

# Java Placement Course (DSA) notes

Chaitanya Shahare

## Contents

<b>1</b>	<b>Introduction to Java Language</b>	<b>4</b>
1.1	Set of Instructions . . . . .	4
1.2	Flowchart . . . . .	4
1.3	Pseudocode . . . . .	4
1.4	Java Class 1 . . . . .	5
1.4.1	Installation . . . . .	5
1.4.2	First Code . . . . .	5
1.4.3	How is code running? . . . . .	5
1.4.4	Code Components . . . . .	6
<b>2</b>	<b>Variables in Java</b>	<b>7</b>
2.1	Output . . . . .	7
2.1.1	Boilerplate code . . . . .	7
2.1.2	Q. Print the pattern . . . . .	7
2.2	Variables . . . . .	8
2.3	Data Type . . . . .	8
2.3.1	Types of Datatypes . . . . .	8
2.3.2	Data Type sizes . . . . .	9
2.4	Inputs in Java . . . . .	10
2.5	Q. Take 2 variables 'a' & 'b' and print their sum. . . . .	10
<b>3</b>	<b>Conditional Statements</b>	<b>11</b>
3.1	if, else . . . . .	11
3.1.1	Syntax . . . . .	11
3.1.2	Q. Write a program to identify if a person is an adult. . . . .	11
3.1.3	Q. Write a program to check if a number is odd or even. . . . .	11
3.2	else if . . . . .	12
3.2.1	Q. Write a program to know if a is greater of lesser than b. . . . .	12
3.3	Switch . . . . .	12
3.3.1	Syntax . . . . .	12
3.3.2	Q. Using switch write a program to greet in different languages . . . . .	12
3.3.3	Q. Make a calculator . . . . .	13
3.3.4	Q. Ask the user to enter the number of the month & print the name of the month. . . . .	13
<b>4</b>	<b>Loops</b>	<b>14</b>
4.1	For Loop . . . . .	14
4.1.1	Syntax . . . . .	14
4.1.2	Q. Print the number from 0 to 10 using for loop . . . . .	14
4.2	While Loop . . . . .	14
4.2.1	Syntax . . . . .	14
4.2.2	Q. Print the number from 0 to 10 using while loop . . . . .	15
4.3	Do While Loop . . . . .	15

4.3.1	Syntax . . . . .	15
4.3.2	Q. Print the number from 0 to 10 using do while loop . . . . .	15
4.4	Questions . . . . .	15
4.4.1	Q. Print the sum of first $n$ natural numbers. . . . .	15
4.4.2	Q. Print the table if a number input by the user. . . . .	16
4.4.3	Q. Print all even numbers till $n$ . . . . .	16
4.4.4	Q. Make a menu driven program. The user can enter 2 numbers, either 1 or 0. . . . .	16
<b>5</b>	<b>Basic Pattern Questions</b>	<b>17</b>
5.1	Nested Loops . . . . .	17
5.2	Q. Print the solid rectangle pattern . . . . .	17
5.3	Q. Print the hollow rectangle pattern . . . . .	18
5.4	Q. Print the half pyramid pattern . . . . .	18
5.5	Q. Print the inverted half pyramid pattern . . . . .	19
5.6	Q. Print the inverted half pyramid pattern (rotated by 180 deg) . . . . .	20
5.7	Q. Print the half pyramid with numbers pattern . . . . .	20
5.8	Q. Print the Inverted half pyramid with numbers pattern . . . . .	21
5.9	Q. Print the Floyd's triangle pattern . . . . .	22
5.10	Q. Print the 0-1 triangle pattern . . . . .	22
<b>6</b>	<b>Advanced Pattern Questions</b>	<b>24</b>
6.1	Q. Print the butterfly Patterns . . . . .	24
6.2	Q. Print the solid rhombus Patterns . . . . .	25
6.3	Q. Print the number pyramid pattern . . . . .	26
6.4	Q. Print a palindrome number pyramid pattern . . . . .	26
6.5	Q. Print the diamond pattern . . . . .	27
6.6	Print a hollow butterfly . . . . .	28
6.7	Print a hollow rhombus . . . . .	29
6.8	Print Pascal's triangle . . . . .	29
6.9	Print Inverted half pyramid pattern . . . . .	30
<b>7</b>	<b>Functions &amp; Methods</b>	<b>31</b>
7.1	Syntax . . . . .	31
7.2	Q. Print a given name in a function . . . . .	31
7.3	What happens in memory? . . . . .	31
7.4	Q. Make a function to add 2 numbers and return the sum . . . . .	32
7.5	Q. Make a function to multiply 2 numbers and return the product . . . . .	32
7.6	Q. Find a factorial of a number . . . . .	32
7.7	Difference between functions & methods . . . . .	33
<b>8</b>	<b>Functions practice questions</b>	<b>34</b>
8.1	Enter 3 numbers from the user & make a function to print their average. . . . .	34
8.2	Write a function to print the sum of all odd numbers from 1 to $n$ . . . . .	34
8.3	Write a function which takes in 2 numbers and returns the greater of those two. . . . .	34
8.4	Write a function that takes in the radius as input and returns the circumference of a circle. . . . .	34
8.5	Write a function that takes in age as input and returns if that person is eligible to vote or not. A person of age $> 18$ is eligible to vote. . . . .	34
8.6	Write an infinite loop using do while condition. . . . .	34
8.7	Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered. . . . .	34
8.8	Two numbers are entered by the user, $x$ and $n$ . Write a function to find the value of one number raised to the power of another i.e. $x^n$ . . . . .	34
8.9	Write a function that calculates the Greatest Common Divisor of 2 numbers. (BONUS) . . . . .	34
8.10	Write a program to print Fibonacci series of $n$ terms where $n$ is input by user: . . . . .	34

<b>9</b>	<b>Basics of Time &amp; Space Complexity</b>	<b>35</b>
9.1	Time Complexity . . . . .	35
9.1.1	Example . . . . .	35
9.1.2	Types of time complexity . . . . .	35
9.1.3	Example . . . . .	35
9.1.4	Comparing Time Complexities . . . . .	36
9.2	Space Complexity . . . . .	36

# 1 Introduction to Java Language

## 1.1 Set of Instructions

- Flowchart
- Psudocode

## 1.2 Flowchart

### Flowchart

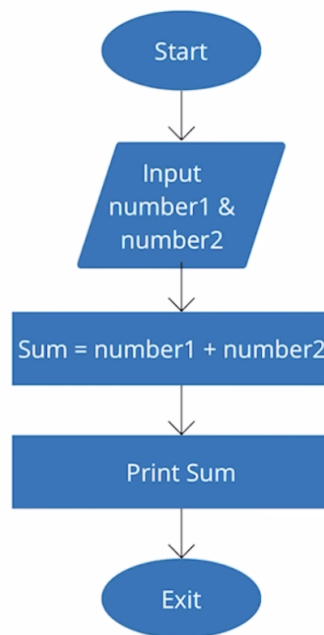


Figure 1: Flowchart

## 1.3 Psudocode

1. Start
2. Input 2 number
3. Calculate  $\text{Sum} = \text{number1} + \text{number2}$
4. Print Sum
5. Exit

## 1.4 Java Class 1

### 1.4.1 Installation

1. Java Development Kit (JDK)
2. Code Editor / IDE
  - VS Code
  - IntelliJ
  - Eclipse

### 1.4.2 First Code

- Extension -> .java

#### 1.4.2.1 Hello World

```
class FirstClass {  
    public static void main(String args[]) {  
        System.out.println("Hello World");  
    }  
}
```

### 1.4.3 How is code running?



Figure 2: Java Development Kit (JDK)

#### 1. Compilation

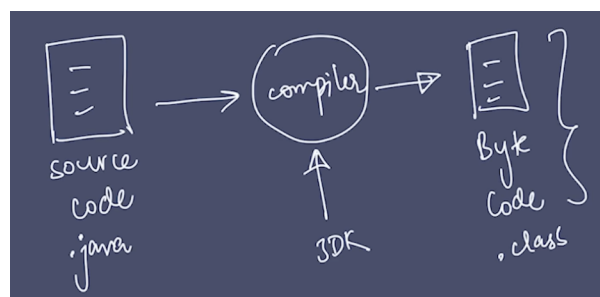


Figure 3: Java compilation

#### 2. Execution

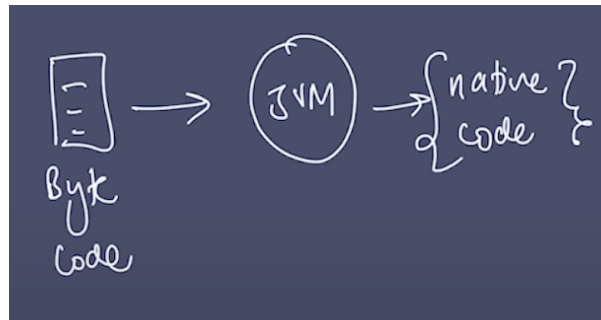


Figure 4: Java Execution

#### 1.4.4 Code Components

##### 1.4.4.1 Function

```
void main(){  
  
}
```

##### 1.4.4.2 Class

```
class Main{  
    void main() {  
  
    }  
}
```

## 2 Variables in Java

### 2.1 Output

```
System.out.print("Hello World");
```

Hello world is the string which is printed.

- Use double quotes for strings

#### 2.1.1 Boilerplate code

```
package com.apnacollege;
```

```
public class Main{  
    public static void main(String[] args) {  
        // Output  
        System.out.print("Hello World");  
    }  
}
```

Here:

- System -> class
- print -> function

```
System.out.println("Hello world with java");
```

- print -> for output on the same line  
System.out.print("Hello World");
- println -> for output on the next line  
System.out.println("Hello world with java");
- "\n" ->  
System.out.print("Hello World\n");

#### 2.1.2 Q. Print the pattern

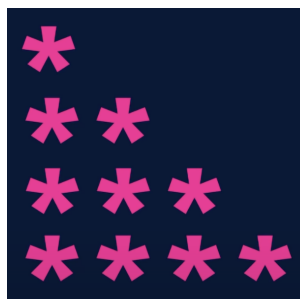


Figure 5: right triangle pattern

```

public class Main{
    public static void main(String[] args) {
        // Output
        System.out.println("*");
        System.out.println("**");
        System.out.println("***");
        System.out.println("****");
    }
}

```

## 2.2 Variables

Perimeter = 2 \* (a + b)

here,

- 2 -> constant
- a&b -> variable



Figure 6: Variables in memory

```

public class Main{
    public static void main(String[] args) {
        // Variables
        String name = "tony stark";
        int age = 48;
        double price = 23.25;
        int a = 25;
        int b = 1;

        b = 20;
        name = "ironman";
    }
}

```

## 2.3 Data Type

Java is a typed language. i.e; you need to tell the datatype.

### 2.3.1 Types of Datatypes

- Primitive
- Non-Primitive



Primitive	Non-Primitive
byte	String
short	Array
char	Class
boolean	Object
int	Interface
long	
float	
double	

### 2.3.2 Data Type sizes

Primitive	Size (in bytes)
byte	1
short	
char	2
boolean	1
int	4
long	8
float	4
double	8

*Above sizes are for a 64-bit System*

```

public class Main {
    public static void main(String[] args) {
        // Variables
        int a = 10;
        int b = 25;

        int sum = a + b;
        System.out.println(sum);

        int diff = b - a;
        System.out.println(diff);

        int mul = a * b;
        System.out.println(mul);
    }
}

```

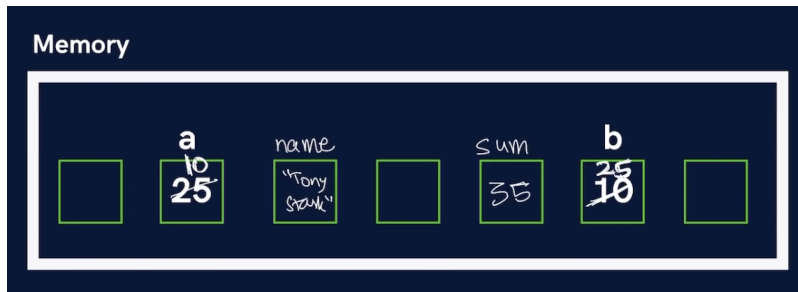


Figure 7: Memory allocation for the above program

## 2.4 Inputs in Java

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // Input
        Scanner sc = new Scanner(System.in);
        String name = sc.next(); // next() -> for next token ie; next word
        String name1 = sc.nextLine(); // nextLine() -> for taking a sentence as Input
        // Similarly
        // nextInt()
        // nextFloat()
        System.out.println(name);
    }
}
```

## 2.5 Q. Take 2 variables 'a' & 'b' and print their sum.

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        int sum = a + b;
        System.out.println(sum);
    }
}
```

## 3 Conditional Statements

Topics covered - if, else - else if - switch - break

### 3.1 if, else

#### 3.1.1 Syntax

```
if (condition){  
  
}  
else {  
  
}
```

---

Example

#### 3.1.2 Q. Write a program to identify if a person is an adult.

```
import java.util.*;  
  
public class Conditions {  
    public static void main(String args[]) {  
        Scanner sc = new Scanner(System.in);  
        int age = sc.nextInt();  
  
        if (age > 18) {  
            System.out.println("Adult");  
        } else {  
            System.out.println("Not Adult");  
        }  
    }  
}
```

#### 3.1.3 Q. Write a program to check if a number is odd or even.

```
import java.util.*;  
  
public class Conditions {  
    public static void main(String args[]) {  
        Scanner sc = new Scanner(System.in);  
        int x = sc.nextInt();  
  
        if (x % 2 == 0) {  
            System.out.println("Even");  
        } else {  
            System.out.println("Odd");  
        }  
    }  
}
```

## 3.2 else if

### 3.2.1 Q. Write a program to know if a is greater of lesser than b.

```
import java.util.*;

public class Conditions {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();

        if (a == b) {
            System.out.println("Equal");
        }
        else if (a > b) {
            System.out.println("a is greater than b");
        }
        else {
            System.out.println("a is lesser than b")
        }
    }
}
```

## 3.3 Switch

### 3.3.1 Syntax

```
switch (variable) {
    case 1:
        break;
    case 2:
        break;
    default:

}
```

### 3.3.2 Q. Using switch write a program to greet in different languages

```
import java.util.*;

public class Conditions {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int button = sc.nextInt();

        switch(button) {
            case 1: System.out.println("hello");
                break;
            case 2: System.out.println("namaste");
                break;
            case 3: System.out.println("bonjour");
                break;
            default: System.out.println("Invalid Button");
        }
    }
}
```

```
}  
}
```

### **3.3.3 Q. Make a calculator**

Make a Calculator. Take 2 numbers (a & b) from the user and an operation as follows :

- : + (Addition)  $a + b$
- : - (Subtraction)  $a - b$
- : \* (Multiplication)  $a * b$
- : / (Division)  $a / b$
- : % (Modulo or remainder)  $a \% b$

Calculate the result according to the operation given and display it to the user.

### **3.3.4 Q. Ask the user to enter the number of the month & print the name of the month.**

For eg - For '1' print 'January', '2' print 'February' & so on.

## 4 Loops

Topics covered - for Loop - while Loop - do while Loop

### 4.1 For Loop

#### 4.1.1 Syntax

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

- initialisation -> int counter = 0
- condition -> counter < 100
- updation -> counter = counter + 2

Example

```
public class Loops {  
    public static void main(String args[]) {  
        for (int counter = 0; counter < 100; counter += 1){  
            System.out.println("Hello world")  
        }  
    }  
}
```

Note: if any condition is not given an infinite loop will run

#### 4.1.2 Q. Print the number from 0 to 10 using for loop

```
public class Loops {  
    public static void main(String args[]) {  
        // counter++ => counter = counter + 1  
        for ( int i = 0; i < 11; i ++ ) [  
            System.out.println(i);  
        ]  
    }  
}
```

Dry Run => When analysing code without actually coding

### 4.2 While Loop

#### 4.2.1 Syntax

```
int i = 0; // initialisation  
  
while(condition){ // condition  
    // do something  
    i++; //updation  
}
```

#### 4.2.2 Q. Print the number from 0 to 10 using while loop

```
public class Loops {
    public static void main(String args[]) {
        int i = 0;
        while(i<11){
            System.out.println(i);
            i++;
        }
    }
}
```

### 4.3 Do While Loop

#### 4.3.1 Syntax

```
int i = 0; // initialisation

do {
    // do something
    i++; // updation
}while(condition) // condition
```

In do while loop, the loop is run at least once.

#### 4.3.2 Q. Print the number from 0 to 10 using do while loop

```
public class Loops {
    public static void main(String args[]) {
        int i = 0;
        do {
            System.out.println(i);
            i++;
        } while(i<11);
    }
}
```

---

### 4.4 Questions

#### 4.4.1 Q. Print the sum of first n natural numbers.

```
import java.util.*;

public class Loops {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int sum = 0;
        for(int i=0; i<=n; i++) {
            sum = sum + i;
        }

        System.out.println(sum);
    }
}
```

```
}  
}
```

#### 4.4.2 Q. Print the table if a number input by the user.

```
import java.util.*;  
  
public class Loops {  
    public static void main(String args[]) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
  
        for(int i=1; i<11; i++) {  
            System.out.println(i*n);  
        }  
    }  
}
```

#### 4.4.3 Q. Print all even numbers till n.

#### 4.4.4 Q. Make a menu driven program. The user can enter 2 numbers, either 1 or 0.

If the user enters 1 then keep taking input from the user for a student's marks(out of 100). If they enter 0 then stop. If he/ she scores : Marks  $\geq 90$  -> print "This is Good" 89  $\geq$  Marks  $\geq 60$  -> print "This is also Good" 59  $\geq$  Marks  $\geq 0$  -> print "This is Good as well" Because marks don't matter but our effort does. (Hint : use do-while loop but think & understand why)

---



## 5 Basic Pattern Questions

### 5.1 Nested Loops

```
for(..){  
    for(..){  
  
    }  
}
```

### 5.2 Q. Print the solid rectangle pattern



Figure 8: Solid rectangle pattern

```
import java.util.*;  
  
class Patterns {  
    public static void main(String args[]) {  
        int n = 4;  
        int m = 5;  
  
        // inner loop  
        for(int i=1; i<=n; i++) {  
            // inner loop  
            for (int j = 1; j <= m; j++) {  
                System.out.print("*");  
            }  
            System.out.println();  
        }  
    }  
}
```

### 5.3 Q. Print the hollow rectangle pattern

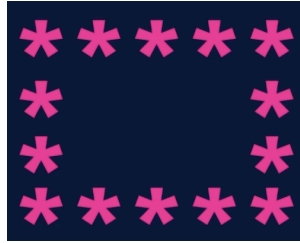


Figure 9: Hollow rectangle pattern

```
import java.util.*;

public class patterns_hollow_rectangle {
    public static void main(String[] args) {
        int n = 4;
        int m = 5;

        // Outer loop
        for (int i = 1; i <= n; i++) {
            // Inner loop
            for (int j = 1; j <= m; j++) {
                // cell -> (i,j)
                if (i == 1 || j == 1 || i == n || j == m) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```

### 5.4 Q. Print the half pyramid pattern

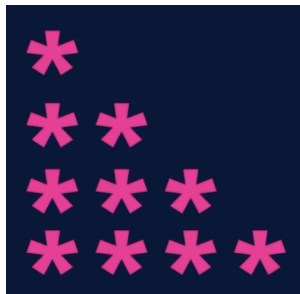


Figure 10: Half pyramid pattern

```
import java.util.*;
```

```

public class patterns_half_pyramid {
    public static void main(String[] args) {
        int n = 4;

        // Outer loop
        for ( int i = 1; i <= n; i++) {
            // Inner Loop
            for (int j = 1; j <= i; j++ ) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}

```

### 5.5 Q. Print the inverted half pyramid pattern



Figure 11: Inverted half pyramid pattern

```

import java.util.*;

public class patterns_half_pyramid {
    public static void main(String[] args) {
        int n = 4;

        // Outer loop
        for ( int i = n; i >= 1; i--) {
            // Inner Loop
            for (int j = 1; j <= i; j++ ) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}

```

5.6 Q. Print the inverted half pyramid pattern (rotated by 180 deg)

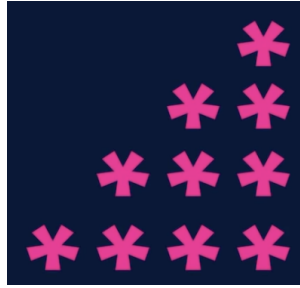


Figure 12: Inverted half pyramid rotated 180 deg

```
import java.util.*;

public class patterns_inverted_half_pyramid_180 {
    public static void main(String[] args) {
        int n = 4;

        // Outer loop
        for (int i = 1; i <= n; i++) {
            // Inner loop
            for (int j = 1; j <= n; j++) {
                if (j > n - i)
                    System.out.print("*");
                else
                    System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

5.7 Q. Print the half pyramid with numbers pattern



Figure 13: Half pyramid with numbers

```
import java.util.*;
```

```

public class patterns_half_pyramid_numbers {
    public static void main(String[] args) {
        int n = 5;

        // Outer loop
        for (int i = 1; i <= n; i++) {
            // Inner loop
            for (int j = 1; j <= i; j++) {
                System.out.print(j);
            }
            System.out.println();
        }
    }
}

```

## 5.8 Q. Print the Inverted half pyramid with numbers pattern

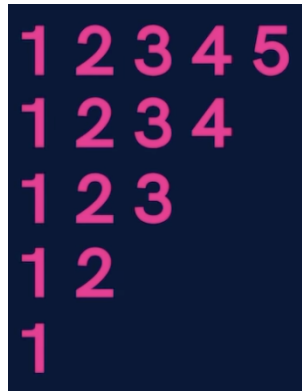


Figure 14: Inverted half pyramid with numbers

```

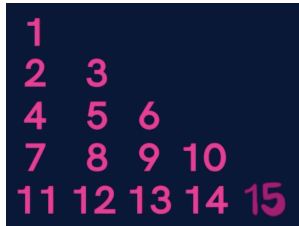
import java.util.*;

public class patterns_inverted_half_pyramid_numbers {
    public static void main(String[] args) {
        int n = 5;

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

```

### 5.9 Q. Print the Floyd's triangle pattern



A Floyd's triangle pattern consisting of 5 rows of numbers. The first row has 1 number (1), the second row has 2 numbers (2, 3), the third row has 3 numbers (4, 5, 6), the fourth row has 4 numbers (7, 8, 9, 10), and the fifth row has 5 numbers (11, 12, 13, 14, 15). The numbers are arranged in a right-angled triangular shape.

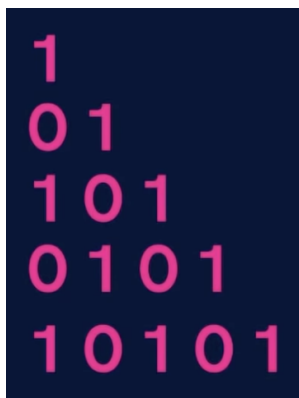
Figure 15: Floyd's triangle pattern

```
import java.util.*;

public class patterns_floyds_triangle {
    public static void main(String[] args) {
        int n = 5;
        int a = 1;

        // Outer loop
        for (int i = 1; i <= n; i++) {
            // Inner loop
            for (int j = 1; j <= i; j++) {
                System.out.print(a);
                a++;
            }
            System.out.println();
        }
    }
}
```

### 5.10 Q. Print the 0-1 triangle pattern



A 0-1 triangle pattern consisting of 5 rows of binary digits. The first row has 1 digit (1), the second row has 2 digits (0, 1), the third row has 3 digits (1, 0, 1), the fourth row has 4 digits (0, 1, 0, 1), and the fifth row has 5 digits (1, 0, 1, 0, 1). The digits are arranged in a right-angled triangular shape.

Figure 16: 0-1 triangle pattern

```
import java.util.*;

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

```
int n = 5;
int a = 1;

// Outer loop
for (int i = 1; i <= n; i++) {
    // Inner loop
    for (int j = 1; j <= i; j++) {
        int sum = i+j;
        if (sum % 2 == 0) { //even
            System.out.print("1 ");
        } else { // odd
            System.out.print("0 ");
        }
    }
    System.out.println();
}
}
```

---

## 6 Advanced Pattern Questions

### 6.1 Q. Print the butterfly Patterns

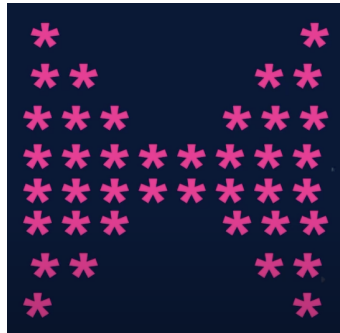


Figure 17: Butterfly pattern

```
import java.util.*;

public class patterns_butterfly {
    public static void main(String[] args) {
        int n = 4;

        //upper part
        for(int i=1; i<=n; i++) {
            for(int j=1; j<=i; j++) {
                System.out.print("*");
            }

            int spaces = 2 * (n-i);
            for(int j=1; j<=spaces; j++) {
                System.out.print(" ");
            }

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

        //lower part
        for(int i=n; i>=1; i--) {
            for(int j=1; j<=i; j++) {
                System.out.print("*");
            }

            int spaces = 2 * (n-i);
            for(int j=1; j<=spaces; j++) {
                System.out.print(" ");
            }
        }
    }
}
```



```

    }

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

```

## 6.2 Q. Print the solid rhombus Patterns

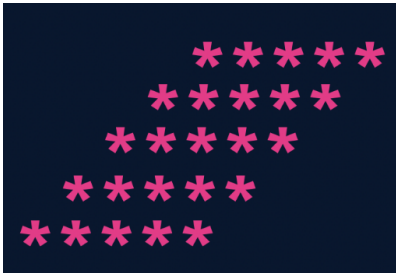


Figure 18: Solid rhombus pattern

```

import java.util.*;

public class patterns_solid_rhombus {
    public static void main(String[] args) {
        int n = 5;

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

            // stars
            for (int j = 1; j <= 5; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}

```

### 6.3 Q. Print the number pyramid pattern

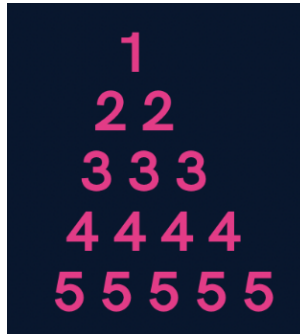


Figure 19: Number pyramid pattern

```
import java.util.*;

public class patterns_number_pyramid {
    public static void main(String[] args) {
        int n = 5;

        // Outer loop
        for (int i = 1; i <= n; i++) {
            // spaces
            for (int j = 1; j <= n-i; j++) {
                System.out.print(" ");
            }
            // numbers => print row no., row no. times
            for (int j = 1; j <= i; j++) {
                System.out.print(i + " ");
            }
            System.out.println();
        }
    }
}
```

### 6.4 Q. Print a palindrome number pyramid pattern

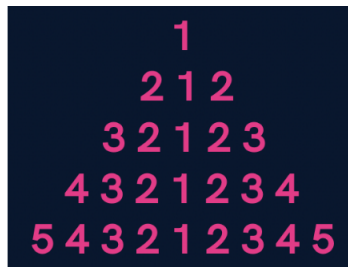


Figure 20: Palindrome number pyramid pattern

```
import java.util.*;

public class patterns_palindrome_pyramid {
```

```

public static void main(String[] args) {
    int n = 5;

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

        // 1st half numbers
        for (int j = i; j >= 1; j--) {
            System.out.print(j);
        }

        // 2nd half numbers
        for (int j = 2; j <= i; j++) {
            System.out.print(j);
        }
        System.out.println();
    }
}

```

## 6.5 Q. Print the diamond pattern

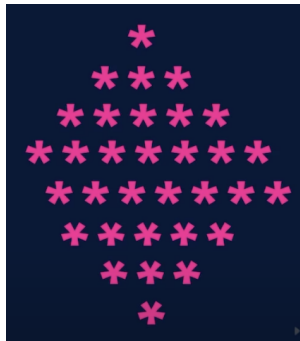


Figure 21: Diamond pattern

```

import java.util.*;

public class patterns_diamond {
    public static void main(String[] args) {
        int n = 4;

        // upper half
        for (int i = 1; i <= n; i++) {
            // spaces
            for (int j = 1; j <= n-i; j++) {
                System.out.print(" ");
            }

            // stars

```

```

    for (int j = 1; j <= 2*i-1; j++) {
        System.out.print("*");
    }
    System.out.println();
}
// lower half
for (int i = n; i >= 1; i--) {
    // spaces
    for (int j = 1; j <= n-i; j++) {
        System.out.print(" ");
    }

    // stars
    for (int j = 1; j <= 2*i-1; j++) {
        System.out.print("*");
    }
    System.out.println();
}
}
}

```

## 6.6 Print a hollow butterfly



Figure 22: Hollow butterfly pattern

## 6.7 Print a hollow rhomubus

```
*****
*      *
*      *
*      *
*      *
*****
```

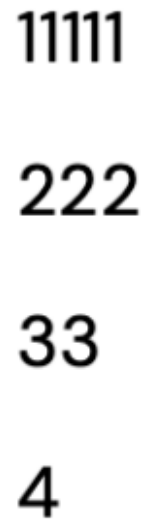
Figure 23: Hollow rhombus pattern

## 6.8 Print Pascal's triangle

```
1
11
121
1331
14641
```

Figure 24: Pascal's triangle

## 6.9 Print Inverted half pyramid pattern



1111  
222  
33  
4

The image displays an inverted half pyramid pattern. It consists of four rows of numbers. The first row has four '1's, the second row has three '2's, the third row has two '3's, and the fourth row has one '4'. The numbers are centered horizontally, creating a triangular shape pointing downwards.

Figure 25: Inverted half pyramid pattern

---

## 7 Functions & Methods

Functions is a block of code with takes input, performs some operations and returns output.

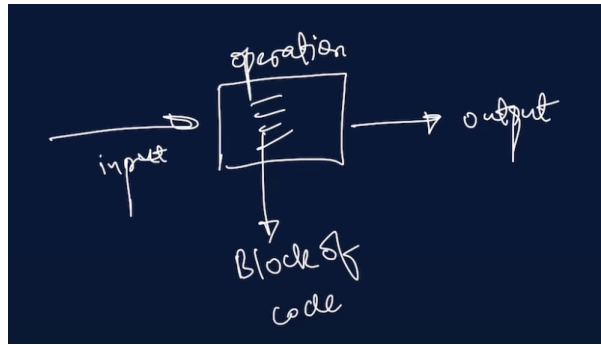


Figure 26: Function working

### 7.1 Syntax

```
returnType functionName(type arg1, type arg2 ..){
    // operations
}
```

- returnType -> int, float, String, ...
  - returnType void -> no return
- public static -> oops, for now prefix of every function
- type -> int, float, String, ...

### 7.2 Q. Print a given name in a function

```
import java.util.*;

public class Functions {
    public static void printMyName(String name) {
        System.out.println(name);
        return;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String name = sc.next();

        printMyName(name); // function is invoked
    }
}
```

### 7.3 What happens in memory?

- All functions are saved in memory in stack form.
  - a single unit in a stack is a **stack frame**.
- function no. ↑, stack size ↑

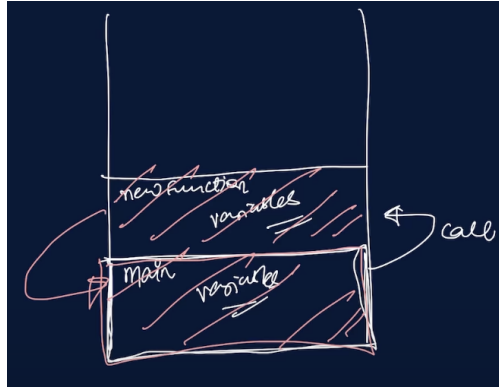


Figure 27: Functions in memory

- When main function is running is created in the memory
- When it invokes some other function that function is created in the memory and saved as a stack
- When the invoked function is executed it is removed from the memory
- And after the completion of the main function it is also removed
- Variables in a particular function are stored in the same stack frame as the function

More about memory in the OOPs Chapter

#### 7.4 Q. Make a function to add 2 numbers and return the sum

```
public static void calculateSum(int a, int b) {
    int sum = a + b;
    return sum;
}
```

#### 7.5 Q. Make a function to multiply 2 numbers and return the product

```
public static int calculateProduct(int a, int b) {
    return a * b;
}
```

#### 7.6 Q. Find a factorial of a number

```
public static void printFactorial(int n) {
    if(n<0){
        System.out.println("Invalid Number");
        return;
    }

    int factorial = 1;

    // loop
    for(int i = n; i >= 1; i--){
        factorial = factorial * i;
    }
}
```



## 7.7 Difference between functions & methods

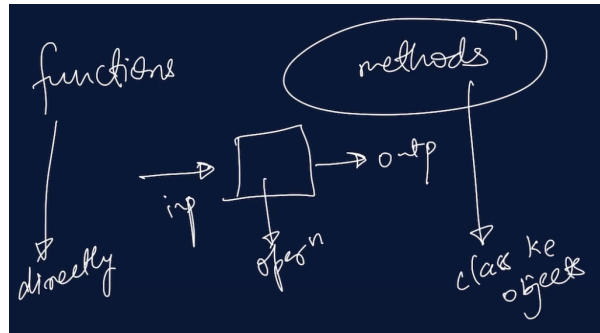


Figure 28: Difference b/w functions & methods

- we call functions directly
  - and methods through objects of class
-

## 8 Functions practice questions

[Link to the pdf document](#)

- 8.1 Enter 3 numbers from the user & make a function to print their average.
- 8.2 Write a function to print the sum of all odd numbers from 1 to n.
- 8.3 Write a function which takes in 2 numbers and returns the greater of those two.
- 8.4 Write a function that takes in the radius as input and returns the circumference of a circle.
- 8.5 Write a function that takes in age as input and returns if that person is eligible to vote or not. A person of age  $> 18$  is eligible to vote.
- 8.6 Write an infinite loop using do while condition.
- 8.7 Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.
- 8.8 Two numbers are entered by the user, x and n. Write a function to find the value of one number raised to the power of another i.e.  $x^n$ .
- 8.9 Write a function that calculates the Greatest Common Divisor of 2 numbers. (BONUS)
- 8.10 Write a program to print Fibonacci series of n terms where n is input by user:

0 1 1 2 3 5 8 13 21 . . . . In the Fibonacci series, a number is the sum of the previous 2 numbers that came before it. (BONUS)

---

## 9 Basics of Time & Space Complexity

### 9.1 Time Complexity

Relation between **Input Size** & **Running Time** (operations).

#### 9.1.1 Example

```
public static void main(String args[]){
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();

    for(int i = 0; i < n; i++) {
        System.out.println("hello");
    }
}
```

- input  $n \rightarrow$  time  $n$
- time complexity  $\propto$  input  $n$
- Linear relation

#### 9.1.2 Types of time complexity

1. Best case  $\rightarrow \Omega()$
  2. Average case  $\rightarrow \theta()$
  3. Worst case  $\rightarrow O()$  bigO
- We always assume worst case time complexity *i.e.*  $O()$

#### 9.1.3 Example

```
public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();

    for(int i = 0 ; i < n ; i++) {
        for(int j = 0; j < n; j++) {
            System.out.println("hello");
        }
    }
}
```

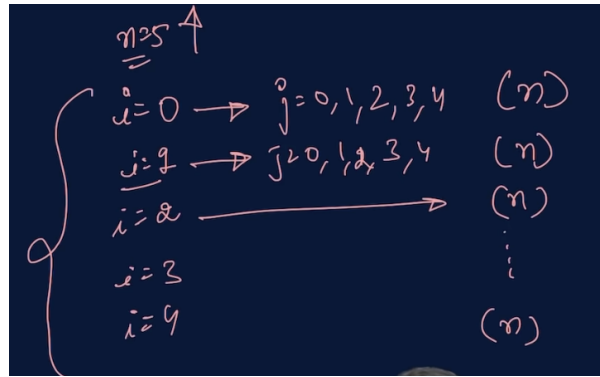


Figure 29: Total time the print operation is done

$$n \times n = n^2$$

worst case time complexity  $\rightarrow O(n^2)$

#### 9.1.4 Comparing Time Complexities

Compare	$O(n)$	$O(n^2)$	$O(n^3)$
n=1	1	1	12
n=2	2	4	8
n=3	3	9	27
n=10 <sup>5</sup>	10 <sup>5</sup>	10 <sup>10</sup>	10 <sup>30</sup>
	Best	2ndBest	Worst

## 9.2 Space Complexity

Space complexity depends on the space the program occupies in the memory.

- input int n  $\rightarrow$  space complexity constant
- Array  $\rightarrow$  space complexity depends on input