Core Java



Java SE: -Java Standard Edition

Core java

Java SE

Adv java

* It deals with developing **stand alone applications** & also basic programs for the network.

Java EE: - Java Enterprise Edition

* It Deals with developing business solutions on a network.

Java ME: - Java Micro Edition (or) Java Mobile Edition

* It deals with developing embedded systems & wireless applications.

HTML: - Hyper Text Markup Language

NIC: - Network Interface Card. Card means Electronic Circuits

🡪A network is an interconnection of Computer.

🡪A group of programs is called Software.

🡪 Without Software, Hardware did not work.

1. Hardware

2. Software digital signals

3. Protocol

1 & 0 called bits & also called

Binary digits

1🡪 Power supply

0🡪 no power supply

Protocol is a specification of rules to be followed by every computer on the network.

When to send / receive data

Protocol contain 2 things

How to send / receive data

What is Hand Shaking mechanism?

A) Two computers establishing a connection, communicating with each other & finally disconnecting from each other is called hand shaking mechanism.

Packet: - A packet represents a group of bytes of data. Envelop is called frame.

🡪 What is IP Address?

A) IP Address is a unique ID number given to every computer in the network.

ID address will have 4 numbers separated by dot (.) symbol.

EX:-

192.55.100.01

b b b b 🡪 4 bytes

TCP/IP: - Transition control protocol / Internet Protocol

Client: - A client is a machine that sends a request for some service.

Server: - A Server is a machine that provides services to the clients. A network is also called client server architecture.

DOD:-Department Of Defense.

ARPANET: - Advance Research Project Agency Network. This is the first network in the network.

They connected slowly

🡪 What is the advantage of Network?

A) Resource sharing is main advantage of network.

This means 🡪Sharing memory

🡪Sharing processor

🡪Sharing data

🡪Sharing software

🡪Sharing hardware

Dummy terminals means without C.P.U

AT&T:-Advance Telephone Technology.

Internet: - Internet is a networking of computers existing in the world.

It is global network of networks on the Earth.

Intranet: - Intranet is a network of all the computers existing on the earth. It is a global network of networks. It is a network of computer with in an organization.

Extranet: - It is a network out side the organization that connects it with the network of some other organization.

Extranet

ISP: - Internet Service Provider

It is a person or company who provides connection in to the internet.

The first ISP in India is VSNL.

` Dial-up

Connection

Web browser

50 GBPS

Web server

Lased line

50 Giga bits for second, it is a band width. Dial-up connection is a connection into internet through an ISP. Disadvantage dial-up connection is slow (low speed). There is no security of data on internet. Because ISP will give speed 64 kbps. Lased line is a direct connection into internet without the help of ISP. Complete security for data for lased line connection. It is a high speed connection.

Web Server: - It is a software that should be installed on an internet server. Popular Web servers are (IIS) Internet Information Server, Web logic, Web Sphere, Apache, Tomcat, J boss etc...

HTTP is the most widely used protocol.

Web Browsers:- 1) Internet Explorer, 2) Netscape Navigation, 3) Fire Fox, 4)Opera Etc….

C Compiler is translate in to machine language generates equivalent machine language code.

Object code: - equivalence machine language code of a ‘c’ program.

C Platform: - Micro processor

**P**

**X.c**

**X.obj**

**X.exe**

Celeron processor

**C**

Object Code

X.obj contains object code & contains machine language code. Header code is included .exe. Header files contains functions is a code of function. X.exe file has a header file code, but X.obj does not contain header file code.

4004 is the first micro processor in the world (8008, 8086 (120 instructions), 80286, 80386, 80486, pentium1, pentium2, pentiem3, pentium4). Pentium3 processor is containing 220 instructions. Every Micro processor understandable to that own instructions. Micro processor is different instruction of other instructions.

Translator: - Translator is a program that converts the computer program in to machine language.

There are 3 types

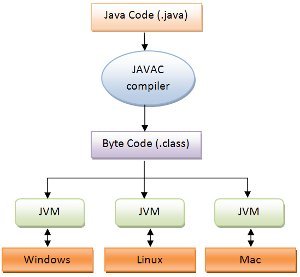
1) Interpreter, 2) Compiler, 3) Assembler

Interpreter converts only one line at a time into machine language. Compiler converts all lines at a time in to machine language. Compiler is at least 10 times faster than interpreter. Assembler converts assembly language into machine language.

C, C++ programs are called system dependent programs, because they can be executed only on that computer a system where they have been developed.

C, C++, Pascal, FORTRAN are not used on internet. Internet is an open network or a public network.

Java Platform







JVM: - Java Virtual Machine. It is a program. It converts byte code instructions in to machine language instructions. It is understandable to the present processor. JVM check what processor you are using. Byte code is a system independent. JVM is a system dependent. JVM is developed in ‘c’ language.

Java is system dependent. Because java programs don’t depend on the computer system where they have been developed. Java writes once and run any ware. It is suitable for internet. Internet doesn’t contain the security for the data.

What is the difference between .exe & .class file?

A) .exe file contains machine language instructions which are executed by the micro processor. .exe files are system depended.

B) .class files byte code instructions executed by the JVM. .class files are system independent.

Security problems for data on internet: -

1. Eaves dropping: - Reading others data illegally.
2. Tampering: - Not only reading others data but also modifying.
3. Impersonation: - A person is acting as another person on internet.
4. Virus: - Virus is a computer program that harms the data software and hardware of a computer system. Virus will spread through .exe, .com, .doc, .jpg and audio video files.

Text file can not spread virus.

Who developed the first virus in the world?

A) A) America B) Afghanistan C) India D) Pakistan

Solutions: -

1. Eaves dropping: - Encryption / Description

🡪 Converting data into unreadable format.

🡪 Converting back into original format.

🡪 Using some password or key.

2. Tampering: - ---- DO ----

3. Impersonation: - Digital signature is a file that contains personal ID information.

4. Virus: - .class files can not carry virus.

In 1990 --- Sun Micro systems INC, US 🡪 James Gauzing.

In 1993 --- Oak – It is completely system Independent.

Later on Oak name has changed. Its name is JAVA.

The Symbol of java is Tea cup.

Features of java: -

1. Simple: - Java is a simple programming language. Learning & practicing java is easy because of its resemblance C and C++ pointers are not available on JAVA.

Why pointers are not available on java?

A) 1. Pointers lead to confusion for a programmer

2. Pointers easily crash a program.

3. Using pointers virus & hacking programs can be written.

1. Object Oriented: - Java is a purely object-oriented programming language. Java programs use objects & classes.

🡪Object: - Object is anything exists really in the world. An object contains properties and performs actions. Properties are represented by variables & actions are formed by functions. Functions are called methods in java.

🡪Class: - A class is a group of name that represents properties & actions of objects. Class does not exist physically. Where as object exist physically. A class is a plan or model to create objects, from the class to create the objects. In every java program we must contain at least one class.

1. Distributed: - Information is distributed on various on a network. Using java, we can write programs which capture information & distribute it to clients.
2. Robust: - Robust means strong. Java programs are strong. Java programs will not crash easily because of its execution handling and its memory management features. In C, C++ less or more than memory the programs are crash. In java this problem is not occurred, because JVM is allotted the memory.

🡪Memory allocation: - JVM’S class loader sub system

🡪Memory management: - JVM’S class garbage collection, this collector removes the unused memory. Exception means run time error. We can not handling the exception in ‘C’. In case java we can handle smoothly

1. Secure: - Java enables the constructions of various free and tamper-free systems.
2. Architecture Neutral: - Java’s byte code isn’t machine dependent. It can be run any machine with any processor & with any operating system.
3. Portable: - Java programs give same results on all machines. Everything is clearly defined in java specification & nothing is left to o / s. If the Program is eliding give same results are called portability.
4. Interpreted: - Java programs are compiled to generate the byte code. This byte code can be downloaded & interpreted by the interpreter in JVM. In java we use interpreter. This interpreter is slow that’s why problem is occurred
5. High performance: - Along with interpreter, there will be JIT (just in time) compiler which enhance the speed of execution.

**Interpreter**

**JIT**

**Compiler**

**S**

**.class**

JVM

Java Compiler

1. High performance: - Along with interpreter, there will be JIT (just in time) compiler which enhance the speed of execution.
2. Multi Threaded: - A thread is a process or execution. We can create different processing in java called ‘threads’. This as an essential feature to design server side programs.
3. Dynamic: - We can develop programs in java which dynamically interact with the user on internet.

Ex: - Applets.

JVM Architecture: - (Black diagram of JVM)

Class files: - Means .class files. It contains byte code.

Class loader sub system: - The class loader sub system does following tasks.

1. It loads .class files into memory.
2. It verifies the byte code instructions.
3. Then it will be a lot necessary memory for the java program.

This memory is divided into 5 parts or blocks are called run time data areas. They are

1. Method area: - Class code & method code is stored on method area.
2. Heap: - (Heap) Objects are created on heap memory. The separated memory is allotted in heap.
3. Java stacks: - These are the areas where java methods are executed. Each java stack is divided into several frames. On each frame separate method is executed.
4. PC registers: - PC means program counter. In PC registers store memory address.

**Class**

**Loader sub system**

Class files

**Run time data areas**

**Method area**

**Heap**

**PC registers**

**Native method areas**

**Java sockets**

**Native method**

**Interface**

**Execution engine**

Native

Method

Libraries

1. Native method stacks: - These are the memory areas where native methods.

EX: - C and CPP functions are executed. JVM executed C and CPP programs are also executed.

Note: - To execute native methods libraries (C, CPP header files) are required. These headers files are linked with JVM by a program called native method interface.

Execution engine: - It contains interpreter & JIT compiler. This converts the byte code instructions in to machine language instructions, so that the processor can execute them.

🡪 What is Hot Spot?

A) The block of code let to JIT compiler for the execution is called hot spot. [www.hotspot.com](http://www.hotspot.com) is developed by JVM.

Comments: -Comments represents description of the features of a program comments increase readability of a program. There are 3 types of comments in java

1. Single line comments: - These comments are started with //-----

EX: - // this is comment line

1. Multi line comments: - They start with /\* and end with \*/

EX: - /\* this is comment line\*/

1. Java documentation comments: - These comments are start with /\*\* and end with \*/

These comments are useful to create a HTML file called API (application programming Interface) document.

🡪 What is API document?

A) API document is a HTML file that contains description of all the features of software, a product or a technology. It works has a heap file.

API document is like user documentation. To create API doc we need java doc compiler. Function code is available in header file.

First program: -

🡪 /\*this is my first java program

To display a message

Author: SUDHEER REDDY

Version: V1.0

Project: POTHURAI \*/

class Hello

{

public static void main(String args[])

{

// prints the string "Hello SUDHEER REDDY" on screen

System.out.println("Hello SUDHEER REDDY");

}

}

D:\psr\Core java>javac Hello.java

D:\psr\Core java>java Hello

Hello SUDHEER REDDY

java library

🡫

package

🡫

class/interface

🡫

methods

Interface is similar to a class

java.lang(package)

🡫

String.system(class)

import java.lang.String;

import java.lang.System;

Star (\*) means all classes and interfaces.

🡪import java.lang.\*;

🡪 What is the difference between #include and import?

A) #include makes a C or C++ compiler to go to standard library and copy the entire header file code in to a C/C++ programs. So the program size increases unnecessarily wasting memory & processor time.

Import statement makes the JVM to go to the java library, execute the code & substitute the result in to the java program. That’s JVM doesn’t copy any code. So import is more efficient than #include

🡪What is JRE?

A) JRE means java run time environment.

JRE=JVM + java library

JRE is available in JDK1.5. It represents entire software.

public static void main(String args[])

JVM starts execution of a java program from main method. Main method is standing of the JVM. The main method is not, the execution will be stopped.

🡪 A method can receive data from outside

🡪 A method can also return some result.

[ ] 🡪 A group of elements. We can store several strings in to args x.

[ ] called array.

Void means no value. Main method does not return any value.

🡪 What are command line arguments?

A) The values passed to main method at the time of running a program or called command line arguments. Its stores in array.

Calling method in java:-

* 1. Create on object to the class

Classname obj=new Classname( );

* 1. Using object name .method name called the method

Obj.method( );

Static: - A static method is a method that can be called can without using any object.

Public: - Public members are available to out side programs

System.out 🡪 It print stream class.

System is a class. Out (field) is a variable.

Variable represent the memory location to store the data.

🡪 Print method displays the results and then keeps the cursor in the same line.

🡪 println method also display the result. After displaying the result it throws the cursor to the next line.

Comments are not executable.

🡪 // calculate sum of 2 no's

class Sum

{

public static void main(String[] args)

{

// vars

int x,y;

// store integer values

x=10;

y=25;

// find sum and store result in z

int z=x+y;

// display the result

System.out.println("Sum of two no's = "+z);

}

}

Save: - “Sum.java”

Double quotation means save as exactly in java file.

D:\psr\Core java>javac Sum.java

D:\psr\Core java>java Sum

Sum of two no's = 35

🡪 \n is an escape sequence. The change the format of the out put.

Backs lace code \

Escape sequence Meaning

\n next line

\t <tab>

\v <enter>

\b <back space>

\\ \

\” “

\’ ‘

Every java program every statement must end with a semicolon ( ; ).

Java is a case sensitive language.

Naming conventions in java (Rules): -

1. Package names in java are written in all small letters

EX: - java.awt;

java.io;

javax.swing;

1. Each word of a class name and interface name start with a capital letter.

EX: - String, DataInputStream, ActionListener

1. Method names starts with a small letter, then each word starts with a capital letter.

EX: - readLine( ); , getNumberInstance( );

1. Variable names also follow the above rules.

EX: - age, empName, employee\_Net\_Sal;

1. Constants should be written using all capital letters.

EX: - PI, MAX-VALUE, Font.BOLD;

Here Font is a class, BOLD is a constant.

1. All key words should be written in all small letters.

EX: - public, static, void, main etc…

🡪 java.lang package by default import in to every java program.

Important packages of core java: -

1. java.lang: - This package got primary classes and interfaces essential for java program. It consists of wrapper classes, strings, threads, etc. wrapper classes are useful to create an object. Thread means process.
2. java.util: - (utility) in this util data structures are available. This package contains useful classes and interfaces like stack, linked list, hash table, arrays. etc.
3. java.io: - This package handles files and input output related tasks. (io – input output).
4. java.awt : - abstract window tool kit.

This package helps to develop GUI. It consists of important sub packages.

Java.awt.event 🡪 event is useful to providing actions.

GUI🡪 graphics user interface.

1. javax.swing: - This package helps to develop GUI like java.awt. X means extended. This package is developed from another package.
2. java.net: - Client – server programming can be done using this package. Net stands for network.
3. java.applet: - applets are programs which come from a server into a client and get executed on the client machine. An applet is a program that comes to the internet server.
4. java.sql: - structured query language. This package helps to connect to database like oracle and utilize them in java. A data base is a storage data.

🡪 A variable is memory location to store data. Int represent you can store integer numbers. Numbers are without decimal points. It is a data type.

🡪 A data type represents the type of data stored into variable.

Data types and literals:-

A data type represents the type of data stored in variable.

Integer data types: -

Data type Memory size min and max value

1. byte 1 Byte - 128 to +127

2. short 2 Bytes - 32768 to +32767

3. int 4 Bytes - 21474836 to +2147483647

4. long 8 Bytes -9223372036854775808 to

+ 9223372036854775807

These data types represent integer numbers

For example 0, 15000,125, and -15 etc….

‘0’ is the stored in +ve range. So1 number is less.

‘0’ comes under the numbers to add.

Ex: - byte rno =10; means literal.

long x = 150L;

A literal represents the value directly stored in to a variable.

This L is writing the JVM will allot of 8 bytes otherwise the JVM will allot 2 bytes.

Float data types: - Float means floating point numbers. Float data type represent numbers with decimal points.

Ex: - 1.5, 3.14159,-2.2, 100.0 etc….

Float means single precision floating point number. Double means double precision floating point number. Precision means accuracy.

Data type Memory size min and max value

1) float 4 bytes 1.4e-45 to +3.4e38.

2) double 8 bytes 4.1e-324 to+1.8e308.

e means into 10 powers. (x 10power).

Ex:-

1.4e-45=1.4x10-45

Float pi = 3.142f;

(Data type) (Variable) (Literal))

double distance = 1.98e8;

🡪 What is the distance between a float and double?

A). Float represent up to 7 digits accurately after decimal point.

Double represent up to 15 digits accurately after decimal point.

Character data type: -

Data type Memories size min and max valve

char 2 bytes 0 to 65535

This data type represents a single character. In C or CPP languages using only one byte in a char.

Ex: - char ch = ‘x’;

Single character literal you writing put them inside ‘ ’(single code).

You write string put the double codes (“ ”).

ASCII: - American Standard Code for Information Interchange. It is a standard. It should followed by all keyboard manufactures.

114 keys are using in keyboard. 230 characters are there in our keyboard approximately (small & large).

🡪 What is uni code?

A) Uni code is a standard to include the alphabet from all human languages in java. Uni code system uses 2 bytes to represent a character.

String data types: - A string represent group of characters.

EX: -

String name=“SUDHEER REDDY”;

String str=new String(“SUDHEER REDDY”);

Boolean data types: - These data types represent only 2 values either a true or false.

EX: - boolean response=true;

Operators: - An operator is a symbol that performs on an operation.

EX: -

If an operator acts on a single variable, it is called unary operator.

If an operator acts on a 2 variables, it is called binary operator.

If an operator acts on a 3 variables, it is called ternary operator.

Operators in java: -

1. Arithmetic operators: - + , - , \* , / , %

These operators perform basic arithmetic calculations.

EX: - If a=13, b=5;

Operator

a + b

Operands

Operator Meaning Example Result

+ Addition a + b 18

- Subtraction a – b 8

\* Multiplication a \* b 65

/ Division a / b 2.6

% Modules a % b 3

2) Unary operators: - - , ++ , --

These operators act upon a single operand.

🡪Unary minus ( - ):- This operator negates the value of a variable.

EX:-

int x=5;

System.out.println(-x); o/p:- -5

System.out.println(-(-x)); o/p:- 5

Increment (++): - This operator increases the value of a variable by ‘1‘.

EX: - int x=1;

++x; O/P:- 2

x++; O/P:- 3

Writing ++ before a variable is called pre – incrementation

Writing ++ after a variable is called post – incrementation

In pre incrementation , incrementation is done and other task is done next. In post incrementation any other task is performed first, incrementation is done at next.

EX: - Pre incrementation

x=1;

System.out.print(x); o/p: - 1

System.out.print(++x); o/p: - 2

System.out.print(x++); o/p: - 2

a=1; b=2;

a=++b;

System.out.print(a); o/p: 3

System.out.print(b); o/p: 3

Post incrementation

x=1;

System.out.print(x); o/p: 1

System.out.print(x++); o/p: 1

System.out.print(x); o/p: 2

a=1; b=2;

a=b++;

System.out.print(a); o/p: 2

System.out.print(b); o/p: 3

🡪What is the value of the following expression? Given a=7? ++a \* a++?

A) A) 49 B) 64 C) 72 D) none of these

Decrement (--): - This operator decreases the value of a variable by ‘1‘.

EX: - int x=1;

--x; O/P: 0

x--; O/P: -1

Writing -- before a variable is called pre – decrementation

Writing -- after a variable is called post – decrementation

In pre decrementation , decrementation is done and other task is done next. In post decrementation any other task is performed first, decrementation is done at next.

EX: - Pre decrementation

x=1;

System.out.print(x); o/p: 1

System.out.print(--x); o/p: 0

System.out.print(x--); o/p: 0

a=1; b=2;

a=--b;

System.out.print(a); o/p: 1

System.out.print(b); o/p: 1

Post decrementation

x=1;

System.out.print(x); o/p: 1

System.out.print(x--); o/p: 1

System.out.print(x); o/p: 0

a=1; b=2;

a=b--;

System.out.print(a); o/p: 2

System.out.print(b); o/p: 1

3) Assignment operators: - = , += , -= , \*= , /= , %=

a) This operator is used to store a value in to a variable.

EX: - int x=5;

b) It is used to store the value of a variable in to another variable.

EX: - int x=y;

c) It is used to store the value of an expression in to a variable.

EX: - int x = y + z - 4;

Expressions mean combination of a variables & values.

Note: - We can not write more than 1 variable at the left side of assignment.

EX: - x + y = 15; // invalid

Compact notation: -

EX: - num +=10;

sal \*=0.5;

p /=k;

i -=34;

y %=5;

These are the compact notations. These are shortcuts of java programs.

4) Relational operator: - < , > , <= , >= , == , !=

These operators are useful to compare to quantities. They are used in construction of condition.

EX: -

if(a>b)----------;

if(x!=1)---------;

if(sal>=5000.00)----;

if(a==b)------;

if(a<=10)----;

5) Logical operators: - && , | | , !

These operators are useful to create compound condition. The compound condition is condition of several simple conditions.

EX: -

If(a>b||b>c||c>d)-----;

Any one thing is true this will be executed;

If(x==10&&y!=10)----;

Both are true this will be executed.

If(!(x==!))------;

If(!(x==1 && y==2))----;

6) Boolean operators: - & , | , !

Boolean operators act upon Boolean variables and they return Boolean result

boolean a,b;

a=true;

b=false;

EX: -

a & b 🡪 false

a | b 🡪 true

a & a 🡪 true boolean type results

b | b 🡪false

! a 🡪 false

! b 🡪 true

7) Bitwise operators: - ~ , & , | , ^ , << , >> , >>>

These operators act up on individual bits binary digits, (0&1) of numbers.

Deci means 10. (0-9).

Ex: - Convert 45 into binary?

Rule: -Divide the decimal number successively by 2 and take the remainders from bottom to top.

2 45

2 22 - 1

2 11 - 0

= 1 0 1 1 0 1 🡪 8 bits

2 5 – 1 = 0 0 1 0 1 1 0 1

2 2 - 1

1

🡪 Convert 00101101 into decimal.

Rule:-Multiply each digit by the powers of 2 and take the sum of the products.

0 0 1 0 1 1 0 1

27  26 25 24 23 22 21 20

0 + 0 +32+0+8 + 4+ 0+1 = 45

-1

-2 3

-3 2

-4 1

0

Bitwise complement (~):-

This operator returns the complement from of a given number.

EX: - int x=10;

System.out.print(~x); o/p: - -11

Truth table: - Truth table is a table giving relationship between input bits & output bits. Anding means multiplication.

Bitwise and (&): - This operator performs anding operation on indusial bits of numbers.

AND Gate

And gate is a electronic circuit. It performs anding operations.

**X**

**0**

**0**

**1**

**1**

**Y**

**0**

**1**

**0**

**1**

**X & Y**

**0**

**0**

**0**

**1**

EX: -

x=10, y=11, x & y = ? ;

Convert binary

X = 0 0 0 0 1 0 1 0

Y = 0 0 0 0 1 0 1 1

0 0 0 0 1 0 1 0 = 10

.

. . x & y = 10.

Bitwise or ( | ): -

This operator performs oring operations on individual bits of numbers. Oring means both adding x & y.

OR Gate

EX: -

x=10, y=11, x & y = ? ;

Convert binary

**X**

**0**

**0**

**1**

**1**

**Y**

**0**

**1**

**0**

**1**

**X | Y**

**0**

**1**

**1**

**1**

X = 0 0 0 0 1 0 1 0

Y = 0 0 0 0 1 0 1 1

0 0 0 0 1 0 1 1 = 11

.

. . x | y = 11.

Bitwise xor ( ^ ) : - Here x means exclusive. ^ Means cap, carpet, circumflex. This operator performs exclusive orring operation on individual bits of numbers. In the input bits if odd number of ‘1’s are there in the output we will get ‘1’.

XOR Gate

**X**

**0**

**0**

**1**

**1**

**Y**

**0**

**1**

**0**

**1**

**X ^ Y**

**0**

**1**

**1**

**0**

EX: -

x=10, y=11, x ^ y = ? ;

Convert binary

X = 0 0 0 0 1 0 1 0

Y = 0 0 0 0 1 0 1 1

0 0 0 0 0 0 0 1 = 1

.

. . x ^ y = 1.

Bitwise left shift ( << ) : -

This operator shifts the bits of a number two words left .a specified number of times.

Ex:- x = 10 ;

x <<2 =?

X=10 🡪

**0**

**0**

**1**

**0**

**1**

**0**

**0**

**0**

X<<2 🡪

**0**

**0**

**0**

**0**

**1**

**0**

**1**

**0**

27  26 25 24 23 22 21 20

0 + 0 + 32 + 0 + 8 + 0 + 0 + 0 = 40

.

. . x<<2 = 40.

Bitwise Right Shift ( >> ): -

This operator shifts the bits of a number two words right a specified number of times.

Ex: - x=10;

x>>2 =?

X=10 🡪

**0**

**0**

**1**

**0**

**1**

**0**

**0**

**0**

X>>2 🡪

**0**

**0**

**1**

**0**

**0**

**0**

**0**

**0**

27  26 25 24 23 22 21 20

0 + 0 + 0 + 0 + 0 + 0 + 2 + 0 = 2

.

. . x>>2 = 2.

Bitwise zero-fill right shift (>>>): - It is the same as >>.

🡪 What is the difference between >> & >>>?

A) >> shift the bits 2 words right & properties the sign bit (0 represent +ve 1 represent –ve sign)

>>> also shift the bits 2 words right bit. It does not protect sign bit. It always fills the sign bit with 0.That reason it is called zero-fill right shift.

🡪 // bitwise operators

class Bits

{

public static void main(String[] args)

{

byte x,y;

x=10;

y=11;

System.out.println("~x="+(~x));

System.out.println("x & y="+(x&y));

System.out.println("x | y="+(x|y));

System.out.println("x ^ y="+(x^y));

System.out.println("x<<2="+(x<<2));

System.out.println("x>>2="+(x>>2));

System.out.println("x>>>2="+(x>>>2));

}

}

D:\psr\Core java>javac Bits.java

D:\psr\Core java>java Bits

~x=-11

x & y=10

x | y=11

x ^ y=1

x<<2=40

x>>2=2

x>>>2=2

Syntax: - Syntax means correct format of writing a statement.

8) Ternary operator or conditional operator ( ? , : ):-

This operator called ternary operator because it acts on three variables. Ternary means 3. This operator is also called conditional operator because it represents a conditional statement.

Syntax: -

var=exp1?exp2:exp3;

🡪 // conditional operators

class Conditional

{

public static void main(String[] args)

{

int x=10,y=11;

int max=(x>y)?x:y;

System.out.println("Maximum value is ="+max);

}

}

D:\psr\Core java>javac Conditinal.java

D:\psr\Core java>java Conditinal

Maximum value is=11

9) Dot operator ( . ) : - (Membership operator)

a) To refer to a class in a package

java.util.Date, java.io.BufferedReader

b) To refer to a method in a class

math.pow( ), emp.getName( ), br.read( )

c) To refer to a variable in a class

Employee.name( ), emp.eno( ), System.out

10) Instanceof operator: - To test whether an object belongs to a calss

EX: -

boolean x=emp instanceof Employee

Instants means object. It is a single word, so no space.

Executing the statements one by one is called sequential execution.

A program can write simple programs only using sequential execution. If we want to develop bigger & complex programs random execution we needed.

Executing the statements randomly & repeatedly is called random execution. It is useful to write a complex program. It is achieved using control statements.

Control statements: - Control statements are the statements which alter or changed the flow of execution. They are given below

1. if-else statement
2. do-while loop
3. while loop
4. for loop
5. for each loop
6. switch statement
7. break statement
8. continue statement
9. return statement

Statements can execute only one time but loop can execute several times repeatedly.

1) if-else statement: - This statement is useful to execute a task. Task means 1 or more statements depending upon whether a condition is true or not.

Syntax: - if(condition)

{

Statements 1;

}

else

{

Statements 2;

}

If the condition is true statements 1 is executed. Otherwise statements 2 are executed.

(or)

if(condition1)

{

Statements 1;

}

else

if(condition2)

{

Statements 2;

}

else

if(condition3)

{

Statements 3;

}

else

{

Statements 4;

}

(or)

if(comdition1)

if(condition2)

if(condition3)

statements1

else

statements2

else

statements3

else

statements4

🡪// test a number is +ve or -ve

class IfElseTest

{

public static void main(String[] args)

{

int num=5;

if(num==0)

System.out.println("It is Zero");

else if(num>0)

System.out.println("It is +ve");

else

System.out.println("It is -ve");

}

}

D:\psr\Core java>javac IfElseTest.java

D:\psr\Core java>java IfElseTest

It is +ve

2) do-while loop: - This loop repeatedly execute a group of statements as long as a condition is true.

Syntax: -

do

{

Statements;

}

while (condition);

🡪// generates no's from 1 to 10

class DoWhileTest

{

public static void main(String[] args)

{

int x=1;

do

{

System.out.print(x+“\t”);

x++;

}

while (x<=10);

}

}

D:\psr\Core java>javac DoWhileTest.java

D:\psr\Core java>java DoWhileTest

1 2 3 4 5 6 7 8 9 10

3) while loop: -This loop is useful to repeatedly execute a group of statements as long as a given condition is true.

Syntax: -

while(condition)

{

Statements;

}

🡪// even no's up to 10

class WhileTest

{

public static void main(String[] args)

{

int x=2;

while (x<=10)

{

System.out.println(x+“\t”);

x+=2;

}

}

}

D:\psr\Core java>javac WhileTest.java

D:\psr\Core java>java WhileTest

2 4 6 8 10

While loop is more efficient from do-while loop. Because will have better control right from beginning.

4) for loop: -

This is used to repeatedly execute a group of statements as long as a condition is true. For loop is suitable for when the user knows the exact number of repetitions.

Syntax: -

for(exp1;exp2;exp3)

{

Statements;

}

Exp1 is called initializing expression.

Exp2 is called conditional expression.

Exp3 is called modifying expression.

Note: - We can write for loop by eliminating exp1 or exp2 or exp3 or any 2 exp or all the 3 expressions.

🡪// generates no's from 1 to 100

class ForTest

{

public static void main(String[] args)

{

for(int i=0;i<=100;i++)

{

System.out.println(i+“\t”);

}

}

}

D:\psr\Core java>javac ForTest.java

D:\psr\Core java>java ForTest

1 2 3 4 5 6 7 8 9 10

11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30

31 32 33 34 35 36 37 38 39 40

41 42 43 44 45 46 47 48 49 50

51 52 53 54 55 56 57 58 59 60

61 62 63 64 65 66 67 68 69 70

71 72 73 74 75 76 77 78 79 80

81 82 83 84 85 86 87 88 89 90

91 92 93 94 95 96 97 98 99 100

Infinite loop is executed then press CTRL + C. That is executes for ever. Infinite loops are draw backs in a program.

(or)

🡪// generates no's from 1 to 50

class ForTest1

{

public static void main(String[] args)

{

int i=0;

for(;;)

{

System.out.println(i);

i++;

if(i>50)

break;

}

}

}

D:\psr\Core java>javac ForTest1.java

D:\psr\Core java>java ForTest1

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

Break is statement to come out from the loop.

Note: - It is possible to write a loop inside other loops, they are called nested loops.

Syntax: -

for(exp1;exp2;exp3)

{

for(exp1;exp2;exp3)

{

statements;

}

}

5) for each loop: - This loop is used to repeatedly executes a group of statements for each element of a collection. Collection means group of elements. Array means a group of elements & also java.util classes.

Syntax: -

for(var:collection)

{

statements;

}

🡪// for each loop

class ForEachTest

{

public static void main(String[] args)

{

int arr[]={10,33,56,20,90};

for(int x:arr)

{

System.out.println(x+”\t”);

}

}

}

D:\psr\Core java>javac ForEachTest.java

D:\psr\Core java>java ForEachTest

10 33 56 20 90

6) switch statement: - switch statement is useful to execute a particular task from several available tasks depending upon the value of a variable

Syntax: -

switch(var)

{

case value1:

Statements 1;

break;

case value2:

Statements 2;

break;

||

case value n:

Statements n;

break;

default :

Statements;

}

🡪// switch statement

class SwitchTest

{

public static void main(String[] args)

{

char color='g';

switch(color)

{

case 'r':

case 'R':

System.out.println("Red");

break;

case 'g':

case 'G':

System.out.println("Green");

break;

case 'b':

case 'B':

System.out.println("Blue");

break;

case 'w':

case 'W':

System.out.println("White");

break;

default:

System.out.println("No color");

}

}

}

D:\psr\Core java>javac SwitchTest.java

D:\psr\Core java>java SwitchTest

Green

Break is statement to come out from a switch statement.

Switch statement is useful to handle menu driven programs. Menu means a list of items or options for the user select.

7) break statement: -

a) it is useful to come out of a loop.

b) it is useful to come out of a switch statement.

c) it is used in a nested blocks to go to end of a block

Syntax: -

break blockname; // go to end of block

go to statement is used in to jump one statement to another statement.

🡪 Why goto statement are not available in java?

A) 1) goto statements lead to confusion for a programmer.

2) goto statement from infinite loops which are draw backs in a program.

3) The documentation of the program will become difficult with more goto’s are used. Documentation means preserving a copy of the program. Algorithm means a step of logic.

4) goto statements are not part of structured programming. Because of above reasons the goto statements are eliminates of java.

🡪// break test

class BreakTest

{

public static void main(String args[])

{

for(int i=0;i<=10;i++)

{

if(i==4)

break;

System.out.println(i);

}

System.out.println("break executed");

}

}

D:\psr\Core java>javac BreakTest.java

D:\psr\Core java>java BreakTest

0

1

2

3

break executed

Labeled break statement: - It can be used as a form of goto statement in java.

🡪// break to go to end of a block

class BreakTest1

{

public static void main(String[] args)

{

boolean x=true;

bl1:{

bl2:{

bl3:{

System.out.println("Block 3");

if(x=true)

break bl2;

}

System.out.println("Block 2");

}

System.out.println("Block 1");

}

System.out.println("Out of all");

}

}

D:\psr\Core java>javac BreakTest1.java

D:\psr\Core java>java BreakTest1

Block 3

Block 1

Out of all

8) continue statement: -

This statement is useful to continue the next repetition of a loop. When the statement will subsequent statements in the loop are not executed.

🡪// continue statement

class ContinueStatementTest

{

public static void main(String[] args)

{

for(int i=10;i>=1;i--)

{

if(i>5)

continue;

System.out.print(i+”\t”);

}

}

}

D:\psr\Core java>java ContinueStatementTest

D:\psr\Core java>java ContinueStatementTest

5 4 3 2 1

9) return statements: - it is used in 2 ways.

1) It is used to come out of a method to a calling method.

main( ) display( )

**display( )**

**return;**

Note: - If return is used in main the entire program is terminated.

2) return statement can be used to return some value to the calling method.

EX: - return x;

🡪// return in main( )

class ReturnTest

{

public static void main(String[] args)

{

int num=1;

System.out.println("Before return");

if(num==1)

return; // System.exit(0);

System.out.println("After return");

}

}

D:\psr\Core java>javac ReturnTest.java

D:\psr\Core java>java ReturnTest

Before return

System.exit(0) will terminate the program from any method.

🡪What is the difference between System.exit(0) and System.exit(1)?

A) System.exit(0) represents normal termination.

System.exit(1) represents termination due to an error.

🡪 // print odd no's up to 50

class Odd

{

public static void main(String args[])

{

int n=1;

System.out.println("odd no's are:");

while(n<100)

{

System.out.print("\t"+n);

n+=2;

}

}

}

D:\psr\Core java>javac Odd.java

D:\psr\Core java>java Odd

odd no's are:

1 3 5 7 9 11 13 15 17

19 21 23 25 27 29 31 33 35 37

39 41 43 45 47 49 51 53 55 57

59 61 63 65 67 69 71 73 75 77

79 81 83 85 87 89 91 93 95 97

99

🡪 // Test whether a number is even or odd

class EvenOROdd

{

public static void main(String[] args)

{

int a=20;

if(a%2==0)

System.out.println("It is even number");

else

System.out.println("It is odd number");

}

}

D:\psr\Core java>javac EvenOROdd.java

D:\psr\Core java>java EvenOROdd

It is even number

🡪 // given no is prime or not

class PrimeOrNot

{

public static void main(String args[])

{

int i,j,k,l;

i=9;

j=2;

l=1;

while(j<9)

{

k=i%j;

if(k==0)

l=k;

j++;

}

if(l==0)

{

System.out.println("not prime");

}

else

{

System.out.println("prime");

}

}

}

D:\psr\Core java>javac PrimeOrNot.java

D:\psr\Core java>java PrimeOrNot

not prime

🡪//display a multiplication table

class MultiflicationTable

{

public static void main(String args[])

{

int i=2,j;

System.out.println("Multiplication table is=:");

for(j=1;j<=10;j++)

{

System.out.println(i+"\*"+j+"="+(i\*j));

}

}

}

D:\psr\Core java>javac MultiflicationTable.java

D:\psr\Core java>java MultiflicationTable

Multiplication table is=:

2\*1=2

2\*2=4

2\*3=6

2\*4=8

2\*5=10

2\*6=12

2\*7=14

2\*8=16

2\*9=18

2\*10=20

🡪 /\* display the stars the following

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\*/

class StarsTest

{

public static void main(String args[])

{

int stars=1,lines=5,spaces=5;s

int i=1,j,k;

for(i=0;i<lines;i++)

{

// print spaces first

for(k=0;k<spaces;k++)

System.out.print(" ");

for(j=0;j<(stars);j++)

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

spaces--;

stars +=1;

System.out.println( );

}

}

}

D:\psr\Core java>javac StarsTest.java

D:\psr\Core java>java StarsTest

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

🡪 // print the Fibonacci series up to 10

class Fibonacci

{

public static void main(String args[])

{

int i=0;

int j=1,c;

System.out.print(i);

System.out.print("\t"+j);

for(int k=3;k<=10;k++)

{

c=i+j;

System.out.print("\t"+c);

i=j;

j=c;

}

}

}

D:\psr\Core java>javac Fibonacci.java

D:\psr\Core java>java Fibonacci

0 1 1 2 3 5 8 13 21 34

🡪 // print prime no’s

class PrimeSeries

{

public static void main(String args[])

{

int k,m,j;

for(j=2;j<100;j++)

{

k=0;

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

{

m=j%i;

if(m==0)

k=k+1;

}

if(k==2)

System.out.print("\t"+j);

}

}

}

D:\psr\Core java>javac PrimeSeries.java

D:\psr\Core java>java PrimeSeries

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Accepting data from keyboard: -

If you want to data read input from keyboard.

Stream:- A stream represent flow of data from one place to another place. Stream carries data. There are 2 types of streams.

1. In put streams: - in put streams read data or accept data.
2. Out put streams: - out put streams send or write data .All streams are represented by classes in java.io package.

System .in: - InputStream obj – keyboard. Here in means field.

System.out: - PrintStream obj – monitor

System.err: - PrintStream obj – monitor

🡪 What is difference between System.out & Sytem.err?

A) System.out is useful to display general messages.

System.err is used to display error messages. Stream is one of the io device.

Input/output devices are represented by streams. Streams are useful to achieve hardware independence in a java program.

InputStreamReader: -It is read data from keyboard.

BufferedReader: - We read the data at BufferdReader.

1. Connect keyboard to InputStreamReader

InputStreamReader isr=new InputStreamReader(System.in)

1. Connect InputStreamReader to BufferedReader

BufferedReader br=new BufferedReader(isr)

1. Read data from br using read( ) or readLine( ) methods.

🡪// accepting a char from keyboard

import java.io.\*;

class CharTest

{

public static void main(String[] args) throws IOException

{

// create BufferedReader obj

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("Enter a char : ");

char ch=(char)br.read( );

System.out.println("you entered : "+ch);

}

}

D:\psr\Core java>javac CharTest.java

D:\psr\Core java>java CharTest

Enter a char : 4

you entered : 4

Type casting: - Converting one data type into another data type is called type casting or casting. Writing the data type with in the simple braces, it is called caste operator. Cast means converting.

Exception represents run time errors. Throws Exception, throw out exception without handling it.

🡪// accepting a string from keyboard

import java.io.\*;

class StringTest

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("Enter a char : ");

String ch=br.readLine( );

System.out.println("you entered : "+ch);

}

}

D:\psr\Core java>javac StringTest.java

D:\psr\Core java>java StringTest

Enter a char : Usha

you entered : Usha

🡪// accepting integers from keyboard

import java.io.\*;

class IntTest

{

public static void main(String[] args) throws IOException

{

// create BufferedReader obj

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("Enter a number : ");

int n=Integer.parseInt(br.readLine( ));

System.out.println("you entered : "+n);

}

}

D:\psr\Core java>javac IntTest.java

D:\psr\Core java>java IntTest

Enter a number : 67

you entered : 67

🡪// accepting float numbers from keyboard

import java.io.\*;

class FloatTest

{

public static void main(String[] args) throws IOException

{

// create BufferedReader obj

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("Enter a number : ");

float n=Float.parseFloat(br.readLine( ));

System.out.println("you entered : "+n);

}

}

D:\psr\Core java>javac FloatTest.java

D:\psr\Core java>java FloatTest

Enter a number : 23

you entered : 23.0

🡪// accepting and displaying emp data

import java.io.\*;

class Empdata

{

public static void main(String[] args) throws IOException

{

// create BufferedReader obj

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

// accept employee data

System.out.print("Enter id : ");

int id=Integer.parseInt(br.readLine( ));

System.out.print("Enter sex : ");

char sex=(char)br.read( );

br.skip(2);

System.out.print("Enter name : ");

String name=br.readLine( );

System.out.println("Id ="+id);

System.out.println("Sex ="+sex);

System.out.prinlnt("Name ="+name);

}

}

D:\psr\Core java>javac Empdata.java

D:\psr\Core java>java Empdata

Enter id : 130

Enter sex : m

Enter name : Sathish

Id =130 Sex =m Name =Sathish

🡪 /\* accepting two no's from key board & find the result

Of their + , - , x , / \*/

import java.io.\*;

class Arithmetic

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter m value: ");

int m=Integer.parseInt(br.readLine( ));

System.out.print("enter n value: ");

int n=Integer.parseInt(br.readLine( ));

System.out.println("Addition is:="+(m+n));

System.out.println("Subtraction is:="+(m-n));

System.out.println("multiplication is:="+(m\*n));

System.out.println("Division is:="+(m/n));

}

}

D:\psr\Core java>javac Arithmetic.java

D:\psr\Core java>java Arithmetic

enter m value: 56

enter n value: 34

Addition is:=90

Subtraction is:=22

multiplication is:=1904

Division is:=1

🡪 // print the even no's between m & n

import java.io.\*;

class EvenNos

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter m value: ");

int m=Integer.parseInt(br.readLine( ));

System.out.print("enter n value: ");

int n=Integer.parseInt(br.readLine( ));

System.out.println("Even no's are:-");

do

{

int i=m%2;

if(i==0)

System.out.print("\t"+m);

m++;

} while(m<n);

}

}

D:\psr\Core java>javac EvenNos.java

D:\psr\Core java>java EvenNos

enter m value: 20

enter n value: 53

Even no's are: -

20 22 24 26 28 30 32 34 36 38 40 42

44 46 48 50 52

🡪 // Test whether a character is vowel or not

import java.io.\*;

class ChTest

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

char ch;

System.out.print("Enter any character: ");

ch=(char)br.read( );

if(ch=='a'||ch=='A'||ch=='e'||ch=='E'||ch=='i'||ch=='I'||ch=='o'||ch=='O'||ch=='u'||ch=='U')

System.out.println("Vowel");

else

System.out.println("Constant");

}

}

D:\psr\Core java>javac ChTest.java

D:\psr\Core java>java ChTest

Enter any character: s

Constant

D:\psr\Core java>java ChTest

Enter any character: o

Vowel

🡪 // test whether a number is perfect squire or not

import java.io.\*;

class PerefectSquare1

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

int i=0,x=0;

System.out.print("enter any number: ");

i=Integer.parseInt(br.readLine( ));

x=(int)Math.sqrt(i);

if(i==(x\*x))

System.out.println("Perfect square ");

else

System.out.println("Not a perfect Square ");

}

}

D:\psr\Core java>javac PerefectSquare1.java

D:\psr\Core java>java PerefectSquare1

Enter any number: 5

Not a perfect Square

D:\psr\Core java>java PerefectSquare1

Enter any number: 9

Perfect square

🡪 // given point lies inside the circle or out side the circle or on the circle

import java.io.\*;

class PointLies

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

int x=0,y=0,r=0;

System.out.println("Enter x & y co-ordinates : ");

x=Integer.parseInt(br.readLine( ));

y=Integer.parseInt(br.readLine( ));

System.out.println("Enter Radius :");

r=Integer.parseInt(br.readLine( ));

int a=(x\*x)+(y\*y);

int b=(r\*r);

if(a==b)

System.out.println("Point lies on the Circle");

else if(a>b)

System.out.println("Point lies out side the Circle");

else

System.out.println("Point lies in side the Circle");

}

}

D:\psr\Core java>javac PointLies.java

D:\psr\Core java>java PointLies

Enter x & y co-ordinates :

10

20

Enter Radius :

4

Point lies out side the Circle

🡪 // to find the Factorial value

import java.io.\*;

class Factorial

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

int fact=1;

System.out.print("Enter any Number : ");

int n=Integer.parseInt(br.readLine( ));

if(n==0)

System.out.println("Factorial of 0 is 1");

while(n>0)

{

fact=fact\*n;

n-=1;

}

System.out.println("Factorial ="+fact);

}

}

D:\psr\Core java>javac Factorial.java

D:\psr\Core java>java Factorial

Enter any Number : 5

Factorial =120

🡪 // sum of the digits

import java.io.\*;

class SumDigits

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

int n=0,sum=0,d=0;

System.out.print("Enter any Number : ");

n=Integer.parseInt(br.readLine( ));

do

{

d=n%10;

sum+=d;

n=n/10;

}

while(n>0);

System.out.println("Sum of digits ="+sum);

}

}

D:\psr\Core java>java SumDigits

Enter any Number : 236

Sum of digits =11

🡪 /\* all the 4 number digits Square of the 1st 2 digits are perfect squire & also last 2 digits \*/

import java.io.\*;

class PerfectSquare

{

public static void main(String args[])

{

int nl=0,nr=0,temp=0,x=0,y=0,z=0;

for(int i=1000;i<=9999;i++)

{

temp=i;

x=(int)Math.sqrt(temp);

{

if(temp==x\*x)

{

nl=temp/100;

nr=temp%100;

y=(int)Math.sqrt(nl);

z=(int)Math.sqrt(nr);

if((nl==y\*y)&&(nr==z\*z))

System.out.print(temp+”\t”);

}

}

}

}

}

D:\psr\Core java>javac PerfectSquare.java

D:\psr\Core java>java PerfectSquare

1600 1681 2500 3600 4900 6400 8100

Arrays: - An array represents a group of elements of same data type. They are 2 types.

1) Single dimensional array. 2) Multi dimensional arrays (2d,3d,……)

1) Single dimensional arrays (1d): -

A Single dimensional array represents is a single row or a single column of an elements.

Row - Horizontal arrangement, Column - Vertical arrangement.

For example marks obtained by a student in 5 subjects are Single dimensional array.

Creating a Single dimensional array: - There are 2 ways

1) We can declare a Single dimensional array & directly store the elements in to it.

EX: - int marks[ ]={50,55,99,86,72};

**50 55 99 86 72**

marks [0] [1] [2] [3] [4]

marks[i] 🡪 i is called index. Index represents the position number of the element in the array.

The Single dimensional array is only one index. That is i.

float sal[ ]={9000f,1200.50f,5678.34f};

char ch[ ]={‘a’,’b’,’c’,’d’};

String name[ ]={“raju”,”sudheer”,”sunil”,”latha”};

2) We can allot for Single dimensional array using new operator and later store the elements.

int marks[ ]=new int[5];

marks[0]=54; marks[1]=65; - - - - marks[4]=98;

double x[ ]=new double[100];

String name[ ]=new String[50];

Note: - Array name .length (is a property of any array) gives the size of the array.

array.length

🡪// total marks & percentage

import java.io.\*;

class SingleDimensionalArray

{

public static void main(String[] args) throws IOException

{

// to accept data from keyboard

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("How many subjects ? =");

int n=Integer.parseInt(br.readLine( ));

// create an array with size n

int marks[]=new int[n];

// store marks into marks[]

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

{

System.out.println("Enter marks :");

marks[i]=Integer.parseInt(br.readLine( ));

}

// display the marks & percentage

System.out.print("The marks are :");

int tot=0;

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

{

System.out.println(marks[i]);

tot+=marks[i];

}

System.out.println("Total marks = ”+tot);

// calculate percentage

float percent=(float)tot/n;

System.out.println("Percentage = "+percent);

System.out.println("Size of marks = "+marks.length);

}

}

D:\psr\Core java>javac SingleDimensionalArray.java

D:\psr\Core java>java SingleDimensionalArray

How many subjects ? =3

Enter marks :

36

Enter marks :

75

Enter marks :

97

The marks are : 36 75 97

Total marks = 208

Percentage = 69.333336

Size of marks = 3

2) Two dimensional array (2d): - A two dimensional array represents several rows & columns of data.

For example obtained by a group of students in 5 different subjects.

Creating Two dimensional array: - There are 2 ways

1. We can declare initialize a two dimensional array with elements.

int marks[ ] [ ]={{50,51,52,53,54},

{60,61,62,63,64},

{70,71,72,73,74}};

**54**

**64**

**74**

**53**

**63**

**73**

**52**

**62**

**73**

**51**

**61**

**72**

**50**

**60**

**70**

r0

r1

r2

c1 c2 c3 c4 c5

marks[i][j]🡪 indexes of marks.

A Two dimensional array has 2 indexes. They represent row position & column position numbers. A Two dimensional array is a combination of single dimensional arrays.

A three dimensional array has 3 indexes. A three dimensional array is a combination of several Two dimensional array.

2) We can created two dimensional arrays using new operator & stored the elements later

int marks[ ][ ]=new int[3][5];

float sal[ ][ ]=new float[5][10];

String str[ ][ ]=new String[3][10];

🡪// Two dimensional array

import java.io.\*;

class TwoDimensionalArray

{

public static void main(String[] args)

{

// take Two dimensional array

float x[][]={{1.1f,1.2f,1.3f,1.4f},

{2.1f,2.2f,2.3f,2.4f},

{3.1f,3.2f,3.3f,3.4f}};

// read from the Two dimensional array & display

for(int i=0;i<3;i++)

{

for(int j=0;j<4;j++)

{

System.out.print(x[i][j]+"\t");

}

System.out.println( );

}

}

}

D:\psr\Core java>javac TwoDimensionalArray.java

D:\psr\Core java>java TwoDimensionalArray

1.1 1.2 1.3 1.4

2.1 2.2 2.3 2.4

3.1 3.2 3.3 3.4

A two dimensional array represent a matrix.

Method parameter: - Method parameter is a variable that receive data from out side into the method.

Command line arguments: -

Command line arguments are values passed to main method from system prompt.

**java Arr 11 22 Sudheer Reddy**

Command line command line arguments

JVM allots memory in run time.

args.length🡪 Size of arguments

🡪// argument length example

class Argument

{

public static void main(String args[])

{

int n=args.length;

System.out.println("no. of arguments:="+n);

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

{

System.out.println(args[i]);

}

}

}

D:\psr\Core java>javac Argument.java

D:\psr\Core java>java Argument 12 34 54 Sunil Kumar Reddy

no. of arguments:=6

12

34

54

Sunil

Kumar

Reddy

🡪 // to find sum of two matrices

import java.io.\*;

class SumMatrix

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter r value : ");

int row=Integer.parseInt(br.readLine( ));

System.out.print("enter c value : ");

int col=Integer.parseInt(br.readLine( ));

int a[][]=new int[row][col];

System.out.println("Enter first matrix elements :\n");

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

a[i][j]=Integer.parseInt(br.readLine( ));

}

}

System.out.println("First matrix elements are:\n");

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

System.out.print(a[i][j]+"\t");

}

System.out.println( );

}

int b[][]=new int[row][col];

System.out.println("Enter second matrix elements :\n");

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

b[i][j]=Integer.parseInt(br.readLine( ));

}

}

System.out.println("Second matrix elements are:\n");

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

System.out.print(b[i][j]+"\t");

}

System.out.println( );

}

System.out.println("Sum matrix is : ");

int c[][]=new int[row][col];

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

c[i][j]=a[i][j]+b[i][j];

System.out.print(c[i][j]+"\t");

}

System.out.println( );

}

}

}

D:\psr\Core java>javac SumMatrix.java

D:\psr\Core java>java SumMatrix

enter r value : 2

enter c value : 3

Enter first matrix elements :

1

2

3

4

5

6

First matrix elements are:

1 2 3

4 5 6

Enter second matrix elements :

7

8

9

10

11

12

Second matrix elements are:

7 8 9

10 11 12

Sum matrix is :

8 10 12

14 16 18

🡪 // to find product of two matrices

import java.io.\*;

class Prodmatrix

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter r value : ");

int row1=Integer.parseInt(br.readLine( ));

System.out.print("enter c value : ");

int col1=Integer.parseInt(br.readLine( ));

int a[][]=new int[row1][col1];

System.out.println("Enter first matrix elements :\n");

for(int i=0;i<row1;i++)

{

for(int j=0;j<col1;j++)

{

a[i][j]=Integer.parseInt(br.readLine( ));

}

}

System.out.println("First matrix elements are:\n");

for(int i=0;i<row1;i++)

{

for(int j=0;j<col1;j++)

{

System.out.print(a[i][j]+"\t");

}

System.out.println( );

}

System.out.println("enter r value :");

int row2=Integer.parseInt(br.readLine( ));

System.out.println("enter c value : ");

int col2=Integer.parseInt(br.readLine( ));

int b[][]=new int[row2][col2];

System.out.println("Enter second matrix elements :\n");

for(int i=0;i<row2;i++)

{

for(int j=0;j<col2;j++)

{

b[i][j]=Integer.parseInt(br.readLine( ));

}

}

System.out.println("Second matrix elements are:\n");

for(int i=0;i<row2;i++)

{

for(int j=0;j<col2;j++)

{

System.out.print(b[i][j]+"\t");

}

System.out.println( );

}

if(row2==col1)

{

System.out.println("Prod matrix is : ");

int c[][]=new int[row1][col2];

for(int i=0;i<row1;i++)

{

for(int j=0;j<col2;j++)

{

for(int k=0;k<row2;k++)

{

c[i][j]+=a[i][k]\*b[k][j];

}

}

}

for(int i=0;i<row1;i++)

{

for(int j=0;j<col2;j++)

{

System.out.print(c[i][j]+"\t");

}

System.out.println( );

}

}

else

System.out.println("can not product the matrix");

}

}

D:\psr\Core java>javac Prodmatrix.java

D:\psr\Core java>java Prodmatrix

enter r value : 3

enter c value : 2

Enter first matrix elements :

1

2

3

4

5

6

First matrix elements are:

1 2

3 4

5 6

enter r value : 2

enter c value : 3

Enter second matrix elements:

1

2

3

4

5

6

Second matrix elements are:

1 2 3

4 5 6

Prod matrix is :

9 12 15

19 26 33

29 40 51

🡪 // to transpose a matrices

import java.io.\*;

class TransposeMatrix

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter r value :");

int row=Integer.parseInt(br.readLine( ));

System.out.print("enter c value : ");

int col=Integer.parseInt(br.readLine( ));

int a[][]=new int[row][col];

System.out.println("Enter matrix elements :\n");

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

a[i][j]=Integer.parseInt(br.readLine( ));

}

}

System.out.println("matrix elements are:\n");

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

{

System.out.print(a[i][j]+"\t");

}

System.out.println( );

}

System.out.println("Transpose matrix is : ");

for(int i=0;i<col;i++)

{

for(int j=0;j<row;j++)

{

System.out.print(a[j][i]+"\t");

}

System.out.println( );

}

}

}

D:\psr\Core java>javac TransposeMatrix.java

D:\psr\Core java>java TransposeMatrix

enter r value : 2

enter c value : 3

Enter matrix elements :

1

2

3

4

5

6

matrix elements are:

1 2 3

4 5 6

Transpose matrix is :

1 4

2 5

3 6

Strings: -strings represent a group of characters. In C or C++ strings are character arrays. \ 0 is null character. It is end of string.

In java strings are not character arrays. Strings are string class object in object. A string is an object of string class. Any string is an object of string class in java.

Creating strings: -

1) We can declare a string type variable & store directly in to it.

String str=”hello”;

2) We can create a string class object using new operator and pass the string to it.

String s1 =new String (“Hai”);

3) We can create a string by converting a character array in to a string object.

Char arr[ ] = {‘a’,‘b’,’c’,’d’,’e’,‘f’};

String s2 = new String (arr);

String s2=new String (arr,1,3);

String methods: -

Java.lang.string: -

1) String concat (String str): - Concat means method. A method can for form a task .Concatenates the contaning string with str.

Note: - +will also do the same

Concat is a method it joins two strings. When you call concat pass a string object to it.

Ex: - String s1 = “Hydera”;

String s2 = “bad”;

String x = s1. concat (s2);

Display Hyderabad.

+ is called concatenation operator.

Concat: - joining string at the end another string.

2) int length ( ):- returns the length of a string.

Ex:- String s1 = “Hydera”;

Int n = s1. length( );

3) char charAt(int index):-

Returns the character at the specified location (from 0).

4) int compare to (string Str): - Returns a negative value, if the calling Sting comes before str in dictionary order, a +ve value, if the String comes after str, or 0, if the String are equal.

EX: -

int n=s1.compareTo(s2);

if s1==s2, n=0

if s1>s2, n>0

if s1<s2, n<0

s1=”boy”, s2=”box”;

It is a +ve number. X is first, first is less value.

5) boolean equals(String str): - Returns true if the calling string equals str. Equals is case sensitive method.

String str=”abc”; String s2=”Xyz”;

If(str.equals(s2)) --- true other wise false

6) boolean equalsIgnoreCase(String str): - Sane as above. This is case insensitive method.

7) boolean startsWith(String prefix): - Returns true if the calling string starts with prefix. Prefix means the string starting with beginning.

String str=”hello Boy”;

startsWith(“h”); gives you true.

startsWith(“k”); gives you false.

1. boolean endsWith(String suffix): - Returns true if the invoking string ends with suffix. Suffix means the string ending with beginning.

String str=”hello Boy”;

endsWith(“y”); gives you true.

sendsWith(“k”); gives you false.

Note: - Above 2 methods use case sensitive comparison.

9) int indexOf(String str): - Returns the first occurrence of str in the string.

10) int lastIndexOf(String str): - Returns the last occurrence of str in the string.

Note: - Both the above methods return –ve value, if str not found in the calling string. Counting starts from 0.

Ex: - String str=”This is a book”;

2 5

int n=str.indexOf(“is”); // It gives 2 only because first position.

int n=str.lastIndexOf(“is”); // it gives 5 only.

1. String replace(char oldChar,char newChar): - Returns a new string. That is obtained by replacing characters old char in the string with new char.

String str=”Chennai”;

Str.replace(‘n’,’a’); will gives Cheaaai

1. String substring(int beginIndex): - Return a new string consisting of all characters from begin index until the end of the string.

String str=”Chennai”;

Str.substring(2) will print ennai

1. String substring(int beginIndex,int endIndex): - Returns a new string consisting of all characters from begin index until end index(exclusive). Last character is exclusive.

String str=”Blue Lotus”;

Str.substring(5,10); will print Lotus ( indexs from 5 to 10)

1. String toLowerCase( ): -Converts all characters into lower case.

String str=”HYDERABAD”;

Str.toLowerCase(); prints hyderbad in lower case letters

1. String toUpperCase( ): - Converts all characters into upper case.

String str=”Chennai”;

Str.toUpperCase(); prints CHENNAI in capital letters

1. String trim( ): - Eliminates all leading & trailing spaces.

Trim means deleting or cutting. Trim method doesn’t remove in the middle spaces. It’s only remove first spaces & last spaces. Last space is removing, it is trail method.

🡪// understanding the strings

class StringsTest

{

public static void main(String[] args)

{

// create 3 strings

String s1="This is java";

String s2=new String(" I like it");

char ch[]={'P','o','t','h','u','r','a','i'};

String s3= new String(ch);

// display the strings

System.out.println("S1 = "+s1);

System.out.println("S2 = "+s2);

System.out.println("S3 = "+s3);

// find no . of charecters in s1

System.out.println("length of S1 = "+s1.length( ));

// join 2 strings

System.out.println("S1 joined with s2 = "+s1.concat(s2));

// join 3 strings

System.out.println(s1+" at "+s3);

// check the starting of s1

boolean x=s1.startsWith ("This");

if(x==true)

System.out.println("s1 starts with This ");

else

System.out.println("S1 does not start with This ");

/\* extract substring from s1 & s3

String p=s2.subString(0,6);

String q=s3.subString(0);

System.out.println(p+q); \*/

//change the case of s3

System.out.println("Upper case of s3 = "+s3.toUpperCase( ));

System.out.println("Lower case of S3 = "+s3.toLowerCase( ));

}

}

D:\psr\Core java>javac StringsTest.java

D:\psr\Core java>java StringsTest

S1 = This is java

S2 = I like it

S3 = Pothurai

length of S1 = 12

S1 joined with s2 = This is java I like it

This is java at Pothurai

s1 starts with This

Upper case of s3 = POTHURAI

Lower case of S3 = pothurai

🡪// equality of strings

class StringEqual

{

public static void main(String args[])

{

String a="Hello";

String b=new String("Hello");

if(a==b)

System.out.println("same");

else

System.out.println("Not same");

}

}

D:\psr\Core java>javac StringEqual.java

D:\psr\Core java>java StringEqual

Not same

🡪 What is hash code?

1. It is a unique identification number that is allotted by JVM to the objects. This number is also called reference number.

S1

**Hello**

**125**

S1==S2

Reference numbers

S2

**Hello**

125!=130

**130**

So the out put is not same. == is only compare reference numbers

In the above program correct format is if(s1.equals(s2))

22🡪 What is the difference between == to and equals method while comparing the strings?

A) == operator compares only the reference of string objects.

Equals method compares the contents of the string objects. Hence the equals’ method gives reliable results.

23🡪 What is string constant pool?

A) It is separate block of memory where the string objects are stored.

**125**

**Hello**

Sting s1=”hello”

String s2=”hello”

**125**

125 is copy to another

Same reference no’s so

125==125 String constant pool

If(s1==s2)

o/p: - equal, because JVM does not waste memory.

24🡪 What is the difference between the following statements?

a) String s=”Hello”;

b) String s=new String(“Hello”);

A) When the first statement is executed JVM searches for the same object in the string constant pool. If the same object is found there then JVM will create another reference to the same object, if the same object not found JVM creates another object & stores it into string constant pool.

When the second statement is executed JVM always create a new object without searching in the string constant pool.

Types of objects: - 1) Mutable 2) Immutable.

1) Mutable: -

Mutable objects are the objects whose contents can be modified.

2) Immutable: -

Immutable objects are those objects whose contents can not be modified.

🡪// String class object are immutable

class Immutable

{

public static void main(String[] args)

{

String s1="hello";

String s2="hai";

s1=s1+s2;

System.out.println(s1);

}

}

D:\psr\Core java>javac Immutable.java

D:\psr\Core java>java Immutable

Hellohai

String buffer: - String buffer is a mutable, where as string is immutable.

Creating a string buffer: -

1. StringBuffer sb=new StringBuffer(“Hello”);
2. StringBuffer sb=new StringBuffer

java.lang. StringBuffer: -

1. StringBuffer append(x): - x may be int, float, double, char, string or StringBuffer. It will be append to the calling StringBuffer.
2. StringBuffer insert(int offset,x): - x may be int, float, double, char, string or StringBuffer. It will be inserted into the StringBuffer at offset. Insert method is insert a value.
3. StringBuffer delete(int start,int end): - removes the characters from start to end.
4. StringBuffer reverse( ): - Reverse the character sequence in the StringBuffer
5. String toString( ): -Converting StringBuffer into a string
6. int length( ): - Returns the length of the StringBuffer

🡪 What is the difference between a string & StringBuffer?

A) String objects are immutable. StringBuffer objects are mutable.

The methods which directly manipulate the data or not available in string class. Such methods are available in StringBuffer class.

🡪// display the full name

import java.io.\*;

class Mutable

{

public static void main(String[] args) throws IOException

{

// to accept data from keyboard

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("Enter sur name : ");

String sur=br.readLine( );

System.out.print("Enter mid name : ");

String mid=br.readLine( );

System.out.print("Enter last name : ");

String last=br.readLine( );

// create String Buffer object

StringBuffer sb=new StringBuffer( );

// append sur,last to sb

sb.append(sur);

sb.append(last);

// insert mid after sur

int n=sur.length( );

sb.insert(n,mid);

// display full name

System.out.println("Full name = "+sb);

System.out.println("In reverse ="+sb.reverse( ));

}

}

D:\psr\Core java>java Mutable

Enter sur name : Sunil

Enter mid name : Kumar

Enter last name : Reddy

Full name = SunilKumarReddy

In reverse =yddeRramuKlinuS

🡪 What is software requirement specification (SRS)?

A) SRS is a document that contains client requirements and strategies to fulfill those requirements. SRS is prepared by project leader.

27🡪 What is software design specification(SDS)?

A) SDS is a document that contains I/O screens, PSUDO code, DFD’s dataflow diagrams, Black end data bases etc.., it is prepared by module leader.

C, PASCAL, FORTRAN, COBAL 🡪 these languages are called procedural languages. Because they follow procedure oriented approach

The languages c++ & java are called object oriented languages. Because they follow object oriented approach.

🡪1) In procedure oriented approach a programmer thinks about the problem and its solution. We need another approach where the behavior of entire system is studied before developing the software.

(So we can create full flexed procedure systems).

🡪2) In procedure oriented approach when the code size exceeds 10,000 lines & before reaching 1,00,000 lines, at a particular level the programs are suddenly loosing control on the code. So we need another approach where we can develop bigger & complex projects.

🡪3) Programming in procedure oriented approach is completely un natural. We need another approach where programming reflex human being life.

Because of the above approaches they new approach called object orient approach is developed by computer scientists. It is also called OOPS. (Object oriented programming language).

Features of OOPS: -

1) Object/class: - An object is anything but exists in the real world. An object contains properties & actions. Properties are represented by variables, actions are represented by methods.

A class is a group name which specifies properties & actions of an object. A class also contains variables & methods.

2) Encapsulation: - Taking the data & methods as a single unit is called encapsulation. Class is example of encapsulation. It is a protective mechanism. This main advantage is one class method will mixed in another class.

3) Abstraction: -Hiding unnecessary data from the user called abstraction.

Ex: - Car

4) Inheritance: - Producing new class from existing class is called inheritance. Re using the code without re writing it is the advantage of inheritance.

5) Polymorphism: - Poly – many, morphs – forms

The ability to exist in different form is called polymorphism, if same method is called performing several tasks.

6) Message passing: - Calling a method & passing data to it is called message passing.

🡪 What is the difference between object oriented programming language & object based programming language?

A) Object oriented programming languages follow all the features of OOP’s.

EX: - C++, Java, Small Talk, Simula 67

Object based programming language follow all the features of OOP’s, expect inheritance.

EX: - Java script, VB script.

Class is a model for creating the object based on the class objects is created.

Instance of a class: - Instance means physically exists.

Objects are created by JVM on heap at run time dynamically. In java executing is created only in dynamic memory, no static memory in java.

🡪// a class & object

class Person

{

// properties instance vars

String name;

int age;

// action methods

public void talk( )

{

System.out.println("Hello I am "+name);

System.out.println("My age is "+age);

}

}

class InstanceTest

{

public static void main(String[] args)

{

// create an obj to person

Person p1;

p1=new Person( );

System.out.println(p1.hashCode( ));

p1.name=" Sudheer Reddy ";

p1.age=23;

p1.talk( );

}

}

D:\psr\Core java>javac InstanceTest.java

D:\psr\Core java>java InstanceTest

8567361

Hello I am Sudheer Reddy

My age is 23

Instance variables are variables whose copy is available in the object.

When the instance variables are not initialized by the programmer, java compiler adds some code to the class where it initializes. The instance variables with default values are shown below.

Class is also called a data type. It is called the user data type.

Data Type default values

int 0

float 0.0

double 0.0

char a space(single space store)

String null

Any class null

boolean false

Initializing the instance variables: -Initializing means storing starting value. Initializing the instance variables are various types.

Type1: - We can initialize one class instance variables in another class.

Private: - Private is called access specifiers. Access specifier is a key word that specifies how to access or read the member of a class or the class it self.

There are 4 access specifiers.

1. Private: - private members are not available out side the class.
2. Public: - public members are available any where out side the class.
3. Default: -Default members are available out side the class.

Note: - If the user does not use any access specifier then java compiler uses default specifier.

1. Protected: - protected members are available out side the class.

🡪 Private can not be used before the class name. In type1 initialization, if the instance variables are declared as private, they are not available to other classes.

🡪 /\* this example demonstrate how to pass an objects to a function & how to return object \*/

import java.io.\*;

class Distance

{

private int feet;

private float inches;

void showDistance( )

{

System.out.println(feet+"-"+inches);

}

void setDistance(int ft,float in)

{

fect=ft;

inches=in;

}

void getDistance( ) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter feet : ");

feet=Integer.parseInt(br.readLine( ));

System.out.print("enter inches : ");

inches=Float.parseFloat(br.readLine( ));

}

Distance addDistance(Distance d2)

{

Distance temp=new Distance( );

temp.feet=0;

temp.inches=inches+d2.inches;

while(temp.inches>=12)

{

temp.inches=temp.inches-12;

temp.feet++;

}

temp.feet=temp.feet+feet+d2.feet;

return temp;

}

}

class DistanceAddition

{

public static void main(String[] args) throws IOException

{

Distance d1=new Distance( );

Distance d2=new Distance( );

Distance d3=new Distance( );

d1.getDistance( );

d1.showDistance( );

d2.getDistance( );

d2.showDistance( );

d3=d1.addDistance(d2);

System.out.println("result Distance =");

d3.showDistance( );

}

}

D:\psr\Core java>javac DistanceAddition java

D:\psr\Core java>java DistanceAddition

enter feet : 10

enter inches : 20

10-20.0

enter feet : 20

enter inches : 10

20-10.0

result Distance =

32-6.0

Type2: - We can initialize the variables within the class.

In type2 initialization every object is initialized with same data. This type of initialization is useful to declare constants.

final double PI=3.14159;

To declare constants variables we use final key word.

Type3: - In this type we use constructors.

Constructors: - A constructor is similar to a method. Which is used to initialize the instance variables?

1. Constructors name & class name must be same.

EX: - Person( )

1. A constructor without parameters is called default constructors. A constructor with one or more parameters is called parameterized.

EX: -

🡪// default constructor

Person( )

{

name="Sudheer";

age=23;

}

A constructor does not return any value. A constructor is called & executed at the time of creating the object automatically. A constructor is only once for object.

EX: -

🡪// parametarized constructor

Person(String s,int i)

{

name=s;

age=i;

}

Person p2=new Person("Sudheer",23);

Person p3=new Person("Sunil",24);

🡪 //use of constructor

class Counter

{

int count;

Counter( )

{

count=0;

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

}

void showCount( )

{

System.out.println("count : "+count);

}

void incrementCount( )

{

count++;

}

void decrementCount()

{

count--;

}

}

class ConstructorDemo

{

public static void main(String[] args)

{

Counter c1=new Counter( );

Counter c2=new Counter( );

c1.incrementCount( );

c2.incrementCount( );

c2.incrementCount( );

c1.decrementCount( );

c2.decrementCount( );

c1.showCount( );

c2.showCount( );

}

}

D:\psr\Core java>javac ConstructorDemo.java

D:\psr\Core java>java ConstructorDemo

Constructor called

Constructor called

count : 0

count : 1

🡪 What is the default constructor & parameterized constructor?

A) Default constructor is used to initialize every object with same data.

Parameterized constructor is used to initialize each object with different data.

🡪 What is constructor overloading?

A) Writing 2 or more constructors with same name with difference in the parameter is called constructor overloading.

Methods: -

A method represent a group of stats with performs a task. A method will have 2 parts.

1) Method header (or) prototype: - It contain name of the method & method parameters (p1,p2,p3,..) & methods return type.

returntype methodname(para1,para2,……);

method parameters are variables which receive data from outside.

EX: -

return type 🡪 data type

void sum( )

double sum(double a,double b)

double sqrt(double x)

double power( )

long fact(int num)

void caluclateTsx(float sal)

2) Method body: - It represents a group of statements which perform the task.

EX: -method body

{

Statements;

}

Note: - If a method returns some value we should write return statement in the body.

EX: -

return x;

return 1;

return x+y;

return obj;

return arr;

Note: - return statement can return only one entity or 1 value it returns.

A method can not return more than one value.

🡪// understanding the methods

class Sample

{

// instance vars

double d1,d2;

// Para constructor

Sample(double a, double b)

{

d1=a;

d2=b;

}

// a method

void sum( )

{

double d3=d1+d2;

System.out.println("Sum = "+d3);

}

}

class MethodTest

{

public static void main(String[] args)

{

// sample object

Sample s=new Sample(20,30.5);

// call the method

s.sum( );

}

}

D:\psr\Core java>javac MethodTest.java

D:\psr\Core java>java MethodTest

Sum = 50.5

Local variable: - It is a variable that is declared locally within a method or constructor. The scope of that variable is limited with in method only or only to that constructor where it is declared.

Methods in java: -

1. Static methods
2. Instance methods
   1. Accessor methods
   2. Mutator methods
3. Factory methods

1) Static methods: -

These are the methods which don’t act upon the instance variables.

These are declared as static.

satic double sum(double d1,double d2)

Static methods are called using object name.method name

Static methods they can not read instance variables why because the reason is JVM first execute static methods & after that it creates a object next). So at the time of calling static method instance variables are not available. Static methods can acts as static variables. It should be declared as static.

🡪 What is different between instance variables and static variables?

A) 1) Instance variables are a variable whose separate copy is available to each object of the class. So any modification to the instance variable in one object will not modify the other objects.

Static variables are variables whose single copy is shared by all the objects of the class. If the static variable value is modified, it affects the value in all the objects.

2) Where instance variables & static variables are stored

Instance variables are created on heap.

Static variables are stored on method area.

Note: -

1) Static blocks, static methods & static variables are stored in method area & only one copy is shared by all objects.

2) A static block represents a block of statements declared as static.

3) JVM execute in the following order

* 1. static block
  2. static method
  3. instance method.

a) static block: -

🡪// static test

class StaticTest

{

static

{

System.out.println("static block");

System.exit(0);

}

}

(or)

🡪// static test

class StaticTest1

{

static

{

System.out.println("static block");

// System.exit(0);

}

public static void main(String[] args)

{

System.out.println("static method");

}

}

D:\psr\Core java>javac StaticTest1.java

D:\psr\Core java>java StaticTest1

static block

static method

🡪 // static block

class StaticBlock

{

static int a=3;

static int b;

static void show(int x)

{

System.out.println("x = "+x);

System.out.println("a = "+a);

System.out.println("b = "+b);

}

static

{

System.out.println("static block initialized");

b=a\*4;

}

}

class Static1Demo

{

public static void main(String[] args)

{

Static1 ob=new Static1( );

ob.show(100);

}

}

D:\psr\Core java>javac Static1Demo.java

D:\psr\Core java>java Static1Demo

static block initialized

x = 100

a = 3

b = 12

b) static method: -

🡪 // static method

class Static

{

int x;

static int y;

static void showy( )

{

System.out.println(y);

}

void showx( )

{

System.out.println(x);

}

}

class StaticMethod

{

public static void main(String[] args)

{

Static ob=new Static( );

ob.y=200;

ob.x=100;

ob.showx( );

ob.showy( );

}

}

D:\psr\Core java>javac StaticMethod.java

D:\psr\Core java>java StaticMethod

100

200

c) Instance methods: - These are those methods which act upon instance variables of a class. Instance methods are called using object name.

.methodName( )

EX: - s.sum( )

A method will return the result from where it is called.

EX: -double sum( )

{

double d3=d1+d2;

return d3;

}

calling

double res=s.sum( );

System.out.println("result = "+res);

These Instance methods are two types

* 1. Accessor methods: - Accessor methods read the instance variables. They don’t modify the instance variables.
  2. Mutator methods: - These methods not only read the instance variables but also modify them.

🡪// accessor & mutator methods

class Person

{

// instance variables

String name;

int age;

char sex;

// constructor to initialize

Person(String name,int age,char sex)

{

this.name=name;

this.age=age;

this.sex=sex;

}

// accessor method

void display( )

{

System.out.println("name= "+name);

System.out.println("age ="+age);

System.out.println("sex ="+sex);

}

// mutator method

Person modify(Person obj)

{

obj.name="Latha";

--obj.age;

obj.sex='f';

return obj;

}

}

class Mutate

{

public static void main(String args[])

{

Person p1=new Person("sudheer",23,'m');

p1.display( );

Person p2=p1.modify(p1);

p2.display( );

Person p3=p1.modify(new Person("sunil",30,'m'));

p3.display( );

}

}

D:\psr\Core java>javac Mutate.java

D:\psr\Core java>java Mutate

name= sudheer

age =23

sex =m

name= Latha

age =22

sex =f

name= Latha

age =29

sex =f

This: - this is a key word that refers to present class object.

It refers to 🡪 present class instance variables

* present class methods.
* present class constructor.

🡪//this demo

class Sample

{

int n;

Sample( )

{

this(55); // present class constructor

this.display( ); // present class method

}

Sample(int n)

{

this.n=n; // present class variable

}

void display( )

{

System.out.println("n="+n);

}

}

class ThisDemo

{

public static void main(String args[])

{

Sample s=new Sample( );

s.display( );

}

}

D:\psr\Core java>javac ThisDemo.java

D:\psr\Core java>java ThisDemo

n=55

n=55

3) Factory methods: - A factory method is a method that returns an object of the class to which to belong.

EX: -

NumberFormat obj=new NumberFormat.getInstance( );

All factory methods are static methods. But all static methods are not factory methods.

🡪 In how many ways can you create on object in java?

A) Ways of creating on object: -

1) Using new operator

Employee obj=new Employee( );

2) using factory method

NumberFormat obj=new NumberFormat.getInstance( );

3) using newInstance( ) method

Class c=Class.forName(“Employee”);

For name method will store employee that class name in an object

This is represented by c.

4) Using cloning( )

Cloning means creating bitwise exact copy of an existing object.

🡪// find the area of circle

import java.text.\*;

class Circle

{

public static void main(String args[])

{

double r=57.5;

final double PI=(double)22/7;

double area=PI\*r\*r;

System.out.println("area is ="+area);

// create NumberFormat obj

NumberFormat obj= NumberFormat.getNumberInstance();

// store format in to obj

obj.setMaximumFractionDigits(2);

obj.setMinimumIntegerDigits(4);

// apply this format into area

String str=obj.format(area);

// display formatted area

System.out.println("formatted area = "+str);

}

}

D:\psr\Core java>javac Circle.java

D:\psr\Core java>java Circle

area is =10391.07142857143

formatted area = 10,391.07

We can pass to array: -

int [ ] myMethod(int arr[ ])

int [ ] [ ] myMethod(int arr[ ] [ ])

return arr;

Array works like a reference similar to that of an object.

A[i][j] + b[i][j] 🡪 temp[i][j]

1. Read matrix1 elements from keyboard

Using getArray( )

1. Read matrix2 elements from keyboard

Using getArray( )

1. Add both matrices

Using findSum( )

1. Display the sum matrix

Using putArray( )

🡪// find sum of 2 matrices

import java.io.\*;

class Arr

{

// instance variables

int r,c;

int arr[][];

// constructor

Arr(int r,int c)

{

this.r=r;

this.c=c;

arr=new int[r][c];

}

// to receive matrix elements from key board

int [][] getArray( ) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

for(int i=0;i<r;i++)

{

for(int j=0;j<c;j++)

{

System.out.print("Enter element : ");

arr[i][j]=Integer.parseInt(br.readLine( ));

}

}

return arr;

}

// to find sum of 2 matrices

int [][] findSum(int a[][],int b[][])

{

int temp[][]=new int[r][c];

for(int i=0;i<r;i++)

{

for(int j=0;j<c;j++)

{

temp[i][j]=a[i][j]+b[i][j];

}

}

return temp;

}

// display the sum matrix

void putArray(int res[][])

{

for(int i=0;i<r;i++)

{

for(int j=0;j<c;j++)

{

System.out.print(res[i][j]+"\t");

}

System.out.println("\n");

}

}

}

class MatrixSum

{

public static void main(String[] args) throws IOException

{

// create 2 objects to arr

Arr obj1=new Arr(3,3);

Arr obj2=new Arr(3,3);

// take 3 array referances

int x[][],y[][],z[][];

System.out.println("enter matrix 1 :");

x=obj1.getArray( );

System.out.println("enter matrix 2 :");

y=obj2.getArray( );

z=obj1.findSum(x,y);

System.out.println("\n The sum matrix is : ");

obj2.putArray(z);

}

}

D:\psr\Core java>javac MatrixSum.java

D:\psr\Core java>java MatrixSum

enter matrix 1 :

Enter element : 1

Enter element : 2

Enter element : 3

Enter element : 4

Enter element : 5

Enter element : 6

Enter element : 7

Enter element : 8

Enter element : 9

enter matrix 2 :

Enter element : 1

Enter element : 2

Enter element : 3

Enter element : 4

Enter element : 5

Enter element : 6

Enter element : 7

Enter element : 8

Enter element : 9

The sum matrix is :

2 4 6

8 10 12

14 16 18

🡪 // find product of 2 matrices

import java.io.\*;

class Arr1

{

// instance variables

int r1,c1;

int arr1[][];

// constructor

Arr1(int r1,int c1)

{

this.r1=r1;

this.c1=c1;

arr1=new int[r1][c1];

}

// to receive matrix elements from key board

int [][] getArray( ) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

for(int i=0;i<r1;i++)

{

for(int j=0;j<c1;j++)

{

System.out.print("Enter element :");

arr1[i][j]=Integer.parseInt(br.readLine( ));

}

}

return arr1;

}

}

class Arr2

{

// instance variables

int r2,c2;

int arr2[][];

// constructor

Arr2(int r2,int c2)

{

this.r2=r2;

this.c2=c2;

arr2=new int[r2][c2];

}

// to receive matrix elements from key board

int [][] getArray( ) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

for(int i=0;i<r2;i++)

{

for(int j=0;j<c2;j++)

{

System.out.println("Enter element :");

arr2[i][j]=Integer.parseInt(br.readLine( ));

}

}

return arr2;

}

}

class Prod

{

int r1,c2,c1;

Prod(int r1,int c2,int c1)

{

this.r1=r1;

this.c2=c2;

this.c1=c1;

}

// to find Prod of 2 matrices

int [][] findProd(int a[][],int b[][])

{

int temp[][]=new int[r1][c2];

for(int i=0;i<r1;i++)

{

for(int j=0;j<c2;j++)

{

for(int k=0;k<c1;k++)

{

temp[i][j]+=a[i][k]\*b[k][j];

}

}

}

return temp;

}

// display the Prod matrix

void putArray(int res[][])

{

for(int i=0;i<r1;i++)

{

for(int j=0;j<c2;j++)

{

System.out.print(res[i][j]+"\t");

}

System.out.println("\n");

}

}

}

class MatrixProd

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("Enter first matrix r1,c1");

int r1=Integer.parseInt(br.readLine( ));

int c1=Integer.parseInt(br.readLine( ));

System.out.println("Enter second matrix r2,c2");

int r2=Integer.parseInt(br.readLine( ));

int c2=Integer.parseInt(br.readLine( ));

if(c1==r2)

{

// create 3 objects to arr

Arr1 obj1=new Arr1(r1,c1);

Arr2 obj2=new Arr2(r2,c2);

Prod p=new Prod(r1,c2,c1);

// take 3 array referances

int x[][],y[][],z[][];

System.out.println("\n enter matrix 1 :");

x=obj1.getArray( );

System.out.println("\n enter matrix 2 :");

y=obj2.getArray( );

z=p.findProd(x,y);

System.out.println("\n The prod matrix is : ");

p.putArray(z);

}

else

System.out.println("can not product of matrix");

}

}

D:\psr\Core java>javac MatrixProd.java

D:\psr\Core java>java MatrixProd

Enter first matrix r1,c1

3

2

Enter second matrix r2,c2

2

3

enter matrix 1 :

Enter element : 1

Enter element : 2

Enter element : 3

Enter element : 4

Enter element : 5

Enter element : 6

enter matrix 2 :

Enter element : 1

Enter element : 2

Enter element : 3

Enter element : 4

Enter element : 5

Enter element : 6

The prod matrix is :

9 12 15

19 26 33

29 40 51

🡪 // write a method to swap to given values

import java.io.\*;

class Swap

{

int a,b,c;

Swap(int x,int y)

{

this.a=x;

this.b=y;

c=a;

a=b;

b=c;

System.out.println("a="+a+"\t b="+b);

}

}

class SwapMethod

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter a value :");

int a=Integer.parseInt(br.readLine( ));

System.out.print("enter b value : ");

int b=Integer.parseInt(br.readLine( ));

System.out.println("before swapping");

System.out.println("a="+a+"\tb="+b);

System.out.println("After swapping");

Swap s=new Swap(a,b);

}

}

D:\psr\Core java>javac SwapMethod.java

D:\psr\Core java>java SwapMethod

enter a value : 25

enter b value : 56

before swapping

a=25 b=56

After swapping

a=56 b=25

What are class variables? : -

A) Static variables are also called class variables.

🡪// relation between two classes & class one obj

class One

{

int x;

Two t;

One(Two t)

{

this.t=t;

x=10;

}

void show( )

{

System.out.println("x= "+x);

t.show( );

System.out.println("y= "+t.y);

}

}

class Two

{

int y;

Two( )

{

y=22;

}

void show( )

{

System.out.println("y= "+y);

}

}

class Relate

{

public static void main(String args[])

{

Two obj2=new Two( );

One obj1=new One(obj2);

obj1.show( );

}

}

D:\psr\Core java>javac Relate.java

D:\psr\Core java>java Relate

x= 10

y= 22

y= 22

🡪 What is object graph?

1. A graph connecting several objects is called object graph.

X

Y

🡪 We can create class 2’s references as an instance variable in class one. When the object is created from obj1 we can refer to all the members of obj2. This is advantage of giving relationship.

Inner class: -

Inner class is a class written in another class. It is a security mechanism.

We can use private in inner class. Private keyword can be used before inner class only. Fire wall means optical that checks the authorization of user.

Out side programmer

**inner class**

Fire wall

🡪// inner class demo

class BankAcct

{

private double bal;

BankAcct(double b)

{

bal=b;

}

void start(double r)

{

Interest in=new Interest(r);

in.calculateInterest( );

}

private class Interest

{

private double rate;

Interest(double r)

{

rate=r;

}

void calculateInterest( )

{

System.out.println("Balance = "+bal);

double interest=bal\*rate/100;

System.out.println("interest = "+interest);

bal+=interest;

System.out.println("New Balance = "+bal);

}

}

}

class InnerDemo

{

public static void main(String args[])

{

BankAcct account=new BankAcct(20000);

account.start(7.5);

}

}

D:\psr\Core java>javac InnerDemo.java

D:\psr\Core java>java InnerDemo

Balance = 20000.0

interest = 1500.0

New Balance = 21500.0

🡪 1) Inner class is a written within another class.

2) It is a security mechanism.

3) Inner class in hidden in outer class from other classes.

4) Only inner class can be ‘private’

5) An object to inner class can not be created in any other class.

6) An object to inner class can be created only in its outer class.

7) Outer class object and inner class objects are created separately in memory.

8) Outer class members are available to inner class.

9) Inner class object contains an additional invisible field ‘this $0’ that contains outer class reference.

**bal**

**rate**

**this$0**

Bank acct

Interest

10) if same names are used for members ,then outer class members can be referenced in inner class as: outer class.this.member;

Ex: -bankAcct.this.bal.

Inner class members are referenced as: this.member

Ex: - this.bal

Inheritance: - Creating new class from existing classes such that all the features of existing classes are available to the new classes is called inheritance. Already existing class is called super class & produced class is called sub class.

Syntax: -

class subclass extends super class

🡪// inheritance example of Teacher & student class

class Teacher

{

int id;

String name;

String address;

float sal;

void setId(int id)

{

this.id=id;

}

int getId( )

{

return id;

}

void setName(String name)

{

this.name=name;

}

String getName( )

{

return name;

}

void setAddress(String address)

{

this.address=address;

}

String getAddress( )

{

return address;

}

void setSal(float sal)

{

this.sal=sal;

}

float getSal( )

{

return sal;

}

}

D:\psr\Core java>javac Teacher.java

🡪// inheritance example of Teacher & student class

class Student extends Teacher

{

int marks;

void setMarks(int marks)

{

this.marks=marks;

}

int getMarks( )

{

return marks;

}

}

D:\psr\Core java>javac Student.java

🡪// inheritance example of Teacher & student class

class StudentDemo

{

public static void main(String args[])

{

// create student class object

Student s=new Student( );

// store data into s

s.setId(100);

s.setName("Sudheer");

s.setAddress("Naghireddy Palli, AP, ATP");

s.setMarks(662);

// get data forms

System.out.println("ID = "+s.getId( ));

System.out.println("Name = "+s.getName( ));

System.out.println("Address = "+s.getAddress( ));

System.out.println("Marks = "+s.getMarks( ));

}

}

D:\psr\Core java>javac StudentDemo.java

D:\psr\Core java>java StudentDemo

ID = 100

Name = Sudheer

Address = Naghireddy Palli, AP, ATP

Marks = 662

🡪 Redundancy of the code is a draw back of programming.

Extends is key word used in inheritance. The subclass object contains object contain a copy of super class object in it. We cal create on object to only sub class.

🡪 What is the advantage of inheritance?

A) In inheritance a programmer can re use already developed code. Because of these developing new programs will become easy. This increases productivity of programmer & hence the over all productivity of the company or organization will also increase.

🡪// if super has a default constructer

class One

{

One( )

{

System.out.println("super");

}

}

class Two extends One

{

Two( )

{

System.out.println("sub");

}

}

class Const

{

public static void main(String args[])

{

Two t=new Two( );

}

}

D:\psr\Core java>javac Const.java

D:\psr\Core java>java Const

super

sub

Super class default constructor is available to sub class by default.

Super class parameterized constructor is not available to sub class by default. Super is a key word that refers to super class. For example

🡪// super used super vars, methods & constructors

class One

{

int x;

One(int x)

{

this.x=x;

}

void show( )

{

System.out.println("x = "+x);

}

}

class Two extends One

{

int y;

Two(int a,int b)

{

super(a); // (or) x=a;

y=b;

}

void show( )

{

System.out.println("x = "+super.x);

System.out.println("y = "+y);

}

}

class Const1

{

public static void main(String args[])

{

Two t=new Two(10,24);

t.show( );

}

}

D:\psr\Core java>javac Const1.java

D:\psr\Core java>java Const1

x = 10

y = 24

Super key word is used in sub class only. Calling constructor first declare the super.

Polymorphism: - It represents many forms. If a method performs several tasks, it is called polymorphism. They are 2 types

1) Dynamic polymorphism 2) Static polymorphism

1) Dynamic polymorphism: -

The polymorphism exhibited at run time is called dynamic polymorphism. In this dynamic polymorphism a method call is linked with method body at the time of running the program by JVM. This is also known as dynamic binding or run time polymorphism.

EX: -

Method over loading: - Writing 2 or more methods with the same name, but with a difference in the method signatures is called method over loading.

In method over loading JVM understand in the method is called depending upon the difference in the method signature. The difference may be due to the following.

1. There is a difference in the no. of parameters.

void add(int a,int b)

void add(int a,int b,int c)

1. There is a difference in the data types of parameters.

void add(int a,float b)

void add(double a,double b)

1. There is a difference in the sequence of parameters.

void swap(int a,char b)

void swap(char a,int b)

Any one of the above differences will make method signature to be different, hence JVM can identifies methods uniquely.

🡪Method over loading is an example for dynamic polymorphism.

🡪// sum of 2 numbers & sum of 3 numbers

class Sample

{

void add(int a,int b)

{

System.out.println("sum of two="+(a+b));

}

void add(int a,int b,int c)

{

System.out.println("sum of three="+(a+b+c));

}

}

class Poly

{

public static void main(String[] args)

{

Sample s=new Sample( );

s.add(20,25);

s.add(20,25,30);

}

}

D:\psr\Core java>javac Poly.java

D:\psr\Core java>java Poly

sum of two=45

sum of three=75

🡪 What is method signature?

A) Method name & its parameters is called method signature

Method overriding: - writing 2 or more methods in super & sub classes with same name & same signatures is called method overriding. In method overriding JVM executes a method depending on the data type of the object.

🡪Method overriding is an example for dynamic polymorphism

🡪//dynamic polymorphism

class One

{

void calculate(double x)

{

System.out.println("Square value ="+(x\*x));

}

}

class Two extends One

{

void calculate(double x)

{

System.out.println("Square root ="+Math.sqrt(x));

}

}

class Poly1

{

public static void main(String args[])

{

Two t=new Two( );

t.calculate(16);

}

}

D:\psr\Core java>javac Poly1.java

D:\psr\Core java>java Poly1

Square root =4.0

🡪 What is the difference between method over loading & method overriding?

A) Method over loading: - Writing 2 or more methods with the same name, but with a difference in the method signatures is called method over loading. In method over loading JVM understand in the method is called depending upon the difference in the method signature.

Method overriding: - writing 2 or more methods in super & sub classes with same name & same signatures is called method overriding. In method overriding JVM executes a method depending on the data type of the object.

Achieving method overloading & method overriding using instance methods is an example of dynamic polymorphism.

Achieving method overloading & method overriding using instance methods is an example of dynamic polymorphism.

2) Static polymorphism: - The polymorphism exhibited at compile time is called Static polymorphism. Here the compiler knows which method is called at the compilation. This is also called compile time polymorphism or static binding.

Achieving method overloading & method overriding using

1) private 2) static 3) final methods, is an example of Static polymorphism.

Note: - A final method is a method written in a final class. A class is declared as a final is called final class.

Ex: - final class A

Note: - final key word before class name prevents inheritance. We can create sub classes to a final class.

Ex: - final class A

class B extends A // invalid

🡪 We can not override final methods, only final methods overloading.

Converting 1 data type into another data type is called casting.

🡪 What is final?

A) 1) final is used to declare Constant

Ex: - final double PI 3.14159

2) final key word is used to prevent inheritance.

Ex: - final class A

class B extends A // invalid

Data type: - A data type represents types of data stored into a variable. These are 2 types.

1) Primitive data type: - (fundamental data types)

These are represents single values; methods are not available to handle them.

Ex: - char, byte, short, int, long, float, double, Boolean

2) Advanced data types: - (Referenced data types)

These are represents several value methods also available to handle them.

Ex: - int array, any classes (ex: employee, string……)

🡪 What is difference between primitive & advanced data types?

A) 1) Casting can be used to convert one primitive data type in to another primitive data type.

2) Casting can be used to convert one advanced data type into another advanced data type.

3) Casting can not be used to convert a primitive data type into an advanced data type and vice versa (means reverse also).

For this purpose we can use wrapper classes.

Casting primitive data types: -

a) Widening: - Casting a lower data type into a higher data type is called widening.

EX: - char, byte, short, int, long, float, double

Lower 🡨------------🡪higher

i) char ch=’A’;

int n=(int)ch;

ii) int num=15;

float f=(float)num;

In widening no digits or precision are lost.

b) Narrowing: - Converting a higher data type into lower type is called narrowing.

EX: - i) int n=66;

Char ch=(char)n;

ii) double d=12.1234;

int n=(int)d;

In narrowing precision or digits are lost. This is called explicit casting.

Casting advanced data types:

We can cast advanced data types provide there is relationship between the classes by the way of inheritance.

🡪//Casting advanced data types:

class One

{

void show1( )

{

System.out.println("Super class");

}

}

class Two extends One

{

void show2( )

{

System.out.println("Sub class");

}

}

class Cast

{

public static void main(String args[])

{

// super class reference to refer to super class object

One o;

o=new One( );

o.show1( );

}

}

D:\psr\Core java>javac Cast.java

D:\psr\Core java>java Cast

Super class

In super reference is used to refer to super class object. The programmer can refer to only super class members but not the sub class members

🡪

class Cast

{

public static void main(String args[])

{

// sub class reference to refer to sub class object

One o=new Two( );

Two t=(Two)o;

t.show1( );

t.show2( );

}

}

D:\psr\Core java>javac Cast.java

D:\psr\Core java>java Cast

Super class

Sub class

In sub class reference is used subclass object; the programmer can access all the members of both the super class& sub class.

🡪

class Cast

{

public static void main(String args[])

{

// super class reference to refer to sub class object

One o;

o=new Two( );

o.show1( );

}

}

D:\psr\Core java>javac Cast.java

D:\psr\Core java>java Cast

Super class

What is generalization & specialization?

A) Moving back from sub class to super class is called generalization. Casting a sub class type into a super class is called a up casting or widening.

Coming down from super class to sub class is called specialization. Casting a super class type into sub class type is called narrowing or down casting.

In widening we can access super class methods. In widening we can access sub class methods. In widening, we can not access sub class methods unless the override super class methods.

🡪

class Cast

{

public static void main(String args[])

{

// sub class reference to refer to super class object

One o=new Two( );

Two t=(Two)o;

t.show1( );

t.show2( );

}

}

D:\psr\Core java>javac Cast.java

D:\psr\Core java>java Cast

Super class

Sub class

Narrowing using super class object can not access either super class methods or sub class methods.

Narrowing using sub class object will make the programmer to access are the members of super class as well as sub class.

🡪 What is widening & narrowing?

A) Widening: - Casting a lower data type into a higher data type is called widening. In widening no digits or precision are lost. Casting a sub class type into a super class is called an up casting or widening. In widening we can access super class methods. In widening we can access sub class methods. In widening us can not access sub class methods unless the override super class methods.

Narrowing: - Converting a higher data type into lower type is called narrowing. In narrowing precision or digits are lost. This is called explicit casting. Casting a super class type into sub class type is called narrowing or down casting. Narrowing using super class object can not access either super class methods or sub class methods. Narrowing using sub class object will make the programmer to access are the members of super class as well as sub class.

Abstract methods: - When different objects need different implementation of same method then we can not write body for the method. Such a method is called abstract method. It is a method without body. Abstract method should be declared with abstract key word. A class that contains abstract methods is called abstract class. Abstract class should also be declared using the abstract key word. We can not create object to abstract class.

🡪// abstract class

abstract class Car

{

// every car will have a regno

int regno;

// to store regno

Car(int r)

{

regno=r;

}

// every car will have fuel tank

// mechanism to fill the fuel is same for all cars

void fillTank( )

{

System.out.println("Use car Key");

System.out.println("And fill the tank");

}

/\* every car will have steering. But different cars will have different steering mechanisms \*/

abstract void steering(int direction);

/\* every cars will have breaks. But different cars will have different breaking mechanisms \*/

abstract void braking(int force);

}

D:\psr\Core java>javac Car.java

🡪 // this is a concrete sub class of car class

class Maruthi extends Car

{

Maruthi(int regno)

{

super(regno);

}

void steering(int direction)

{

System.out.println("Maruthi uses manual steering");

System.out.println("please drive the car");

}

void braking(int force)

{

System.out.println("Maruthi uses hydraulic breaks");

System.out.println("apply breaks & stop the car");

}

}

D:\psr\Core java>javac Maruthi.java

🡪 // this is a concrete sub class of car class

class Santro extends Car

{

Santro(int regno)

{

super(regno);

}

void steering(int direction)

{

System.out.println("Santro uses power steering");

System.out.println("Start it");

}

void braking(int force)

{

System.out.println("Santro uses gas breaks");

System.out.println(" stop it");

}

}

D:\psr\Core java>javac Santro.java

🡪 // using the car

class UseCar

{

public static void main(String[] args)

{

// create a maruthi & Santro obj

Maruthi m=new Maruthi(6666);

Santro s=new Santro(2222);

// create a reference of car class

Car ref;

// use ref to refer to maruthi

ref=m;

ref.fillTank( );

ref.steering(2);

ref.braking(200);

// use ref to refer to Santro

ref=s;

ref.fillTank( );

ref.steering(4);

ref.braking(400);

}

}

D:\psr\Core java>javac UseCar.java

D:\psr\Core java>java UseCar

Use car Key

And fill the tank

Maruthi uses manual steering

please drive the car

Maruthi uses hydraulic breaks

apply breaks & stop the car

Use car Key

And fill the tank

Santro uses power steering

Start it

Santro uses gas breaks

stop it

Reference statement is used to element the representation of statements.

🡪 1) An abstract class is class with zero or more abstract methods

2) An abstract class contains instance variables & concrete methods in addition to abstract methods.

3) It is not possible to create objects to abstract class.

4) But we can create a reference of abstract class type.

5) All the abstract methods of the abstract class should be implemented in its sub classes.

6) If any method is not implemented, then that sub class should be declared as ‘abstract’.

7) Abstract class reference can be used to refer to the objects of its sub classes.

8) Abstract class references cannot refer to the individual methods of sub classes.

9) A class can not be both ‘abstract’ & ‘final’.

EX: - final abstract class A // invalid

🡪 How can enforce discipline in the programmers to implement only your class features?

A) By writing an abstract class or an interface I can enforce.

Interfaces: - An interface is a specification of method properties. We can not create on object in interface. We create reference variable for interface.

Interface class

Implementation class implementation class

Driver: - A driver represents implementation classes of an interface.

**Oracle**

**SyBase**

**My Sql**

**void connect( );**

**{**

**--------**

**}**

API

🡪 // interface example

interface MyInter

{

void connect( );

}

class OracleDB implements MyInter

{

public void connect( )

{

System.out.println("Connecting to oracle data base .....");

}

}

class SybaseDB implements MyInter

{

public void connect( )

{

System.out.println("Connecting to Sybase data base .....");

}

}

class DataBase

{

public static void main(String args[]) throws Exception

{

// accapt user dats base name through command line arguments store in an object c

Class c=Class.forName(args[0]);

// create another object to the class whose name is in c

MyInter mi=(MyInter)c.newInstance( );

// call convert method of the object

mi.connect( );

}

}

D:\psr\Core java>javac DataBase.java

D:\psr\Core java>java DataBase SybaseDB

Connecting to sybase data base.....

D:\psr\Core java>java DataBase OracleDB

Connecting to oracle data base.....

**OracleDB**

I

**C**

**mi**

**connect( )**

II

🡪 1) An interface is a specification of method prototype.

2) An interface contains 0 or more abstract methods.

3) All the methods as the interface are public and abstract by default.

4) An interface can also contain variables which are public, static and final by default.

5) We can not create an object to an interface.

6) But we can create a reference variable of interface type.

7) All the methods of the interface should be implemented in its implementation classes.

8) If any method is not implemented, then that class should be declared as ‘abstract’.

9) Interface reference can be used to refer to all the objects of its implementation classes.

10) Once an interface is written, any third party vendor can provide implementation classes.

11) An interface can not implement another interface.

12) An interface can extend another interface.

13) We can create classes in an interface.

14) A class can implement multiple interfaces.

Class A implements B,C…

What is the difference between abstract & interface? : -

A) Interface contains only abstract methods. Abstract class contains instance variables. Interface contains constant. The abstract method should be implemented in its sub classes. All the methods of interface should be implemented in implemented classes. An abstract class is declared using the key word ‘abstract’. Interface is declared using ‘interface’.

\* A programmer writes an abstract class when there are some common features shared by all the objects.

\* A programmer writes an interface when all the features should be implemented differently in different objects.

When abstract class is written the programmer should provide the sub class.

When interface is written any third party vendor can provide implementation class.

🡪 How much memory is taken by class two’s object?

1) 4 b 2) 8 b 3) >8 b 4) <8 b

A) Ans is >8 b because it is also taken memory of class two

Ex: -

**int x**

class One

**int x**

{

int x;

}

class Two extends One

{ 4 + 4

int x;

} 8

Types of inheritance: -

1) Single inheritance: - A producing sub class from a single super class is called single inheritance.

**A**

class A

class B extends A

**B**

**A**

class A

class B extends A

class C extends A

class D extends A

**D**

**B**

**C**

2) Multiple inheritances: - Producing sub class from multiple super classes is called multiple inheritances. Multiple means more than one.

Multiple inheritances are not supported in java.

EX: -

**B**

**A**

class A

class B

class C extends A,B

**C**

// invalid

Statements

class A in java

**A**

**C**

**B**

class B

class C

class D extends A,B,C

class E extends A,B,C

**D**

**E**

🡪 Why multiple inheritances are not supported in java?

1. 1) Multiple inheritance leads to confusion for a programmer. Operator over loading, multiple inheritances are not available in java.

**A**

**x**

**B**

**x**

**C**

**x**

2) If the programmer wants we can achieve multiple inheritance using interfaces.

class D extends A,B,C // invalid

class D implements A,B,C // valid

Here A,B,C are not classes only interfaces. So it is not multiple inheritances. This is way of multiple inheritances indirectly.

3) A programmer can achieve multiple inheritances by using single inheritance repeatedly.

Because the above reasons the direct multiple inheritances is not available in java.

🡪// multiple inheritances Using interfaces

interface Father

{

int prop1=500000;

float ht1=6.2f;

}

interface Mother

{

int prop2=400000;

float ht2=5.2f;

}

class Child implements Father,Mother

{

void property( )

{

System.out.println("child property="+(prop1+prop2));

}

void height( )

{

System.out.println("child height="+(ht1+ht2)/2);

}

}

class Multi

{

public static void main(String args[])

{

Child ch=new Child( );

ch.property( );

ch.height( );

}

}

D:\psr\Core java>javac Multi.java

D:\psr\Core java>java Multi

child property=900000

child height=5.7

Multi Hierarchical Hybrid

Inheritance Inheritance Inheritance

Packages: - A package represents a sub directory that contains a group of elements.

EX: - import.java.io.\*;

Compile: - javac –d . class name.java

Advantages of packages: -

1. Packages hide classes & interfaces. Thus they provide protection for them.
2. The classes of one Package are isolated from the classes of another Package. So it is possible to use same names for the classes into different packages.
3. Using package concept we can create our own Packages & also we can extend already available Packages.
4. Packages provide re usability of code.

🡪 // creating our own Package: pack

package pack;

public class Addition

{

private double d1,d2;

public Addition(double a,double b)

{

d1=a;

d2=b;

}

public void sum( )

{

System.out.println("Sum="+(d1+d2));

}

}

D:\psr\Core java>javac -d . Addition.java

🡪 // using package: pack (Addition)

class AdditionUse

{

public static void main(String args[])

{

pack.Addition obj=new pack.Addition(10,15.5);

obj.sum( );

}

}

D:\psr\Core java>javac -d . AdditionUse.java

D:\psr\Core java>java AdditionUse

Sum=25.5

🡪 // using package: pack

package pack;

public class Subtraction

{

public static double sub(double a,double b)

{

return a-b;

}

}

D:\psr\Core java>javac -d . Subtraction.java

🡪 // using package: pack (Subtraction)

class SubtractionUse

{

public static void main(String args[])

{

pack.Addition obj=new pack.Addition(13,43.5);

obj.sum( );

double res=pack.Subtraction.sub(13,43.5);

System.out.println("result = "+res);

}

}

D:\psr\Core java>javac -d . SubtractionUse.java

D:\psr\Core java>java SubtractionUse

Sum=56.5

result = -30.5

(or)

import pack.Addition; import pack.\*

import pack.Subtraction;

class SubtractionUse1

{

public static void main(String args[])

{

Addition obj=new Addition(13,43.5);

obj.sum( );

double res=Subtraction.sub(13,43.5);

System.out.println("result = "+res);

}

}

D:\psr\Core java>set CLASSPATH=pskr;.;%CLASSPATH%

D:\psr\Core java>javac -d . SubtractionUse1.java

D:\psr\Core java>java SubtractionUse1

Sum=56.5

result = -30.5

Import the package name eliminates the before adding class names.

Error: - bad class file : .\Addition.java

We can handle these errors must follow some steps

Step 1: - Java compile searches from the package in .jar(java archair) file format in the following directory

C:\program files\java\jre1.5.0\lib\ext

🡪 What is java archair file?

A) Jar file is a compressed version of several. Class or. Java files. A jar file created using jar utility provided by sun micro systems.

Step 2:- Java compiler tapes the present directory as the package and searches for .class files there.

Step 3:- Java compiler searches for the package in current directory.

Compile:- pack directory cut & paste in to pskr folder

D:\ psr> cd pskr.

D:\ psr\ pskr>javac SubtractionUse.java

`

Step 4:- Finally java compiler searches for the package in the class path.

Class path: -

Class path is an operating system variable. That stores active directory path.

🡪 To sea the class path

D:\ psr> echo %classpath%

Display content of class path

.\;C:\Program Files\Java\jdk1.5.0\_07\jre\lib\rt.jar;d:\psr;pskr;

🡪 To set the class path to the package directory.

D:\psr> set classpath=d:\psr\pskr;.;%classpath%

Compile the above program & run it.

🡪 // adding interface to the package: pack

package pack;

public interface MyDate // just display date & time

{

void showDate( ); // public abstract

}

D:\psr\Core java>javac -d . MyDate.java

🡪 // this is implementation class of MyDate

package pack;

import pack.MyDate;

import java.util.Date;

public class MyDateImpl implements MyDate

{

public void showDate( )

{

Date d=new Date( );

System.out.println(d);

}

}

D:\psr\Core java>javac -d . MyDateImpl.java

🡪 // using the implementation class object

import pack.MyDateImpl;

class MyDateImplUse

{

public static void main(String[] args)

{

MyDateImpl obj=new MyDateImpl( );

obj.showDate( );

}

}

D:\psr\Core java>javac -d . MyDateImplUse.java

D:\psr\Core java>java MyDateImplUse

Mon Sep 03 13:23:55 IST 2007

🡪 // creating sub package

package inet;

public class Sample

{

public void show( )

{

System.out.println("Hai how are u");

}

}

D:\psr\Core java>javac –d . Sample.java

🡪 // using the sub package

import inet.Sample;

public class SampleUse

{

public static void main(String args[])

{

Sample s=new Sample( );

s.show( );

}

}

D:\psr\Core java>javac SampleUse.java

D:\psr\Core java>java SampleUse

Hai how are u

Suppose inet directory cut & paste the directory in the drive pskr directory and set the path

set classpath d:\pskr;.;%classpath%

Access specifiers (or) access modifiers: - An access specifier is a key word that specifies how to read or access the members of a class or the class itself. There are 4 types of access specifiers. They are

1) private 2) public 3) protected 4) default

1) private members of a class are not accessible in other classes of same package or another package.

2) public members of a class are available to other classes of same package or another package. Its scope is global scope.

3) protected members are accessible to the other classes of the same package but not in another package.

B A C

**private**

**public**

**protected**

**default**

Same directory another

4) default members are accessible in other classes of the same package. They are not accessible in another package. Its scope is package scope.

Note: - protected members of a class are always available to its sub classes in the same package or another package also. The scope of protected of some times global scope & some time package scope.

🡪 // access specifiers

package same;

public class A

{

private int a=1;

public int b=2;

protected int c=3;

int d=4;

}

D:\psr\Core java>javac -d . A.java

🡪 // access specifier with same package

package same;

import same.A;

public class B

{

public static void main(String args[])

{

A obj=new A( );

System.out.println(obj.a);

System.out.println(obj.b);

System.out.println(obj.c);

System.out.println(obj.d);

}

}

D:\psr\Core java>javac -d . B.java

B.java:9: a has private access in same.A

System.out.println(obj.a);

^

1 error

🡪 // access specifier with another package

package another;

import same.A;

public class C

{

public static void main(String args[])

{

A obj=new A( );

System.out.println(obj.a);

System.out.println(obj.b);

System.out.println(obj.c);

System.out.println(obj.d);

}

}

D:\psr\Core java>javac -d . C.java

C.java:9: a has private access in same.A

System.out.println(obj.a);

^

C.java:11: c has protected access in same.A

System.out.println(obj.c);

^

C.java:12: d is not public in same.A; cannot be accessed from outside package

System.out.println(obj.d);

^

3 errors

🡪 // access specifier with another package

package another;

import same.A;

public class C extends A

{

public static void main(String args[])

{

C obj=new C( );

System.out.println(obj.a);

System.out.println(obj.b);

System.out.println(obj.c);

System.out.println(obj.d);

}

}

D:\psr\Core java>javac -d . C.java

C.java:9: a has private access in same.A

System.out.println(obj.a);

^

C.java:12: d is not public in same.A; cannot be accessed from outside package

System.out.println(obj.d);

^

2 errors

API document: - Application programming interface.

Create API document: - D:\psr>javadoc \*.java (press enter)

Open internet explorer & select one choose (dos) file & press enter.

Exception handling: - An exception is a runtime error.

An error is software is called a bug. Removing errors from software is called debugging.

1. Compile time errors: - These are errors in the syntax or the grammar of the language. These errors are detected by the compiler at run time of compilation. (Java compiler can display up to 100 errors). Desk checking is the solution of compile time errors.
2. Run time errors: - These errors are the errors which occur because in sufficiently of compiler system. These errors are detected by JVM at the time of running the program. Run time errors are serious errors. Only the programmers are responsible for it.
3. Logical errors: - These errors represent bad logic (or) these errors are not detected by the compiler or JVM. Logical errors are detected by comparing the program out put with manually calculated result.

🡪 What is checked exception?

A) The exception will occur during compilation & are executed by the java compiler are called checked exception. The exception which are detected at runtime by the JVM are called unchecked exceptions or run time errors.

Exception: - Any abnormal event in the program is called an exception. All exception are represented by classes in java

🡪 // Exception example

class ExceptionDemo

{

public static void main(String args[])

{

System.out.println("open files");

int n=args.length;

System.out.println("n="+n);

int a=45/n;

System.out.println("a="+a);

System.out.println("close files");

}

}

D:\psr\Core java>javac ExceptionDemo.java

D:\psr\Core java>java ExceptionDemo Sunil Kumar Reddy

open files

n=3

a=15

close files

🡪 All exceptions are subclasses of which class?

A) Exception class.

🡪 What is throwable?

A) Throwable is a class that represents is errors & exceptions in java

🡪 What is Exception?

1. Any abnormal event in the program is called an exception.

🡪 What is the difference between an error & exception?: -

An exception is an error that can be handled. An error can not be handled

When there is an exception JVM displays exception name & its description & then abnormally terminates the program. Because of these the files are data bases are not closed & hence user’s entire data will be lost. This is effect of exception.

java.lang

throwable

errors exceptions

all errors all exception

🡪 In case of an exception the programmer to do the following tasks.

1) The programmer should write all the statements, where there an exception, inside a try blocks.

try {

Statements;

}

When there is an exception in try block JVM will not terminate the program. It will store exception details, in an exception stack & then jumps into catch block.

try

{

Statements;

**Exception**

**details**

}

Catch(ExceptionClass obj)

{

Statements; Exception block

}

2) In catch block the programmer has to display exception details & any other messages to the user.

3) The programmer should close the files & data bases In finally block.

Note: - finally blocks is always executed whether there is exception or not.

finally {

statements;

}

🡪 // Exception example

class ExceptionExample

{

public static void main(String args[])

{

try

{

System.out.println("open files");

int n=args.length;

System.out.println("n="+n);

int a=45/n;

System.out.println("a="+a);

int b[]={10,19,12,13};

b[50]=100;

}

catch(ArithmeticException ae)

{

System.out.println("ae");

System.out.println("plz type data while running the data");

}

catch(ArrayIndexOutOfBoundsException aie)

{

System.out.println("aie");

System.out.println("plese see that array index is within the range");

}

finally

{

System.out.println("close files");

}

}

}

D:\psr\Core java>javac ExceptionExample.java

D:\psr\Core java>java ExceptionExample

open files

n=0

ae

plz type data while running the data

close files

D:\psr\Core java>java ExceptionExample Swarna Latha Reddy

open files

n=3

a=15

aie

plese see that array index is within the range

close files

🡪 Even though multi exceptions are found in the program, only one exception is raised at a time.

🡪 We can handle multiple exceptions by writing multiple catch blocks.

🡪 A single try block can be followed by several catch blocks.

🡪 Catch block does not always exit without a try, but a try block exit without a catch block.

🡪 Finally block is always executed whether there is exception or not.

🡪 // not handling the exception

import java.io.\*;

class Sample

{

void accept( )throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("enter ur name : ");

String name=br.readLine( );

System.out.println("Hai "+name);

}

}

class ExceptionNotHandle

{

public static void main(String args[])throws IOException

{

Sample s=new Sample( );

s.accept( );

}

}

D:\psr\Core java>javac ExceptionNotHandle.java

D:\psr\Core java>java ExceptionNotHandle

enter ur name :

Rangha Reddy

Hai Rangha Reddy

throws: - throws statement is useful to throw out an exception without handling it.

Note: - if the programmer does not handle any type of exception we should write throws Exception.

throw:-it is useful to create an exception object and throw it out of try block

🡪 // Throw Example

class ThrowDemo

{

static void Demo( )

{

try

{

System.out.println("inside method");

throw new NullPointerException("my data");

}

catch(NullPointerException ne)

{

System.out.println("ne");

}

}

public static void main(String args[])

{

ThrowDemo.Demo( );

}

}

D:\psr\Core java>javac ThrowDemo.java

D:\psr\Core java>java ThrowDemo

inside method

ne

🡪 What is the difference between throws & throw?

A) throws is used to throw out of an exception without handling it.

throw is used to create an exception & throwing it & handle it.

Uses of throw: - 1) throw is used in software testing to test whether a program is handling all the exceptions have climbed by programmer.

2) throw is used to create & throw user defined exceptions

Types of exceptions: -

1) Built-in exceptions: - These are the exceptions which are already available in java.

EX: -

a) ArithmeticException

b) ArrayIndexOutOfBoundsException

c) StringIndexOutOfBoundsException

d) NullPointerException

e) NoSuchMethodException

f) ClassNotFoundException

g) FileNotFoundException

h) NumberFormatException

i) RuntimeException

j) InterruptedException

2) User-defined exceptions: - These are the exceptions created by user of the language.

Creating user defined exceptions: -

1. Write user exception class as a sub class to exception class

Ex: - class MyException extends Exception

1. Write a default constructor in user exception class

Ex – MyException( ) { }

1. Write a string parameterized constructor & from their call the super class constructor.

Ex: - MyException(String str)

{

Super(str);

}

1. To raise the exception creates user exception class object & throw it using throw statement.

Ex: - throw me;

🡪 // user defined exception

class MyException extends Exception

{

private static int accno[]={12,13,11,14,15};

private static String name[]={"sun","Sud","sree","siri","latha"};

private static double bal[]={345.53,5345.45,534.534,5435.56,324.643};

MyException( )

{

}

MyException(String str)

{

super(str);

}

public static void main(String args[])

{

try

{

System.out.println("accno "+"\t"+"name"+ "balance ");

for(int i=0;i<5;i++)

{

System.out.println(accno[i]+"\t"+name[i]+"\t"+bal[i]);

if(bal[i]<340.00)

{

MyException me=new MyException("balance ammount is less");

throw me;

}

}

}

catch(MyException me)

{

me.printStackTrace( );

}

}

}

D:\psr\Core java>javac MyException.java

D:\psr\Core java>java MyException

accno namebalance

12 sun 345.53

13 Sud 5345.45

11 sree 534.534

14 siri 5435.56

15 latha 324.643

MyException: balance ammount is less

at MyException.main(MyException.java:24)

🡪 All application on internet send or receive data is the form of objects.

2) The class in java.util package acts up on object only. Because of the above reason it is necessary to convert primitive data types into object form. This is done by wrapper classes.

Wrapper class:- All wrapper classes are defined in java.long package.

A wrapper class wraps contains a primitive data type in its object.

Character is a wrapper class in to a char data type.

Primitive data type Wrapper class:-

char Character

byte Byte

int Integer

float Float

double Double

long Long.

1) Character class:-

The character class wraps a value of the primitive type ‘char’ an object. An object of type character contains a single field whose type is char.

Constructors:-

🡪 1) Character (char ch)

Ex: - char ch = ‘A’

Character obj=new Character (ch);

Methods:-

🡪 1) char charValue ( )

Returns the char value of the invoking object.

Ex: - char x =obj.charValue( );

🡪 2) static boolean isDigit(char ch)

Returns true if ch is a digit (0 to 9) otherwise returns false.

Ex:- char = ‘9’; boolean x = character.isDigit(ch);

🡪 3) static boolean isLetter(char ch).

returns true if ch is a letter (A to Z or a to z)

🡪 4) static boolean isupperCase(char ch).

returns true if ch is an upper case letter (A to Z).

🡪5) static boolean isLowerCase(char ch)

returns true if ch is an Lower case letter (a to z).

🡪 6) static boolean isSpaceChar(char ch).

returns true if ch is coming from Space Bar.

🡪 7) static boolean iswhitespace(char ch )

returns true if ch is coming from Tab, Enter, Backspace.

🡪 8) static char toUpperCase (char ch ).

converts ch into upper case.

🡪 9) static char toLowerCase (char ch)

Converts ch into lowercase.

🡪 Which of the wrapper classes does not a constructor with string parameter? (or) Which of the wrapper classes contains only one constructor?

A) Character

🡪 // test a character

import java.io.\*;

class CharacterTest

{

public static void main(String[] args) throws IOException

{

while(true)

{

// to accept data from key board

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("Enter a char");

char ch=(char)br.read( );

// test the ch

if(ch=='@')

System.exit(0);

else if(Character.isDigit(ch))

System.out.println("It is a digit");

else if(Character.isUpperCase(ch))

System.out.println("It is a capital letter");

else if(Character.isLowerCase(ch))

System.out.println("It a small Letter");

else if(Character.isSpaceChar(ch))

System.out.println("It is coming from space bar");

else if(Character.isWhitespace(ch))

System.out.println("It is a white space");

else

System.out.println("Sorry, i don't know that character");

}

}

}

D:\psr\Core java>javac CharacterTest.java

D:\psr\Core java>java CharacterTest

Enter a char

p

It a small Letter

Enter a char

4

It is a digit

Enter a char

~

Sorry, i don't know that character

Enter a char

@

2) Byte class: - The byte class wraps a value of primitive type ‘byte’ in an object. An object of type byte contains a single field whose type is byte.

Constructor: -

1. Byte(byte num)
2. Byte(String str)

Ex: - 1) byte b=99;

Byte obj=new Byte(b);

2) String str=”120”

Byte obj=new Byte(str);

Methods: -

🡪1) byte byteValue( )

Returns the value of invoking object as a byte.

🡪2) int compareTo(Byte b)

Compares the numerical value of invoking object with that of ‘b’. returns 0, if the values all equal. Returns a –ve value, if the invoking object has a lower value. Returns a +ve value, if the invoking object has a grater value.

Ex: -int n=b1.CompareTo(b2)

If b1==b2; n==0;

If b1<b2; n== -ve;

If b1>b2; n== +ve;

🡪3) static byte parseBytes(String str) throws NumberFormatException

Returns the byte equivalent ot the number contained in the string specified by ‘str’.

🡪4) String toString( )

Returns a string that contains the decimal equivalent of the invoking object.

🡪5) static ByteValueOf(String str) throws NumberFormatException

Returns a byte object that contains the value specified by string ‘str’.

🡪 // creating & comparing byte objects

import java.io.\*;

class Bytes

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("enter a byte no : ");

String str=br.readLine( );

Byte b1=new Byte(str);

System.out.print("enter another byte no : ");

str=br.readLine( );

Byte b2=Byte.valueOf(str);

int n=b1.compareTo(b2);

if(n==0)

System.out.println("both same ");

else

if(n>0)

System.out.println(b1+"is bigger than"+b2);

else

System.out.println(b1+"is lesser than"+b2);

}

}

D:\psr\Core java>javac Bytes.java

D:\psr\Core java>java Bytes

enter a byte no : 34

enter another byte no : 56

34is lesser than56

3) Integer Class: - The integer class wraps a value of the primitive type ‘int’ in a object. An object of type integer contains a single field whose type is int.

Constructors: -

1. Integer(int num)
2. Integer(String str)

Methods: -

🡪1) int intValue( )

Returns the value of invoking object as an ‘int’.

🡪2) int compareTo(Integer obj)

Compare the numerical value of the invoking object of ‘obj’. Returns 0,-ve or +ve value.

🡪3) static int parseInt(String str) throws NumberFormatException

Returns int equivalent of the string str.

Ex: - String str=”Hyderabad”;

Int n=Integer.parseInt(str);

🡪4) String toString( )

Returns a string from of the invoking object.

🡪5) static Integer valueOf(String str) throws NumberFormatException

Returns an integer object that contain the value shown by str.

🡪6) static String toBinaryString(int i)

Returns a string representation of the integer argument in base 2.

🡪7) static String toHexString(int i)

Returns a string representation of the integer argument in base 6.

🡪8) static String toOctalString(int i)

Returns a string representation of the integer argument in base 8.

🡪 // converting into other number system

import java.io.\*;

class ConvertIntegers

{

public static void main(String args[])throws IOException

{

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("Enter int :");

String str=br.readLine( );

int n=Integer.parseInt(str);

System.out.println("in Decimal ="+n);

str=Integer.toBinaryString(n);

System.out.println("In Binary ="+str);

str=Integer.toHexString(n);

System.out.println("In HexaDecimal ="+str);

str=Integer.toOctalString(n);

System.out.println("In Octal ="+str);

}

}

D:\psr\Core java>javac ConvertIntegers.java

D:\psr\Core java>java ConvertIntegers

Enter int :

123

in Decimal =123

In Binary =1111011

In HexaDecimal =7b

In Octal =173

4) Float class: - The float class wraps a value of primitive type ‘float’ in an object. An object of type float contains a single field whose type is float.

Constructers:-

🡪1)Float (float num)

🡪2)Float (String str)

Methods:-

🡪1) float floatValue( )

returns the value of the involving object as a float.

🡪2) double doubleValue( )

returns the value of the involving object as a double.

🡪3) int compareTo(Float f)

compares the numeric value of the involving object with that of ‘f’ returns .,-ve.+ve value.

🡪4) static float parseFloat (string str) throwsnumberFormatExecution.

Returns the float equivalent the string str .

🡪5) String toString ( )

returns the string equivalent of the involving object.

🡪6) static float valueOf (String str) throwsnumberFormatExecution

.Returns the float object with the value specified by string str.

5) double class: - The double class wraps a value of primitive type ‘double’ in an object. An object of type float contains a single field whose type is double.

Constructers:-🡪1) Double (double num)

🡪2) Double (String str)

Methods:-

🡪1) double floatValue( )

returns the value of the involving object as a double.

🡪2) double floatValue( )

returns the value of the involving object as a float.

🡪3) int compareTo(Double d)

compares the numeric value of the involving object with that of ‘d’ returns .,-ve.+ve value.

🡪4)static double parseDouble (string str)

throws NumberFormatExecution

Returns the double equivalent the string str .

🡪5) String toString ( )

returns the string equivalent of the involving object.

🡪6) static double valueOf (String str) throws NumberFormatExecution

Returns the double object with the value specified by string str.

6) Math class: - The class math contains methods for performing numerical operations.

Methods: -

🡪1) static double sin(double arg)

Returns the sine value of arg. Arg is an radians.

🡪2) static double cos(double arg)

Returns the cosine value of arg. Arg is an radians.

🡪3) static double tan(double arg)

Returns the tangent value of arg. Arg is an radians.

🡪4) static double log(double arg)

Returns the natural logarithm value of arg.

🡪5) static double pow(double x,double y)

Returns x to the power of y value.

🡪6) static double sqrt(double arg)

Returns the square root of arg.

🡪7) static double abs(double arg)

Returns the absolute value of arg.

🡪8) static double ceil(double arg)

Returns the smallest integers which is grater or equal to arg.

Ex:- Math.ceil(4.5) 🡪 is 5.0

Ceil means ceiling. It means up.

🡪9) static double floor(double arg)

Returns the grater integer which is lower or equal to arg.

Ex: - Math.floor(4.5) 🡪 is 4.0

🡪10) static double min(arg1,arg2)

Returns the minimum value of arg1 & arg2.

🡪11) static double max(arg1,arg2)

Returns the maximum value of arg1 & arg2.

🡪12) static long round(arg)

Returns the rounded value.

Ex: - Math,round(4.6) 🡪 is 5

🡪13) static double random( )

Returns the random numbers between 0&1.

🡪14) static double toRadians(double angle)

Converts angle in degrees in to radians.

🡪15) static double toDegrees(double angle)

Converts angle in radians in to degrees.

🡪 // random numbers between 1 & 10

class Rand

{

public static void main(String args[]) throws InterruptedException

{

while(true)

{

double d=10\*Math.random( );

int n=(int)d;

System.out.print(n+”\t”);

if(n==0)

System.exit(0);

Thread.sleep(3000);

}

}

}

D:\psr\Core java>javac Rand.java

D:\psr\Core java>java Rand

7 7 8 5 6 2 7 6 1 0

Java.utility: -

Stack: - A stack represents arrangement of elements (objects) in LIFO (last in first out) order.

Inserting the elements & deleting the elements will take place only from one side of the stack, called top of the stack. The other side is closed & it called bottom of the stack.

Inserting elements into stack is called push operation. Deleting element from top of the stack is called pop operation. Searching for an element in the element is called peep operation.

Ex: - A file of plates in a hotel, linking of coaches to a railway engine,

CD holder, bangles to hand.

How to create a stack: -

🡪1) To create a stack

Stack st=new Stack( );

🡪2) To know whether a stack is empty or not, use empty( )

bolean b=st.empty( );

🡪3) To push an element in to the stack

st.push(element);

🡪4) To pop an element from a stack

Object element=st.pop( );

🡪5) to search for an element in the stack

int i=st.search(element);

This method returns -1, if the element is not found.

**4**

**3**

**2**

**1**

Top

**5**

Bottom

🡪 // A stack with int values

import java.io.\*;

import java.util.\*;

class StackDemo

{

static void stpush(Stack st,int a)

{

st.push(new Integer(a));

}

static void stpop(Stack st)

{

Integer a=(Integer)st.pop( );

System.out.println("popped = "+a);

}

static int stSearch(Stack st,int a)

{

int i=st.search(new Integer(a));

return i;

}

public static void main(String args[])throws Exception

{

Stack st=new Stack( );

int element,pos,choice=0;

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

// menu

while(choice<4)

{

System.out.println("stack operations ");

System.out.println("1 push element ");

System.out.println("2 pop element ");

System.out.println("3 search for element ");

System.out.println("4 exit ");

System.out.print(" your choice ");

choice=Integer.parseInt(br.readLine( ));

// perform a task depending on choice

switch(choice)

{

case 1:

element=Integer.parseInt(br.readLine( ));

stpush(st,element);

break;

case 2:

stpop(st);

break;

case 3:

System.out.print("enter element :");

element=Integer.parseInt(br.readLine( ));

pos = stSearch(st,element);

if(pos==-1)

System.out.println("element not found ");

else

System.out.println("element found at position :"+pos);

break;

default:

return;

}

System.out.println("stack ="+st);

}

}

}

D:\psr\Core java>javac StackDemo.java

D:\psr\Core java>java StackDemo

stack operations

1 push element

2 pop element

3 search for element

4 exit

your choice 1

2

stack =[2]

stack operations

1 push element

2 pop element

3 search for element

4 exit

your choice 1

3

stack =[2, 3]

stack operations

1 push element

2 pop element

3 search for element

4 exit

your choice 1

4

stack =[2, 3, 4]

stack operations

1 push element

2 pop element

3 search for element

4 exit

your choice 2

popped = 4

stack =[2, 3]

stack operations

1 push element

2 pop element

3 search for element

4 exit

your choice 3

enter element :2

element found at position :2

stack =[2, 3]

stack operations

1 push element

2 pop element

3 search for element

4 exit

your choice 4

Linked list: -It represents a set of nodes such that each node contains two fields.

node

**data**

**link**

1. The data field stores data.
2. The link field stores reference of the next node

Crate a linked list: -

🡪1) To create a linked list:

LinkedList ll=newLinkedList( );

🡪2) To add elements to a linked list

ll.add(element);

To add element in second position

ll.add(2,element);

🡪3) To remove first element from the linked list

ll.remove( );

To remove last element from the linked list.

ll.removeLast( );

To remove second element

ll.remove(2);

🡪4) To change the second element with a new element

ll.set(2,New element);

🡪 to retrieve element by element from collection objects. We can use the following one of the following interfaces

1. Iterator 2) ListIterator 3) Enumeration

🡪 // a linked list with string

import java.io.\*;

import java.util.\*;

class LLDemo

{

public static void main(String args[])throws IOException

{

// create an empty linked list

LinkedList ll=new LinkedList( );

// add elements to ll

ll.add("America");

ll.add("India");

ll.add("China");

ll.add("Japan");

ll.add("France");

ll.add("Sri lanka");

// display the linked list

System.out.println("Linked List = "+ll);

// vars

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

String element;

int pos,choice=0;

// menu

while(choice<4)

{

System.out.println("Linked List operations ");

System.out.println("1 . insert element : ");

System.out.println("2 . remove element : ");

System.out.println("3 . change element : ");

System.out.println("4 . exit ");

System.out.println(" your choice :");

choice=Integer.parseInt(br.readLine( ));

// depending on choice, perform a task

switch(choice)

{

case 1:

System.out.print("Enter element : ");

element=br.readLine( );

System.out.print("Enter Position No : ");

pos=Integer.parseInt(br.readLine( ));

ll.add(pos,element);

break;

case 2:

System.out.print("Enter Position No : ");

pos=Integer.parseInt(br.readLine( ));

ll.remove(pos);

break;

case 3:

System.out.print("Enter element : ");

element=br.readLine( );

System.out.print("Enter Position No : ");

pos=Integer.parseInt(br.readLine( ));

ll.set(pos,element);

break;

default :

return;

}

System.out.println("List ="+ll);

}

}

}

D:\psr\Core java>javac LLDemo.java

D:\psr\Core java>java LLDemo

Linked List = [America, India, china, Japan, France, Sri lanka]

Linked List operations

1 . insert element :

2 . remove element :

3 . change element :

4 . exit

your choice :

1

Enter element : 2

Enter Position No : 4

List =[America, India, china, Japan, 2, France, Sri lanka]

Linked List operations

1 . insert element :

2 . remove element :

3 . change element :

4 . exit

your choice :

2

Enter Position No : 3

List =[America, India, china, 2, France, Sri lanka]

Linked List operations

1 . insert element :

2 . remove element :

3 . change element :

4 . exit

your choice :

2

Enter Position No : 5

List =[America, India, china, 2, France]

Linked List operations

1 . insert element :

2 . remove element :

3 . change element :

4 . exit

your choice :

3

Enter element : 2

Enter Position No : 1

List =[America, 2, china, 2, France]

Linked List operations

1 . insert element :

2 . remove element :

3 . change element :

4 . exit

your choice :

4

Arrays: - Arrays class contains methods to handle any array.

🡪1) To sort the elements of an array in to ascending order.

Arrays.sort(arr);

🡪2) To sort only a range of elements of the array

Arrays.sort(arr,start,end);

These sort elements of ‘arr’ starting from ‘start’ till ‘end’ -1 element.

🡪3) To search for an element in an array

Arrays.binarySearch(arr,element);

This searches for ‘element’ in the array ‘arr’ & returns its position. If the ‘elements’ is not found, it reyurns a –ve value.

Note: - Binary search( ) will not act only a sorted array.

🡪4) To compare two arrays, to know if they are same or not.

Arrays.equals(arr1,arr2);

This returns true, if arr1 & arr2are same, else it returns false.

🡪 // sorting & searching in an array

import java.io.\*;

import java.util.\*;

class ArraysDemo

{

public static void main(String args[])throws IOException

{

// to accept data from key board

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.print("How many elements you want : ");

int n=Integer.parseInt(br.readLine( ));

// create int type array

int arr[]=new int[n];

// store elements into arr

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

{

System.out.print("Enter element : ");

arr[i]=Integer.parseInt(br.readLine( ));

}

// display the elements

System.out.print("Your Array is : "+”\t”);

display(arr);

// sort the elements in to ascending order

Arrays.sort(arr);

// display the sorted elements

System.out.print("Sorted array is : "+”\t”);

display(arr);

// searches for an element in an array

System.out.println("Enter element to Search : ");

int x=Integer.parseInt(br.readLine( ));

int pos=Arrays.binarySearch(arr,x);

if(pos<0)

System.out.println("Element not Found");

else

System.out.println("it position is :"+(pos+1));

}

static void display(int arr[])

{

for(int i:arr)

System.out.println(i);

}

}

D:\psr\Core java>javac ArraysDemo.java

D:\psr\Core java>java ArraysDemo

How many elements you want :

4

Enter element : 7

Enter element : 9

Enter element : 3

Enter element : 6

Your Array is : 7 9 3 6

Sorted array is : 3 6 7 9

Enter element to Search :

9

it position is :4

Cloning: - Cloning is a technology to obtain exact copy of a plant, a bird an animal and a human being.

Cloning in java: -

Obtaining bitwise exact copy of an object is called cloning.

1. Shallow cloning: - In this, any modifications to the cloned object, will also modify the original object.
2. Deep cloning: - in this cloning any modifications to the cloned object will not affect the original object.

EX: -

🡪 // cloning example

class Employee implements Cloneable

{

int id;

String name;

Employee(int i,String s)

{

id=i;

name=s;

}

void display( )

{

System.out.println("Id = "+id);

System.out.println("Name = "+name);

}

Employee myClone( )throws CloneNotSupportedException

// (or) protected Object Clone( ) throws

CloneNotSupportedException

{

return(Employee)super.clone( );

// (or) return super.clone( );

}

}

class CloneDemo

{

public static void main(String args[]) throws CloneNotSupportedException

{

Employee e1=new Employee(10,"sudheer");

e1.display( );

Employee e2=e1.myClone( );

// (or) Employee e2=(Employee)e1.Clone( );

e2.display( );

}

}

D:\psr\Core java>javac CloneDemo.java

D:\psr\Core java>java CloneDemo

Id = 10

Name = sudheer

Id = 10

Name = sudheer

🡪 Cloneable method does not contain any interface. A tagging interface or a marking interface is an interface without any methods or with zero abstract methods

Interface indicates a special purpose for the object of the class.

Ex: - Cloneable, Serialization.

🡪 Getting back object from the file is called Serialization. Storing a object in to a file is called a Serialization.

