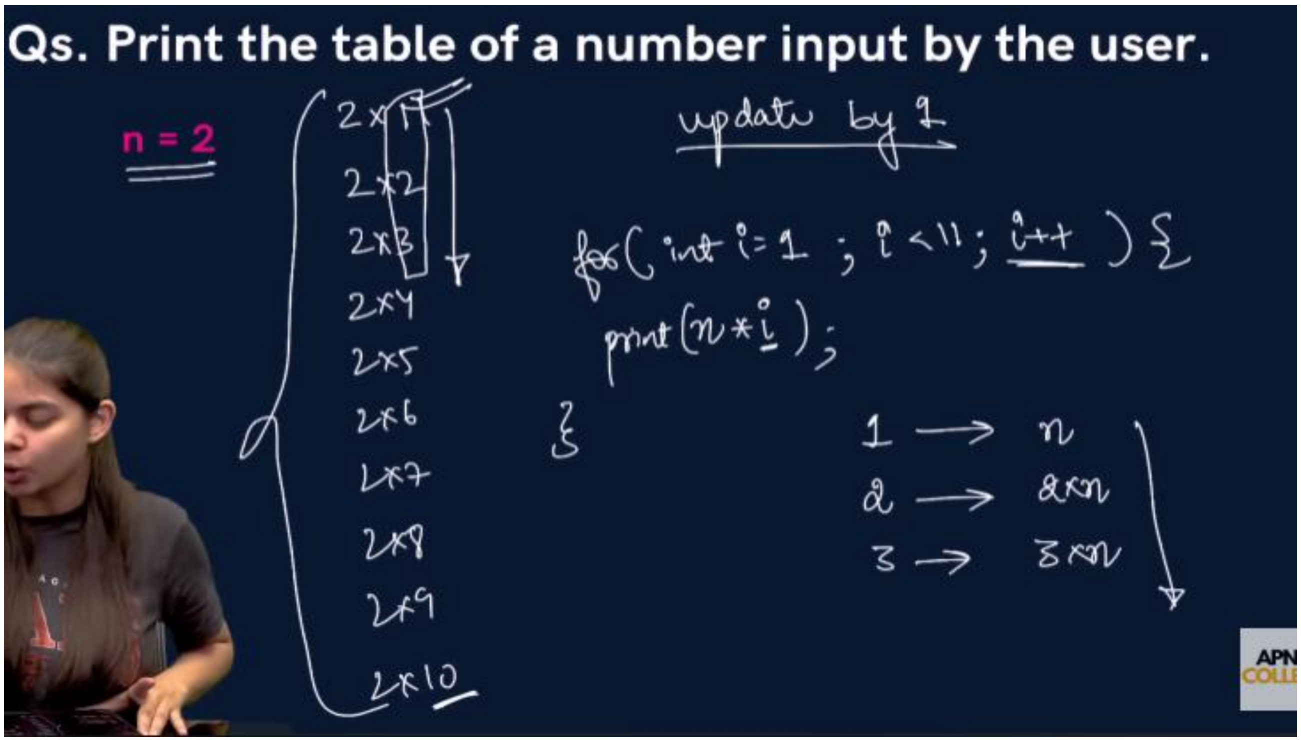


Qs. Print the Sum of First n Natural Numbers.

```java
package Loops;

import java.util.*;

public class eg {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the no :");
        int n = sc.nextInt();
        int sum = 0;

        for (int i = 1; i <= n; i++) {
            sum = sum + i;
        }
        System.out.println("The sum of the no is : " + sum);
        sc.close();
    }
}
```

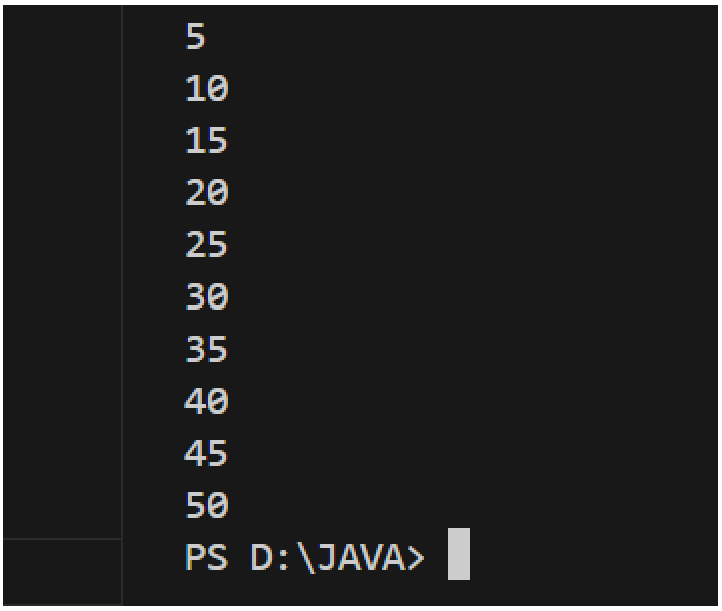


```
Enter the no :
10
The sum of the no is : 55
PS D:\JAVA>
```



Qs. Print the table of a number input by the user.

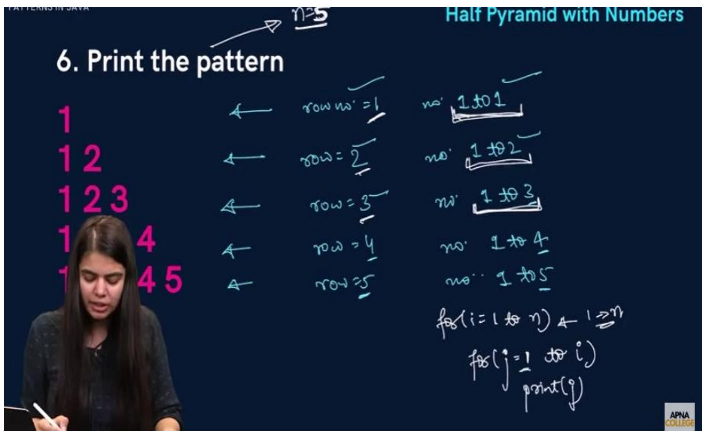
```
package Loops;

import java.util.Scanner;

public class eg2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the no :");
        int n = sc.nextInt();

        for (int i = 1; i < 11; i++) {
            System.out.println(i * n);
        }
        sc.close();
    }
}
```

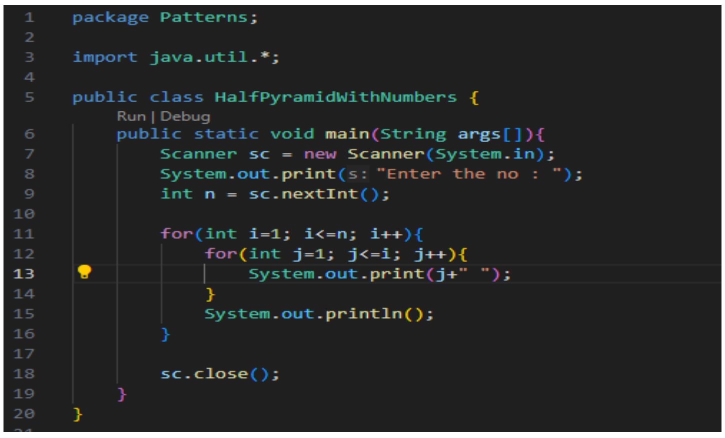


```
5
10
15
20
25
30
35
40
45
50
PS D:\JAVA>
```

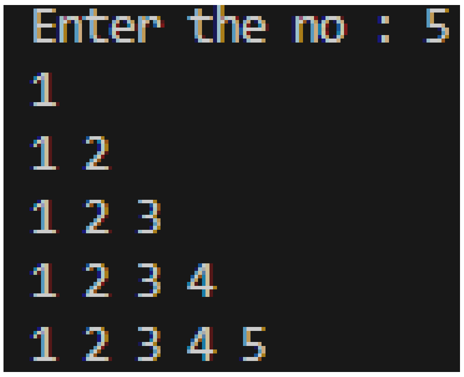
PATTERNS

Half Pyramind with Numbers



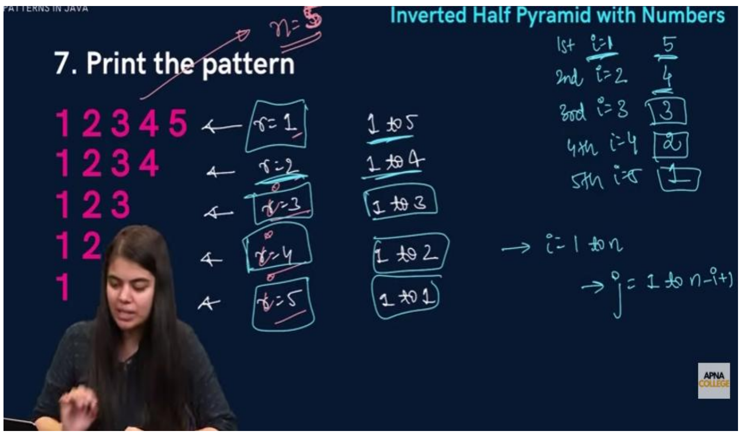


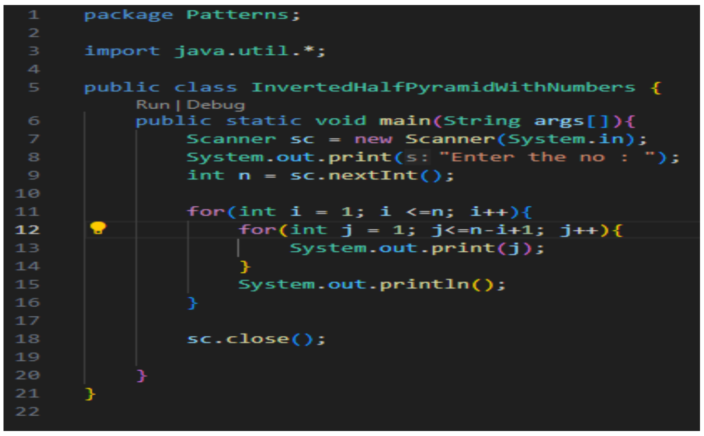
```
package Patterns;

import java.util.*;

public class HalfPyramidWithNumbers {
    Run | Debug
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter the no : ");
        int n = sc.nextInt();

        for(int i=1; i<=n; i++){
            for(int j=1; j<=i; j++){
                System.out.print(j+" ");
            }
            System.out.println();
        }

        sc.close();
    }
}
```



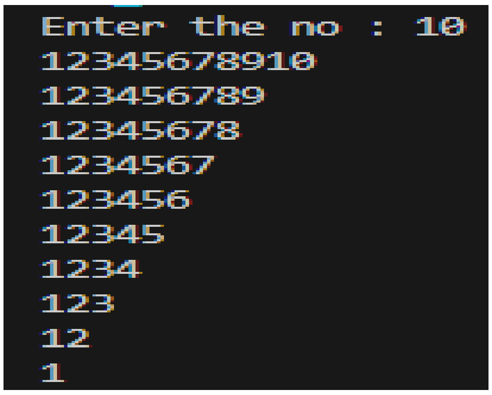
```
Enter the no : 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Inverted Half Pyramind with Numbers





```
package Patterns;

import java.util.*;

public class InvertedHalfPyramidWithNumbers {
    Run | Debug
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter the no : ");
        int n = sc.nextInt();

        for(int i = 1; i <=n; i++){
            for(int j = 1; j<=n-i+1; j++){
                System.out.print(j);
            }
            System.out.println();
        }

        sc.close();

    }
}
```



```
Enter the no : 10
12345678910
123456789
12345678
1234567
123456
12345
1234
123
12
1
```