**Java - Introduction to Programming**

**Lecture 2**

**Variables & Data Types**

1. **Variables**

A variable is a container (storage area) used to hold data.

Each variable should be given a unique name (identifier).

package com.apnacollege;

public class Main {

public static void main(String[] args) {

// Variables

String name = "Aman";

int age = 30;

String neighbour = "Akku";

String friend = neighbour;

}

}

1. **Data Types**

Data types are declarations for variables. This determines the type and size of data associated with variables which is essential to know since different data types occupy different sizes of memory.

There are 2 types of Data Types :

* Primitive Data types : to store simple values
* Non-Primitive Data types : to store complex values

**Primitive Data Types**

These are the data types of fixed size.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type** | **Meaning** | **Size**  **(in Bytes)** | **Range** |
| byte | 2’s complement integer | 1 | -128 to 127 |
| short | 2’s complement integer | 2 | -32K to 32K |
| int | Integer numbers | 4 | -2B to 2B |
| long | 2’s complement integer  (larger values) | 8 | -9,223,372,036,854,775,808  to 9,223,372,036,854,775,807 |
| float | Floating-point | 4 | Upto 7 decimal digits |
| double | Double Floating-point | 8 | Upto 16 decimal digits |
| char | Character | 2 | a, b, c .. A, B, C ..  @, #, $ .. |
| bool | Boolean | 1 | True, false |

**Non-Primitive Data Types**

These are of variable size & are usually declared with a ‘new’ keyword.

Eg : String, Arrays

String name = new String("Aman");

int[] marks = new int[3];

marks[0] = 97;

marks[1] = 98;

marks[2] = 95;

1. **Constants**

A constant is a variable in Java which has a fixed value i.e. it cannot be assigned a different value once assigned.

package com.apnacollege;

public class Main {

public static void main(String[] args) {

// Constants

final float PI = 3.14F;

}

}

**Homework Problems**

1. **Try to declare meaningful variables of each type. Eg - a variable named age should be a numeric type (int or float) not byte.**
2. **Make a program that takes the radius of a circle as input, calculates its radius and area and prints it as output to the user.**
3. **Make a program that prints the table of a number that is input by the user.**

**(HINT - You will have to write 10 lines for this but as we proceed in the course you will be studying about ‘LOOPS’ that will simplify your work A LOT!)**

**Java - Introduction to Programming**

**Lecture 3**

1. **Conditional Statements ‘if-else’**

**The if block is used to specify the code to be executed if the condition specified  in if is true, the else block is executed otherwise.**

**int age = 30;**

**if(age > 18) {**

**System.*out*.println("This is an adult");**

**} else {**

**System.*out*.println("This is not an adult");**

**}**

1. **Conditional Statements ‘switch’**

**Switch case statements are a substitute for long if statements that compare a**

**variable to multiple values. After a match is found, it executes the**

**corresponding code of that value case.**

**The following example is to print days of the week:**

**int n = 1;**

**switch(n) {**

**case 1 :**

**System.*out*.println("Monday");**

**break;**

**case 2 :**

**System.*out*.println("Tuesday");**

**break;**

**case 3 :**

**System.*out*.println("Wednesday");**

**break;**

**case 4 :**

**System.*out*.println("Thursday");**

**break;**

**case 5:**

**System.*out*.println("Friday");**

**break;**

**case 6 :**

**System.*out*.println("Saturday");**

**break;**

**default :**

**System.*out*.println("Sunday");**

**}**

**Homework Problems**

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

**1 : + (Addition) a + b**

* **2 : - (Subtraction) a - b**
* **3 : \* (Multiplication) a \* b**
* **4 : / (Division) a / b**
* **5 : % (Modulo or remainder) a % b**

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

1. **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.**

**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.**

**For Loop**

**The syntax of the for loop is  :**

**for (initialization; condition; update) {**

**// body of-loop**

**}**

**for (int i=1; i<=20; i++) {**

**System.*out*.println(i);**

**}**

**While Loop**

**The syntax for while loop is :**

**while(condition) {**

**// body of the loop**

**}**

**int i = 0;**

**while(i<=20) {**

**System.*out*.println(i);**

**i++;**

**}**

**Do-While Loop**

**The syntax for the do-while loop is :**

**do {**

**// body of loop;**

**}**

**while (condition);**

**int i = 0;**

**do {**

**System.*out*.println(i);**

**i++;**

**} while(i<=20);**

**Homework Problems**

1. **Print all even numbers till n.**
2. **Run**

**for(; ;) {**

**System.out.println("Apna College");**

**}**

**loop on your system and analyze what happens. Try to think of the reason for the output produced.**

1. **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 >=90 -> print “This is Good”**

**89 >= Marks >= 60 -> print “This is also Good”**

**59 >= Marks >= 0 -> print “This is Good as well”**

**Because marks don’t matter but our effort does.**

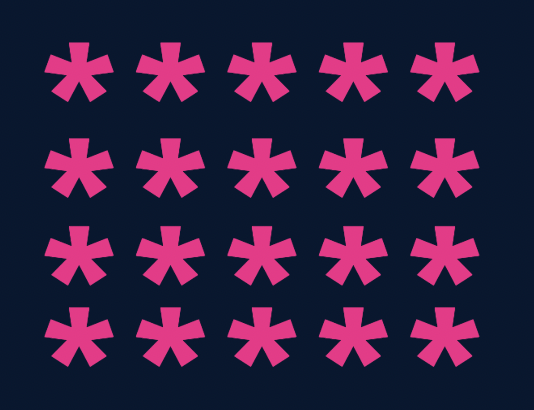
**(Hint : use do-while loop but think & understand why)**

**BONUS**

**Qs. Print if a number is prime or not (Input n from the user).**

**[In this problem you will learn how to check if a number is prime or not]**

**Patterns - Part 1**

****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 5;**

**int m = 4;**

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

**for(int j=0; j<m; j++) {**

**System.out.print("\*");**

**}**

**System.out.println();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 5;**

**int m = 4;**

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

**for(int j=0; j<m; j++) {**

**if(i == 0 || i == n-1 || j == 0 || j == m-1) {**

**System.out.print("\*");**

**} else {**

**System.out.print(" ");**

**}**

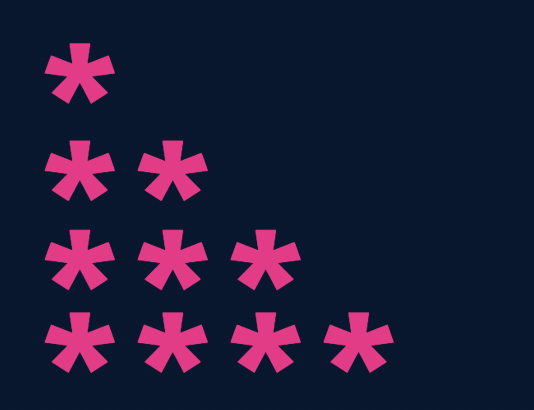
**}**

**System.out.println();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 4;**

**for(int i=1; i<=n; i++) {**

**for(int j=1; j<=i; j++) {**

**System.out.print("\*");**

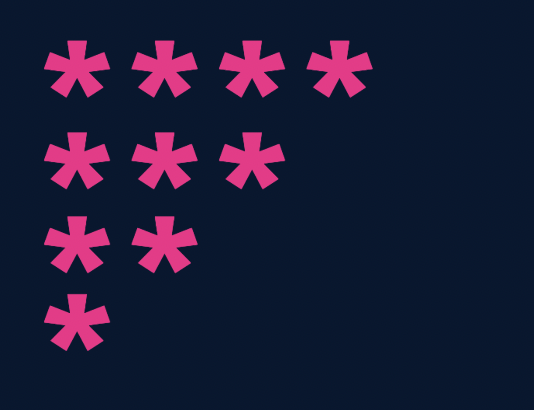
**}**

**System.out.println();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 4;**

**for(int i=n; i>=1; i--) {**

**for(int j=1; j<=i; j++) {**

**System.out.print("\*");**

**}**

**System.out.println();**

**}**

**}**

**}**

****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 4;**

**for(int i=n; i>=1; i--) {**

**for(int j=1; j<i; j++) {**

**System.out.print(" ");**

**}**

**for(int j=0; j<=n-i; j++) {**

**System.out.print("\*");**

**}**

**System.out.println();**

**}**

**}**

**}**

****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 5;**

**for(int i=1; i<=n; i++) {**

**for(int j=1; j<=i; j++) {**

**System.out.print(j);**

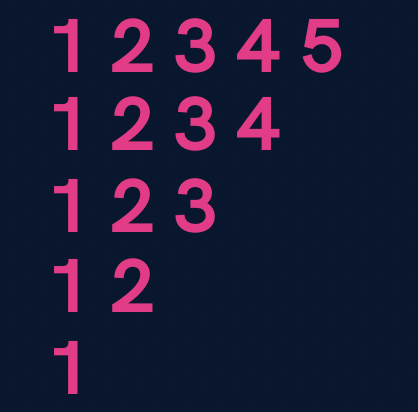
**}**

**System.out.println();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 5;**

**for(int i=n; i>=1; i--) {**

**for(int j=1; j<=i; j++) {**

**System.out.print(j);**

**}**

**System.out.println();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 5;**

**int number = 1;**

**for(int i=1; i<=n; i++) {**

**for(int j=1; j<=i; j++) {**

**System.out.print(number+" ");**

**number++;**

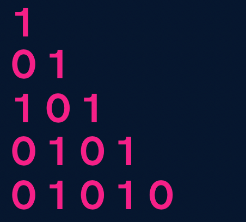
**}**

**System.out.println();**

**}**

**}**

**}**

****

**import java.util.\*;**

**public class Patterns {**

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

**int n = 5;**

**for(int i=1; i<=n; i++) {**

**for(int j=1; j<=i; j++) {**

**if((i+j) % 2 == 0) {**

**System.out.print(1+" ");**

**} else {**

**System.out.print(0+" ");**

**}**

**}**

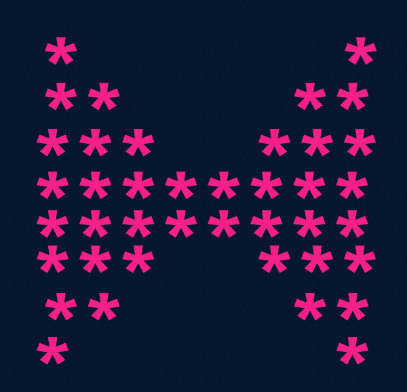
**System.out.println();**

**}**

**}**

**}**

**Patterns - Part 2**

****

**import java.util.\*;**

**public class Solutions {**

**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();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Solutions {**

**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<=n; j++) {**

**System.out.print("\*");**

**}**

**System.out.println();**

**}**

**}**

**}**

****

**import java.util.\*;**

**public class Solutions {**

**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(" ");**

**}**

**//numbers**

**for(int j=1; j<=i; j++) {**

**System.out.print(i+" ");**

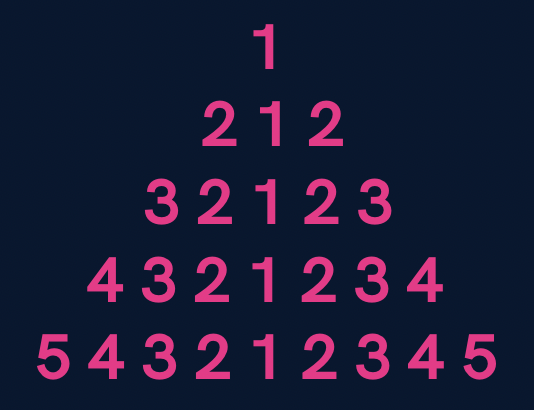
**}**

**System.out.println();**

**}**

**}**

**}**

1. ****

**import java.util.\*;**

**public class Solutions {**

**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(" ");**

**}**

**//first part**

**for(int j=i; j>=1; j--) {**

**System.out.print(j);**

**}**

**//second part**

**for(int j=2; j<=i; j++) {**

**System.out.print(j);**

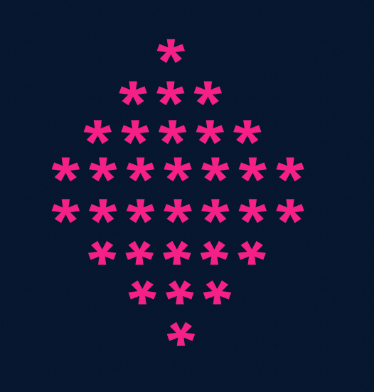
**}**

**System.out.println();**

**}**

**}**

**}**

****

**import java.util.\*;**

**public class Solutions {**

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

**int n = 5;**

**//upper part**

**for(int i=1; i<=n; i++) {**

**//spaces**

**for(int j=1; j<=n-i; j++) {**

**System.out.print(" ");**

**}**

**for(int j=1; j<=2\*i-1; j++) {**

**System.out.print("\*");**

**}**

**System.out.println();**

**}**

**//lower part**

**for(int i=n; i>=1; i--) {**

**//spaces**

**for(int j=1; j<=n-i; j++) {**

**System.out.print(" ");**

**}**

**for(int j=1; j<=2\*i-1; j++) {**

**System.out.print("\*");**

**}**

**System.out.println();**

**}**

**}**

**}**

**Homework Problems**

1. **Print a hollow Butterfly.**

1. **Print a hollow Rhombus.**

**\*\*\*\*\***

**\*      \***

**\*     \***

**\*     \***

**\*\*\*\*\***

1. **Print Pascal’s Triangle.**

**1**

**1 1**

**1 2 1**

**1 3 3 1**

**1 4 6 4 1**

1. **Print half Pyramid.**

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

1. **Print Inverted Half Pyramid.**

**11111**

**222**

**33**

**4**

**Java - Introduction to Programming**

**Lecture 7**

**Methods/Functions**

**A function is a block of code that performs a specific task.**

**Why are functions used?**

1. **If some functionality is performed at multiple places in software, then  rather than writing the same code, again and again, we create a function  and call it everywhere. This helps reduce code redundancy.**
2. **Functions make maintenance of code easy as we have to change at one  place if we make future changes to the functionality.**
3. **Functions make the code more readable and easy to understand.**

**The syntax for function declaration is :**

**return-type function\_name (parameter 1, parameter2,  …… parameter n){  //function\_body**

**}**

**return-type**

**The return type of a function is the data type of the variable that that function  returns.**

**For eg - If we write a function that adds 2 integers and returns their sum then  the return type of this function will be ‘int’ as we will return a sum that is an  integer value.**

**When a function does not return any value, in that case the return type of the  function is ‘void’.**

**function\_name**

**It is the unique name of that function.**

**It is always recommended to declare a function before it is used.**

**Parameters**

**A function can take some parameters as inputs. These parameters are specified  along with their data types.**

**For eg- if we are writing a function to add 2 integers, the parameters would be  passed like –**

**int add (int num1, int num2)**

**main function**

**The main function is a special function as the computer starts running the code  from the beginning of the main function. Main function serves as the entry  point for the program.**

**Example :**

**package com.apnacollege;**

**public class Main {**

**//A METHOD to calculate sum of 2 numbers - a & b**

**public static void sum(int a, int b) {**

**int sum = a + b;**

**System.*out*.println(sum);**

**}**

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

**int a = 10;**

**int b = 20;**

***sum*(a, b); // Function Call**

**}**

**}**

**Qs. Write a function to multiply 2 numbers.**

**import java.util.\*;**

**public class Functions {**

**//Multiply 2 numbers**

**public static int multiply(int a, int b) {**

**return a\*b;**

**}**

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

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

**int a = sc.nextInt();**

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

**int result = multiply(a, b);**

**System.out.println(result);**

**}**

**}**

**Qs. Write a function to calculate the factorial of a number.**

**​​import java.util.\*;**

**public class Functions {**

**// public static int calculateSum(int a, int b) {**

**//     int sum = a + b;**

**//     return sum;**

**// }**

**// public static int calculateProduct(int a, int b) {**

**//    return a \* b;**

**// }**

**public static void printFactorial(int n) {**

**//loop**

**if(n < 0) {**

**System.out.println("Invalid Number");**

**return;**

**}**

**int factorial = 1;**

**for(int i=n; i>=1; i--) {**

**factorial = factorial \* i;**

**}**

**System.out.println(factorial);**

**return;**

**}**

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

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

**int n = sc.nextInt();**

**printFactorial(n);**

**}**

**}**

**Qs. Write a function to calculate the product of 2 numbers.**

**import java.util.\*;**

**public class Functions {**

**// public static int calculateSum(int a, int b) {**

**//     int sum = a + b;**

**//     return sum;**

**// }**

**public static int calculateProduct(int a, int b) {**

**return a \* b;**

**}**

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

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

**int a = sc.nextInt();**

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

**System.out.println(calculateProduct(a, b));**

**}**

**}**

**Homework Problems**

1. **Make a function to check if a number is prime or not.**
2. **Make a function to check if a given number n is even or not.**
3. **Make a function to print the table of a given number n.**
4. **Read about Recursion.**
5. 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.
6. 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. xn.
7. Write a function that calculates the Greatest Common Divisor of 2 numbers. (BONUS)
8. 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)

1. **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.**

**import java.util.\*;**

**public class Solutions {**

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

**int positive = 0, negative = 0, zeros = 0;**

**System.out.println("Press 1 to continue & 0 to stop");**

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

**int input = sc.nextInt();**

**while(input == 1) {**

**System.out.println("Enter your number : ");**

**int number = sc.nextInt();**

**if(number > 0) {**

**positive++;**

**} else if(number < 0) {**

**negative++;**

**} else {**

**zeros++;**

**}**

**System.out.println("Press 1 to continue & 0 to stop");**

**input = sc.nextInt();**

**}**

**System.out.println("Positives : "+ positive);**

**System.out.println("Negatives : "+ negative);**

**System.out.println("Zeros : "+ zeros);**

**}**

**}**

1. **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.**

**//Try to convert it into a function on your own.**

**import java.util.\*;**

**public class Solutions {**

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

**System.out.println("Enter x");**

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

**int x = sc.nextInt();**

**System.out.println("Enter n");**

**int n = sc.nextInt();**

**int result = 1;**

**//Please see that n is not too large or else result will exceed the size of int**

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

**result = result \* x;**

**}**

**System.out.println("x to the power n is : "+ result);**

**}**

**}**

1. **Write a function that calculates the Greatest Common Divisor of 2 numbers. (BONUS)**

**import java.util.\*;**

**public class Solutions {**

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

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

**int n1 = sc.nextInt();**

**int n2 = sc.nextInt();**

**while(n1 != n2) {**

**if(n1>n2) {**

**n1 = n1 - n2;**

**} else {**

**n2 = n2 - n1;**

**}**

**}**

**System.out.println("GCD is : "+ n2);**

**}**

**}**

**//Try to convert it into a function on your own.**

1. **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)**

**import java.util.\*;**

**public class Solutions {**

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

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

**int n = sc.nextInt();**

**int a = 0, b = 1;**

**System.out.print(a+" ");**

**if(n > 1) {**

**//find nth term**

**for(int i=2; i<=n; i++) {**

**System.out.print(b+" ");**

**//the concept below is called swapping**

**int temp = b;**

**b = a + b;**

**a = temp;**

**}**

**System.out.println();**

**}**

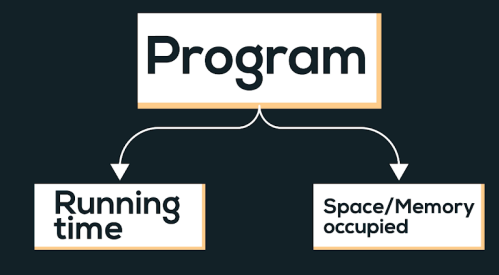
**}**

**}**

**Java - Introduction to Programming**

**Lecture 8**

**Time & Space Complexity**

****

**Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input.**

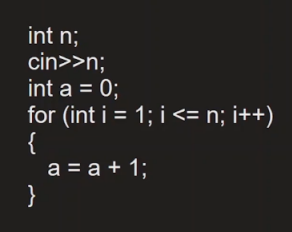
**Types of notations**

**1. O-notation: It is used to denote asymptotic upper bound. For a given function g(n), we denote it by O(g(n)). Pronounced as “big-oh of g of n”. It is also known as worst case time complexity as it denotes the upper bound in which the algorithm terminates.**

**2. Ω-notation: It is used to denote asymptotic lower bound. For a given function g(n), we denote it by Ω(g(n)). Pronounced as “big-omega of g of n”. It is also known as best case time complexity as it denotes the lower bound in which the algorithm terminates.**

**3. 𝚯-notation: It is used to denote the average time of a program.**

**Examples :**

****

**Linear Time Complexity. O(n)**

**Comparison of functions on the basis of time complexity**

**It follows the following order in case of time complexity:**

**O(nn) > O(n!) > O(n3) > O(n2) > O(n.log(n)) > O(n.log(log(n))) > O(n) > O(sqrt(n)) > O(log(n)) > O(1)**

**Note: Reverse is the order for better performance of a code with corresponding time complexity, i.e. a program with less time complexity is more efficient.**

**Space Complexity**

**Space complexity of an algorithm quantifies the amount of time taken by a program to run as a function of length of the input. It is directly proportional to the largest memory your program acquires at any instance during run time.**

**For example: *int* consumes 4 bytes of memory.**

**Java - Introduction to Programming**

**Lecture 10**

**Arrays In Java**

**Arrays in Java are like a list of elements of the same type i.e. a list of integers, a list of booleans etc.**

1. **Creating an Array (method 1) - with new keyword**

**int[] marks = new int[3];**

**marks[0] = 97;**

**marks[1] = 98;**

**marks[2] = 95;**

1. **Creating an Array (method 2)**

**int[] marks = {98, 97, 95};**

1. **Taking an array as an input and printing its elements.**

**import java.util.\*;**

**public class Arrays {**

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

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

**int size = sc.nextInt();**

**int numbers[] = new int[size];**

**for(int i=0; i<size; i++) {**

**numbers[i] = sc.nextInt();**

**}**

**//print the numbers in array**

**for(int i=0; i<arr.length; i++) {**

**System.out.print(numbers[i]+" ");**

**}**

**}**

**}**

**Homework Problems**

1. **Take an array of names as input from the user and print them on the screen.**

**import java.util.\*;**

**public class Arrays {**

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

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

**int size = sc.nextInt();**

**String names[] = new String[size];**

**//input**

**for(int i=0; i<size; i++) {**

**names[i] = sc.next();**

**}**

**//output**

**for(int i=0; i<names.length; i++) {**

**System.out.println("name " + (i+1) +" is : " + names[i]);**

**}**

**}**

**}**

1. **Find the maximum & minimum number in an array of integers.**

**[HINT : Read about Integer.MIN\_VALUE & Integer.MAX\_VALUE in Java]**

**import java.util.\*;**

**public class Arrays {**

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

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

**int size = sc.nextInt();**

**int numbers[] = new int[size];**

**//input**

**for(int i=0; i<size; i++) {**

**numbers[i] = sc.nextInt();**

**}**

**int max = Integer.MIN\_VALUE;**

**int min = Integer.MAX\_VALUE;**

**for(int i=0; i<numbers.length; i++) {**

**if(numbers[i] < min) {**

**min = numbers[i];**

**}**

**if(numbers[i] > max) {**

**max = numbers[i];**

**}**

**}**

**System.out.println("Largest number is : " + max);**

**System.out.println("Smallest number is : " + min);**

**}**

**}**

1. **Take an array of numbers as input and check if it is an array sorted in ascending order.**

**Eg : { 1, 2, 4, 7 } is sorted in ascending order.**

**{3, 4, 6, 2} is not sorted in ascending order.**

**import java.util.\*;**

**public class Arrays {**

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

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

**int size = sc.nextInt();**

**int numbers[] = new int[size];**

**//input**

**for(int i=0; i<size; i++) {**

**numbers[i] = sc.nextInt();**

**}**

**boolean isAscending = true;**

**for(int i=0; i<numbers.length-1; i++) { // NOTICE numbers.length - 1 as termination condition**

**if(numbers[i] > numbers[i+1]) { // This is the condition for descending order**

**isAscending = false;**

**}**

**}**

**if(isAscending) {**

**System.out.println("The array is sorted in ascending order");**

**} else {**

**System.out.println("The array is not sorted in ascending order");**

**}**

**}**

**}**

**Java - Introduction to Programming**

**Lecture 11**

**2D Arrays In Java**

**It is similar to 2D matrices that we studied in 11th and 12th class.**

1. **Creating a 2D Array - with new keyword**

**int[][] marks = new int[3][3];**

1. **Taking a matrix as an input and printing its elements.**

**import java.util.\*;**

**public class TwoDArrays {**

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

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

**int rows = sc.nextInt();**

**int cols = sc.nextInt();**

**int[][] numbers = new int[rows][cols];**

**//input**

**//rows**

**for(int i=0; i<rows; i++) {**

**//columns**

**for(int j=0; j<cols; j++) {**

**numbers[i][j] = sc.nextInt();**

**}**

**}**

**for(int i=0; i<rows; i++) {**

**for(int j=0; j<cols; j++) {**

**System.out.print(numbers[i][j]+" ");**

**}**

**System.out.println();**

**}**

**}**

**}**

1. **Searching for an element x in a matrix.**

**import java.util.\*;**

**public class TwoDArrays {**

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

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

**int rows = sc.nextInt();**

**int cols = sc.nextInt();**

**int[][] numbers = new int[rows][cols];**

**//input**

**//rows**

**for(int i=0; i<rows; i++) {**

**//columns**

**for(int j=0; j<cols; j++) {**

**numbers[i][j] = sc.nextInt();**

**}**

**}**

**int x = sc.nextInt();**

**for(int i=0; i<rows; i++) {**

**for(int j=0; j<cols; j++) {**

**//compare with x**

**if(numbers[i][j] == x) {**

**System.out.println("x found at location (" + i + ", " + j + ")");**

**}**

**}**

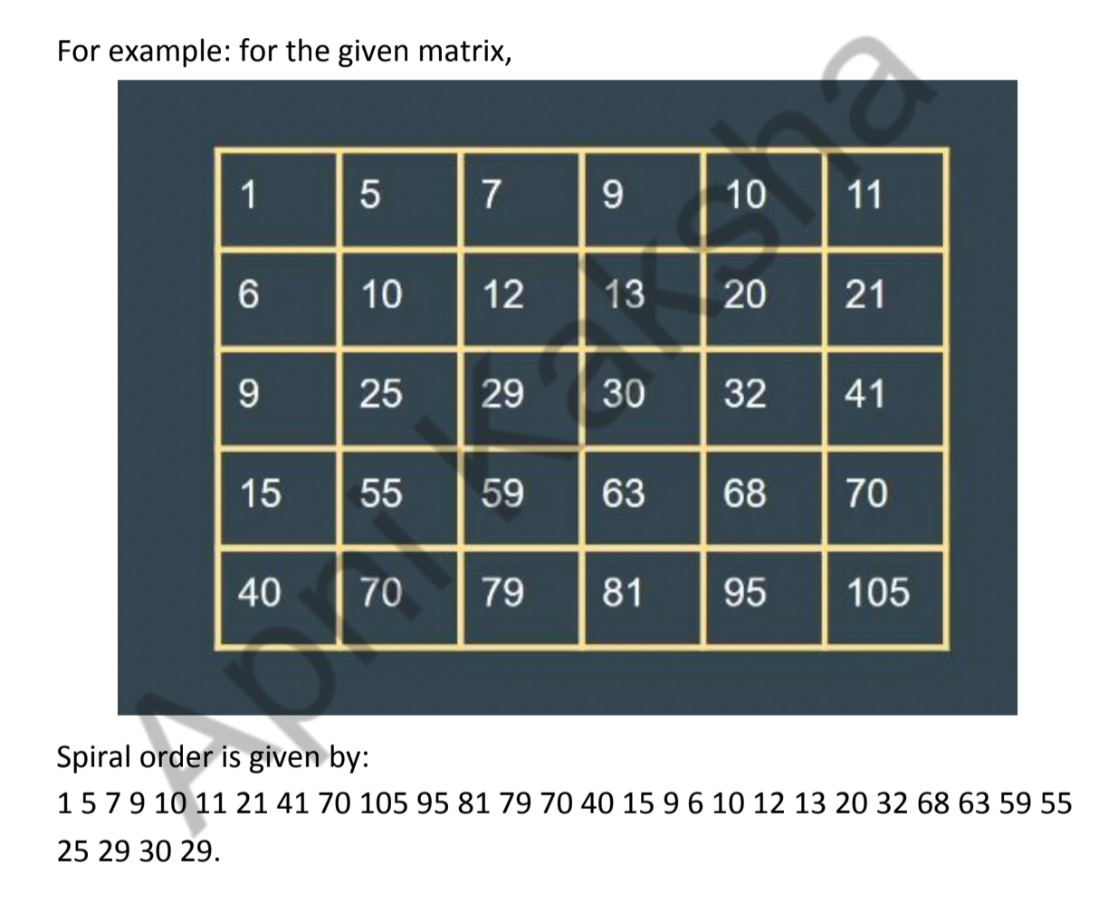
**}**

**}**

**}**

**Homework Problems**

1. **Print the spiral order matrix as output for a given matrix of numbers. [Difficult for Beginners]**

****

**APPROACH :**

**Algorithm: (We are given a 2D matrix of n X m ).**

***1. We will need 4 variables:***

***a. row\_start - initialized with 0.***

***b. row\_end - initialized with n-1.***

***c. column\_start - initialized with 0.***

***d. column\_end - initialized with m-1.***

***2. First of all, we will traverse in the row row\_start from column\_start***

***to column\_end and we will increase the row\_start with 1 as we have***

***traversed the starting row.***

***3. Then we will traverse in the column column\_end from row\_start to***

***row\_end and decrease the column\_end by 1.***

***4. Then we will traverse in the row row\_end from column\_end to***

***column\_start and decrease the row\_end by 1.***

***5. Then we will traverse in the column column\_start from row\_end to***

***row\_start and increase the column\_start by 1.***

***6. We will do the above steps from 2 to 5 until row\_start <= row\_end***

***and column\_start <= column\_end.***

**import java.util.\*;**

**public class Arrays {**

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

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

**int n = sc.nextInt();**

**int m = sc.nextInt();**

**int matrix[][] = new int[n][m];**

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

**for(int j=0; j<m; j++) {**

**matrix[i][j] = sc.nextInt();**

**}**

**}**

**System.out.println("The Spiral Order Matrix is : ");**

**int rowStart = 0;**

**int rowEnd = n-1;**

**int colStart = 0;**

**int colEnd = m-1;**

**//To print spiral order matrix**

**while(rowStart <= rowEnd && colStart <= colEnd) {**

**//1**

**for(int col=colStart; col<=colEnd; col++) {**

**System.out.print(matrix[rowStart][col] + " ");**

**}**

**rowStart++;**

**//2**

**for(int row=rowStart; row<=rowEnd; row++) {**

**System.out.print(matrix[row][colEnd] +" ");**

**}**

**colEnd--;**

**//3**

**for(int col=colEnd; col>=colStart; col--) {**

**System.out.print(matrix[rowEnd][col] + " ");**

**}**

**rowEnd--;**

**//4**

**for(int row=rowEnd; row>=rowStart; row--) {**

**System.out.print(matrix[row][colStart] + " ");**

**}**

**colStart++;**

**System.out.println();**

**}**

**}**

**}**

1. **For a given matrix of N x M, print its transpose.**

**import java.util.\*;**

**public class Arrays {**

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

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

**int n = sc.nextInt();**

**int m = sc.nextInt();**

**int matrix[][] = new int[n][m];**

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

**for(int j=0; j<m; j++) {**

**matrix[i][j] = sc.nextInt();**

**}**

**}**

**System.out.println("The transpose is : ");**

**//To print transpose**

**for(int j=0; j<m ;j++) {**

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

**System.out.print(matrix[i][j]+" ");**

**}**

**System.out.println();**

**}**

**}**

**}**

**Java - Introduction to Programming**

**Lecture 12**

**Strings**

**Declaration**

**String name = "Tony";**

**Taking Input**

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

**String name = sc.next();**

**Concatenation (Joining 2 strings)**

**String firstName = "Tony";**

**String secondName = "Stark";**

**String fullName = firstName + " " + secondName;**

**System.out.println(fullName);**

**Print length of a String**

**String firstName = "Tony";**

**String secondName = "Stark";**

**String fullName = firstName + " " + secondName;**

**System.out.println(fullName.length());**

**Access characters of a string**

**String firstName = "Tony";**

**String secondName = "Stark";**

**String fullName = firstName + " " + secondName;**

**for(int i=0; i<fullName.length(); i++) {**

**System.out.println(fullName.charAt(i));**

**}**

**Compare 2 strings**

**import java.util.\*;**

**public class Strings {**

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

**String name1 = "Tony";**

**String name2 = "Tony";**

**if(name1.equals(name2)) {**

**System.out.println("They are the same string");**

**} else {**

**System.out.println("They are different strings");**

**}**

**//DO NOT USE == to check for string equality**

**//Gives correct answer here**

**if(name1 == name2) {**

**System.out.println("They are the same string");**

**} else {**

**System.out.println("They are different strings");**

**}**

**//Gives incorrect answer here**

**if(new String("Tony") == new String("Tony")) {**

**System.out.println("They are the same string");**

**} else {**

**System.out.println("They are different strings");**

**}**

**}**

**}**

**Substring**

**The substring of a string is a subpart of it.**

**public class Strings {**

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

**String name = "TonyStark";**

**System.out.println(name.substring(0, 4));**

**}**

**}**

**ParseInt Method of Integer class**

**public class Strings {**

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

**String str = "123";**

**int number = Integer.parseInt(str);**

**System.out.println(number);**

**}**

**}**

**ToString Method of String class**

**public class Strings {**

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

**int number = 123;**

**String str = Integer.toString(number);**

**System.out.println(str.length());**

**}**

**}**

**ALWAYS REMEMBER : Java Strings are Immutable.**

**Homework Problems**

1. **Take an array of Strings input from the user & find the cumulative (combined) length of all those strings.**

**import java.util.\*;**

**public class Strings {**

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

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

**int size = sc.nextInt();**

**String array[] = new String[size];**

**int totLength = 0;**

**for(int i=0; i<size; i++) {**

**array[i] = sc.next();**

**totLength += array[i].length();**

**}**

**System.out.println(totLength);**

**}**

**}**

1. **Input a string from the user. Create a new string called ‘result’ in which you will replace the letter ‘e’ in the original string with letter ‘i’.**

**Example :**

**original = “eabcdef’ ; result = “iabcdif”**

**Original = “xyz” ; result = “xyz”**

**import java.util.\*;**

**public class Strings {**

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

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

**String str = sc.next();**

**String result = "";**

**for(int i=0; i<str.length(); i++) {**

**if(str.charAt(i) == 'e') {**

**result += 'i';**

**} else {**

**result += str.charAt(i);**

**}**

**}**

**System.out.println(result);**

**}**

**}**

1. **Input an email from the user. You have to create a username from the email by deleting the part that comes after ‘@’. Display that username to the user.**

**Example :**

**email = “**[**apnaCollegeJava@gmail.com**](mailto:apnacollegeJava@gmail.com)**” ; username = “apnaCollegeJava”**

**email = “**[**helloWorld123@gmail.com**](mailto:helloWorlds123@gmail.com)**”; username = “helloWorld123”**

**import java.util.\*;**

**public class Strings {**

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

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

**String email = sc.next();**

**String userName = "";**

**for(int i=0; i<email.length(); i++) {**

**if(email.charAt(i) == '@') {**

**break;**

**} else {**

**userName += email.charAt(i);**

**}**

**}**

**System.out.println(userName);**

**}**

**}**

**Java - Introduction to Programming**

**Lecture 13**

**String Builder**

**Declaration**

**StringBuilder sb = new StringBuilder("Apna College");**

**System.out.println(sb);**

**Get A Character from Index**

**StringBuilder sb = new StringBuilder("Tony");**

**//Set Char**

**System.out.println(sb.charAt(0));**

**Set a Character at Index**

**StringBuilder sb = new StringBuilder("Tony");**

**//Get Char**

**sb.setCharAt(0, 'P');**

**System.out.println(sb);**

**Insert a Character at Some Index**

**import java.util.\*;**

**public class Strings {**

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

**StringBuilder sb = new StringBuilder("tony");**

**//Insert char**

**sb.insert(0, 'S');**

**System.out.println(sb);**

**}**

**}**

**Delete char at some Index**

**import java.util.\*;**

**public class Strings {**

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

**StringBuilder sb = new StringBuilder("tony");**

**//Insert char**

**sb.insert(0, 'S');**

**System.out.println(sb);**

**//delete char**

**sb.delete(0, 1);**

**System.out.println(sb);**

**}**

**}**

**Append a char**

**Append means to add something at the end.**

**import java.util.\*;**

**public class Strings {**

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

**StringBuilder sb = new StringBuilder("Tony");**

**sb.append(" Stark");**

**System.out.println(sb);**

**}**

**}**

**Print Length of String**

**import java.util.\*;**

**public class Strings {**

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

**StringBuilder sb = new StringBuilder("Tony");**

**sb.append(" Stark");**

**System.out.println(sb);**

**System.out.println(sb.length());**

**}**

**}**

**Reverse a String (using StringBuilder class)**

**import java.util.\*;**

**public class Strings {**

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

**StringBuilder sb = new StringBuilder("HelloWorld");**

**for(int i=0; i<sb.length()/2; i++) {**

**int front = i;**

**int back = sb.length() - i - 1;**

**char frontChar = sb.charAt(front);**

**char backChar = sb.charAt(back);**

**sb.setCharAt(front, backChar);**

**sb.setCharAt(back, frontChar);**

**}**

**System.out.println(sb);**

**}**

**}**

**Homework Problems**

**Try Solving all the String questions with StringBuilder.**

**Java - Introduction to Programming**

**Lecture 14**

**Bit Manipulation**

**Get Bit**

**import java.util.\*;**

**public class Bits {**

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

**int n = 5; //0101**

**int pos = 3;**

**int bitMask = 1<<pos;**

**if((bitMask & n) == 0) {**

**System.out.println("bit was zero");**

**} else {**

**System.out.println("bit was one");**

**}**

**}**

**}**

**Set Bit**

**import java.util.\*;**

**public class Bits {**

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

**int n = 5; //0101**

**int pos = 1;**

**int bitMask = 1<<pos;**

**int newNumber = bitMask | n;**

**System.out.println(newNumber);**

**}**

**}**

**Clear Bit**

**import java.util.\*;**

**public class Bits {**

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

**int n = 5; //0101**

**int pos = 2;**

**int bitMask = 1<<pos;**

**int newBitMask = ~(bitMask);**

**int newNumber = newBitMask & n;**

**System.out.println(newNumber);**

**}**

**}**

**Update Bit**

**import java.util.\*;**

**public class Bits {**

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

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

**int oper = sc.nextInt();**

**// oper=1 -> set; oper=0 -> clear**

**int n = 5;**

**int pos = 1;**

**int bitMask = 1<<pos;**

**if(oper == 1) {**

**//set**

**int newNumber = bitMask | n;**

**System.out.println(newNumber);**

**} else {**

**//clear**

**int newBitMask = ~(bitMask);**

**int newNumber = newBitMask & n;**

**System.out.println(newNumber);**

**}**

**}**

**}**

**Homework Problems**

1. **Write a program to find if a number is a power of 2 or not.**
2. **Write a program to toggle a bit a position = “pos” in a number “n”.**
3. **Write a program to count the number of 1’s in a binary representation of the number.**
4. **Write 2 functions => decimalToBinary() & binaryToDecimal() to convert a number from one number system to another. [BONUS]**

**Sorting in JAVA**

1. **Bubble Sort**

**Idea: if arr[i] > arr[i+1] swap them. To place the element in their respective position, we have to do the following operation N-1 times.**

**Time Complexity: O(N2)**

**Code**

**import java.util.\*;**

**class Sorting {**

**public static void printArray(int arr[]) {**

**for(int i=0; i<arr.length; i++) {**

**System.out.print(arr[i]+" ");**

**}**

**System.out.println();**

**}**

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

**int arr[] = {7, 8, 1, 3, 2};**

**//bubble sort**

**for(int i=0; i<arr.length-1; i++) {**

**for(int j=0; j<arr.length-i-1; j++) {**

**if(arr[j] > arr[j+1]) {**

**//swap**

**int temp = arr[j];**

**arr[j] = arr[j+1];**

**arr[j+1] = temp;**

**}**

**}**

**}**

**printArray(arr);**

**}**

**}**

1. **Selection Sort**

**Idea: The inner loop selects the minimum element in the unsorted array and places the elements in increasing order.**

**Time complexity: O(N2)**

**Code**

**import java.util.\*;**

**class Sorting {**

**public static void printArray(int arr[]) {**

**for(int i=0; i<arr.length; i++) {**

**System.out.print(arr[i]+" ");**

**}**

**System.out.println();**

**}**

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

**int arr[] = {7, 8, 1, 3, 2};**

**//selection sort**

**for(int i=0; i<arr.length-1; i++) {**

**int smallest = i;**

**for(int j=i+1; j<arr.length; j++) {**

**if(arr[j] < arr[smallest]) {**

**smallest = j;**

**}**

**}**

**//swap**

**int temp = arr[smallest];**

**arr[smallest] = arr[i];**

**arr[i] = temp;**

**}**

**printArray(arr);**

**}**

**}**

1. **Insertion Sort**

**Idea: Take an element from the unsorted array, place it in its corresponding position in the sorted part, and shift the elements accordingly.**

**Time Complexity: O(N2)**

**Code**

**import java.util.\*;**

**class Sorting {**

**public static void printArray(int arr[]) {**

**for(int i=0; i<arr.length; i++) {**

**System.out.print(arr[i]+" ");**

**}**

**System.out.println();**

**}**

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

**int arr[] = {7, 8, 1, 3, 2};**

**//insertion sort**

**for(int i=1; i<arr.length; i++) {**

**int current = arr[i];**

**int j = i - 1;**

**while(j >= 0 && arr[j] > current) {**

**//Keep swapping**

**arr[j+1] = arr[j];**

**j--;**

**}**

**arr[j+1] = current;**

**}**

**printArray(arr);**

**}**

**}**

**OBJECT ORIENTED PROGRAMMING SYSTEMS**

**JAVA**

**Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts defined below :**

**Class is a user-defined data type which defines its properties and its functions. Class is the only logical representation of the data. For example, Human being is a class. The body parts of a human being are its properties, and the actions performed by the body parts are known as functions. The class does not occupy any memory space till the time an object is instantiated.**

**Object is a run-time entity. It is an instance of the class. An object can represent a person, place or any other item. An object can operate on both data members and member functions.**

**Example 1:**

**class Student {**

**String name;**

**int age;**

**public void getInfo() {**

**System.out.println("The name of this Student is " + this.name);**

**System.out.println("The age of this Student is " + this.age);**

**}**

**}**

**public class OOPS {**

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

**Student s1 = new Student();**

**s1.name = "Aman";**

**s1.age = 24;**

**s1.getInfo();**

**Student s2 = new Student();**

**s2.name = "Shradha";**

**s2.age = 22;**

**s2.getInfo();**

**}**

**}**

**Example 2:**

**class Pen {**

**String color;**

**public void printColor() {**

**System.out.println("The color of this Pen is " + this.color);**

**}**

**}**

**public class OOPS {**

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

**Pen p1 = new Pen();**

**p1.color = blue;**

**Pen p2 = new Pen();**

**p2.color = black;**

**Pen p3 = new Pen();**

**p3.color = red;**

**p1.printColor();**

**p2.printColor();**

**p3.printColor();**

**}**

**}**

**Note : When an object is created using a new keyword, then space is allocated for the variable in a heap, and the starting address is stored in the stack memory.**

**‘this’ keyword :  ‘this’ keyword in Java that refers to the current instance of the class. In OOPS it is used to:**

1. **pass the current object as a parameter to another method**
2. **refer to the current class instance variable**

**Constructor : Constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally.**

* **Constructors have the same name as class or structure.**
* **Constructors don’t have a return type. (Not even void)**
* **Constructors are only called once, at object creation.**

**There can be three types of constructors in Java.**

**1. Non-Parameterized constructor : A constructor which has no argument is known as non-parameterized constructor(or no-argument constructor). It is invoked at the time of creating an object. If we don’t create one then it is created by default by Java.**

**class Student {**

**String name;**

**int age;**

**Student() {**

**System.out.println("Constructor called");**

**}**

**}**

**2. Parameterized constructor : Constructor which has parameters is called a parameterized constructor. It is used to provide**

**different values to distinct objects.**

**class Student {**

**String name;**

**int age;**

**Student(String name, int age) {**

**this.name = name;**

**this.age = age;**

**}**

**}**

**3. Copy Constructor : A Copy constructor is an overloaded**

**constructor used to declare and initialize an object from another object. There is only a user defined copy constructor in Java(C++ has a default one too).**

**class Student {**

**String name;**

**int age;**

**Student(Student s2) {**

**this.name = s2.name;**

**this.age = s2.age;**

**}**

**}**

**Note : Unlike languages like C++, Java has no Destructor. Instead, Java has an efficient  garbage collector that deallocates memory automatically.**

**Polymorphism**

**Polymorphism is the ability to present the same interface for differing underlying forms (data types). With polymorphism, each of these classes will have different underlying data. Precisely, Poly means ‘many’ and morphism means ‘forms’.**

**Types of Polymorphism IMP**

**1. Compile Time Polymorphism (Static)**

**2. Runtime Polymorphism (Dynamic)**

**Let’s understand them one by one :**

**Compile Time Polymorphism : The polymorphism which is implemented at the compile time is known as compile-time polymorphism. Example - Method Overloading**

**Method Overloading : Method overloading is a technique which allows you to have more than one function with the same function name but with different functionality. Method overloading can be possible on the following basis:**

**1. The type of the parameters passed to the function.**

**2. The number of parameters passed to the function.**

**class Student {**

**String name;**

**int age;**

**public void displayInfo(String name) {**

**System.out.println(name);**

**}**

**public void displayInfo(int age) {**

**System.out.println(age);**

**}**

**public void displayInfo(String name, int age) {**

**System.out.println(name);**

**System.out.println(age);**

**}**

**}**

**Runtime Polymorphism : Runtime polymorphism is also known as dynamic polymorphism. Function overriding is an example of runtime polymorphism. Function overriding means when the child class contains the method which is already present in the parent class. Hence, the child class overrides the method of the parent class. In case of function overriding, parent and child classes both contain the same function with a different definition. The call to the function is determined at runtime is known as runtime polymorphism.**

**class Shape {**

**public void area() {**

**System.out.println("Displays Area of Shape");**

**}**

**}**

**class Triangle extends Shape {**

**public void area(int h, int b) {**

**System.out.println((1/2)\*b\*h);**

**}**

**}**

**class Circle extends Shape {**

**public void area(int r) {**

**System.out.println((3.14)\*r\*r);**

**}**

**}**

**Inheritance**

**Inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such a way, you can reuse, extend or modify the attributes and behaviors which are defined in other classes.**

**In Java, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.**

**Types of Inheritance :**

**1. Single inheritance : When one class inherits another class, it is known as single level inheritance**

**class Shape {**

**public void area() {**

**System.out.println("Displays Area of Shape");**

**}**

**}**

**class Triangle extends Shape {**

**public void area(int h, int b) {**

**System.out.println((1/2)\*b\*h);**

**}**

**}**

**2. Hierarchical inheritance : Hierarchical inheritance is defined as the process of deriving more than one class from a base class.**

**class Shape {**

**public void area() {**

**System.out.println("Displays Area of Shape");**

**}**

**}**

**class Triangle extends Shape {**

**public void area(int h, int b) {**

**System.out.println((1/2)\*b\*h);**

**}**

**}**

**class Circle extends Shape {**

**public void area(int r) {**

**System.out.println((3.14)\*r\*r);**

**}**

**}**

**3. Multilevel inheritance : Multilevel inheritance is a process of deriving a class from another derived class.**

**class Shape {**

**public void area() {**

**System.out.println("Displays Area of Shape");**

**}**

**}**

**class Triangle extends Shape {**

**public void area(int h, int b) {**

**System.out.println((1/2)\*b\*h);**

**}**

**}**

**class EquilateralTriangle extends Triangle {**

**int side;**

**}**

**4. Hybrid inheritance : Hybrid inheritance is a combination of**

**simple, multiple inheritance and hierarchical inheritance.**

**Package in Java**

**Package is a group of similar types of classes, interfaces and sub-packages. Packages can be built-in or user defined.**

**Built-in packages - java, util, io etc.**

**import java.util.Scanner;**

**import java.io.IOException;**

**Access Modifiers in Java**

* **Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.**
* **Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.**
* **Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.**
* **Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.**

**package newpackage;**

**class Account {**

**public String name;**

**protected String email;**

**private String password;**

**public void setPassword(String password) {**

**this.password = password;**

**}**

**}**

**public class Sample {**

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

**Account a1 = new Account();**

**a1.name = "Apna College";**

**a1.setPassword("abcd");**

**a1.email = "hello@apnacollege.com";**

**}**

**}**

**Encapsulation**

**Encapsulation is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.(Data hiding: a language feature to restrict access to members of an object, reducing the negative effect due to dependencies. e.g. "protected", "private" feature in Java).**

**Abstraction**

**We try to obtain an abstract view, model or structure of a real life problem, and reduce its unnecessary details. With definition of properties of problems, including the data which are affected and the operations which are identified, the model abstracted from problems can be a standard solution to this type of problems. It is an efficient way since there are nebulous real-life problems that have similar properties.**

**In simple terms, it is hiding the unnecessary details & showing only the essential parts/functionalities to the user.**

**Data binding : Data binding is a process of binding the application UI and business logic. Any change made in the business logic will reflect directly to the application UI.**

**Abstraction is achieved in 2 ways :**

* **Abstract class**
* **Interfaces (Pure Abstraction)**

1. **Abstract Class**

* **An abstract class must be declared with an abstract keyword.**
* **It can have abstract and non-abstract methods.**
* **It cannot be instantiated.**
* **It can have constructors and static methods also.**
* **It can have final methods which will force the subclass not to change the body of the method.**

**abstract class Animal {**

**abstract void walk();**

**void breathe() {**

**System.out.println("This animal breathes air");**

**}**

**Animal() {**

**System.out.println("You are about to create an Animal.");**

**}**

**}**

**class Horse extends Animal {**

**Horse() {**

**System.out.println("Wow, you have created a Horse!");**

**}**

**void walk() {**

**System.out.println("Horse walks on 4 legs");**

**}**

**}**

**class Chicken extends Animal {**

**Chicken() {**

**System.out.println("Wow, you have created a Chicken!");**

**}**

**void walk() {**

**System.out.println("Chicken walks on 2 legs");**

**}**

**}**

**public class OOPS {**

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

**Horse horse = new Horse();**

**horse.walk();**

**horse.breathe();**

**}**

**}**

**2. Interfaces**

* **All the fields in interfaces are public, static and final by default.**
* **All methods are public & abstract by default.**
* **A class that implements an interface must implement all the methods declared in the interface.**
* **Interfaces support the functionality of multiple inheritance.**

**interface Animal {**

**void walk();**

**}**

**class Horse implements Animal {**

**public void walk() {**

**System.out.println("Horse walks on 4 legs");**

**}**

**}**

**class Chicken implements Animal {**

**public void walk() {**

**System.out.println("Chicken walks on 2 legs");**

**}**

**}**

**public class OOPS {**

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

**Horse horse = new Horse();**

**horse.walk();**

**}**

**}**

**Static Keyword**

**Static can be :**

1. **Variable (also known as a class variable)**
2. **Method (also known as a class method)**
3. **Block**
4. **Nested class**

**class Student {**

**static String school;**

**String name;**

**}**

**public class OOPS {**

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

**Student.school = "JMV";**

**Student s1 = new Student();**

**Student s2 = new Student();**

**s1.name = "Meena";**

**s2.name = "Beena";**

**System.out.println(s1.school);**

**System.out.println(s2.school);**

**}**

**}**

**ArrayList in Java**

**Operations :**

1. **Declare an ArrayList of different Types**
2. **Add Element**
3. **Get Element**
4. **Add Element at a specific Index**
5. **Set Element at a specific Index**
6. **Delete Element from an Index**
7. **Size of the List**
8. **Loop/Iterate on the List**
9. **Sort the List**

**import java.util.ArrayList;**

**import java.util.Collections;**

**class ArrayLists {**

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

**ArrayList<Integer> list = new ArrayList<Integer>();**

**ArrayList<String> list2 = new ArrayList<String>();**

**ArrayList<Boolean> list3 = new ArrayList<Boolean>();**

**//add elements**

**list.add(1);**

**list.add(3);**

**list.add(4);**

**list.add(5);**

**System.out.println(list);**

**//to get an element**

**int element = list.get(0); // 0 is the index**

**System.out.println(element);**

**//add element in between**

**list.add(1,2); // 1 is the index and 2 is the element to be added**

**System.out.println(list);**

**//set element**

**list.set(0,0);**

**System.out.println(list);**

**//delete elements**

**list.remove(0); // 0 is the index**

**System.out.println(list);**

**//size of list**

**int size = list.size();**

**System.out.println(size);**

**//Loops on lists**

**for(int i=0; i<list.size(); i++) {**

**System.out.print(list.get(i) + " ");**

**}**

**System.out.println();**

**//Sorting the list**

**list.add(0);**

**Collections.sort(list);**

**System.out.println(list);**

**}**

**}**

**Homework Problems**

**Try solving all problems of arrays with arraylists.**