**Java**:

IT : non IT:

Computer: system: binary: 0,1::::::::0000001111110000000: Hello All: 00000011111

COBOL, PASCAL, FORTON, FOXPRO::::

Categories: Special purpose & General:

Low Level programming : basically in 0 ‘s and 1’s: : -----

Middle level /Assembly: ------- 00001111: English: Store, Register…load…

**High level language**: : these languages: code is exactly to English:

PASCAL:

C Programming: Structural languages: /Function-Data Languages:

Data growing: main focus functions: data:

**Function** were active holder: **data** passive: add(10,20);

**void add(int a, int b){**

**Int c=a+b;**

**Printf(c);**

**}**

**1990: OOPL: C++, Java:**

**Object:**

Function: data:

Data:

How we can make our Data more secure:

Technology: object: object is a basic building block of ur system:

You have to construct an application for Library management system: function:

Real world object: librarian: books, students, professor: distributors:

Convert all these real world objects in to the soft objects;

class Librarian{

id, name, salary, age…..

Procuring book, issuingbooks, return(), calculatingfine()

}

Librarian: id, name, salary, age….

Procuring book, issuingbooks, return(), calculatingfine()

Librarian :

Mechanical: Programming:

Low: 0000111: computer system: No translator:

Middle : LOAD, STORE, : **Assembler**: machine level language: CS:

High: :::: English Like: Compiler and Interpreters: : machine language: compiler or it can have interpreter: either or:

Any programing:

class Employee{

}

Execute: compiler / Interpreters : compile: I ll convert/ translate that code from source code to Machine code:

{

State1; machine code

State2; machine code

Stat10;

}

At a time to machine code: if there is no error: mistake: highlight all the mistakes: 3 displayed:

Translators : Indian: Hindi, Marathi: English ::::: English : translator: Japanese: Japan:

Features of oo: function data languages:

1. **Inheritance**: reusability of the code: Process in which a class/object can acquiring the properties of another class:

**Inheriting** : hair color, skin color, face str, behavior

**Shape**: area: perimeter:

Rectangle: Circle.. square

1. Polymorphism: : area(){ area=pi\*r\*r; }// circle
   * 1. area({area=side\*side; }// squre:
     2. + ::: 10 + 10=20: Addition of 2 nos
     3. +: Simpli + Learn: Concatenation: SimpliLearn
2. Data Abstraction: ATM: With, ChangePIN, MiniStatements… logs , maintain ATM: ADMIN:

Order meal: 15 - 20: meal: enjoy: taste : abstraction: getMoney(){ }

Data hiding:

1. Encapsulation: wrapping up of the data: capsules: ingr: insulation:

Translator: machine:

What is java?: OOPL: pure OOPL: not 100% OOPL: **Primitive DT**: Do not have direct Multiple inheritance:

High-level oopl, **robust**: OOPL: chance of error is very less: Abc.java: compile: javac Abc.java: : Abc.class

Both : Compiler and interpreter: byte code: interpret that code : output:

History of Java:1991: ;;;; 1995 James Gostling known as a father of java: Green team: Sun microsystem: green team: green talk: OAK : JAVA:

Features of Java:

1. Simple:
2. OO: pure
3. Portable: compile: JVM: .class:
4. Secured:
5. Robust:
6. Architectural Neutral: soft/hw:
7. Interpreted:
8. Platform independent: hw sw
9. Multi-threading: T1:
10. Dynamic:
11. Distributed: client- server:

**JVM architecture: queries: JVM:**

**JVM: java virtual machine: heart and brain of java: portable: WORA: write once and run anywhere:**

**Virtual machine :**

**P3… p4::::: windows98…. xp: execute on linux: 128 mb 16 gb: physical: 2 dual OS:**

**Windows: Virtaul Box: Linux: illusion linux: windows: VM: .exe…. .txt. doc::: linux:**

**7-8 years: desktop: android newly : games.. application APK: windows xp: : bluestack : illusion of Android OS; on our computers: whats app… games:**

**VM:**

**JVM: .class: converting in to machine code:**

**C, C++ : compiled: transalated in to platform specific machine code: compiled languges:**

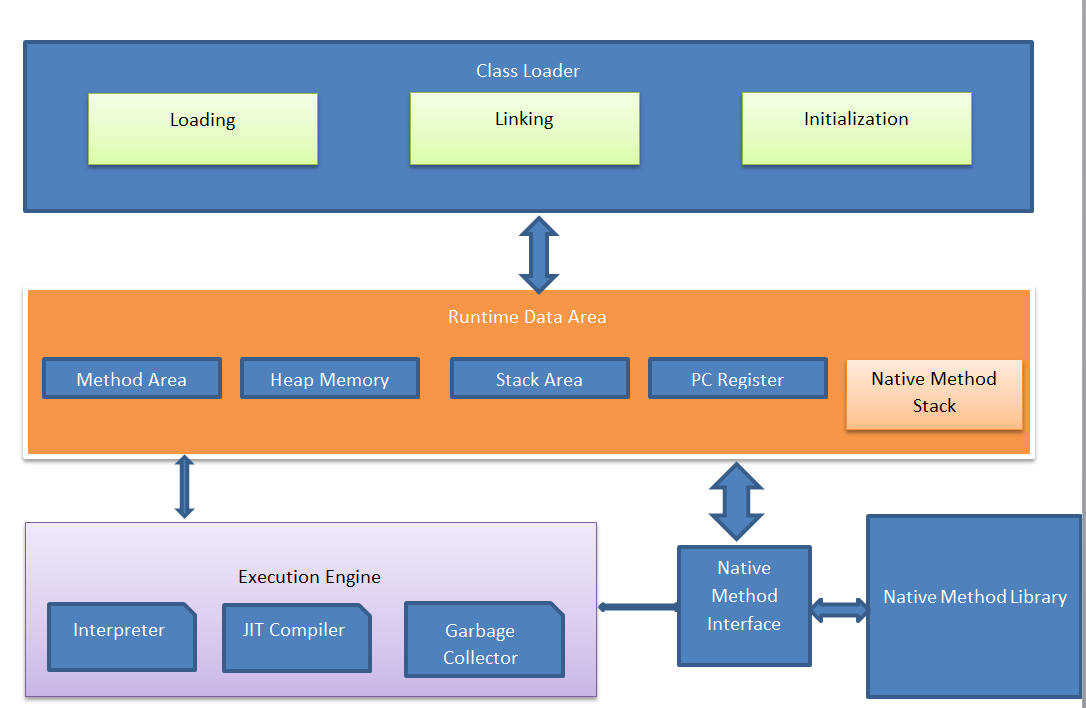
**Javascript, python…. Interpreter:**

**Java: compiler and interpreted**

**Abc.java javac : compiled;;;; class file/byte code: .class**

**JVM: interpreter the respective code: underlying platform: byte code in to the machine code:**

**JVM architecture:**

****

**Class Loader:**

1. **Load class:** 
   1. **Loading :**
      1. **Bootstrap class loader: folder: /packages in java: java.lang, java.net, java.io…**
      2. **Extension class loader: extension for standard java libraries:**
      3. **Application class loader:**
   2. **Linking:**
      1. **Verification: .class : VerifyException may occur: 19: compiled: 11:**
      2. **Preparation: JVM allocates memory for static fields of class and interfaces: and initial values are set by JVM:**
      3. **Resolution : runtime constant pool**

**Employee{**

**Int a, int b, int c;**

**Void getData(){**

**}**

**}**

**Employee e1=new Employee();**

**Employee e2=new Employee();**

**e1.getData()**

* 1. **Initialization : constructors: : in the constructor or static block:**

**Runtime Data Area:**

1. **Method Area: all the class level data: runtime constant pool, fields, methods area, code for method , constructor…. OutofMemoryException**
2. **Heap Memory: all the objects and their corresponding instance variables:**
3. **Stack area: Thread: OS: t1: t2: t3:……. R1: R2: R3: : local varibles, method calls, and partially calculated results** 
   1. **Local variable**
   2. **Operand stack**
   3. **Frame data:**
4. **PC: Program counter register: to hold the**

**Address: of instruction:**

**T1: {**

**1;//addresss must be stored some where**

**2;**

**3**

**4**

**}**

**Native method stack: Native methods: which are not written in Java: C, C++:**

**Execution Engine:**

**Print some document : libraries: :**

Installation: JDK, Eclipse:

Hello world program without using any IDE:

Programming using IDE:

Variables in Java : int a=10

Int b=20;

Whose value can be changed: there values can be changed:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 |
| a | b | C | D | E | F | G | h |

Types of variables:

1 ) Local variables: a variables which is declared inside any method or block is known as local variable: there scope is limited to their block: they are not accessible outside the block

**2) Instance variables: a variable which is accessible using instance/object of the class of the class:**

**3) static variables: We need to have a single copy of the variables:**

**Class is a collection of similar kinds of objects:**

**Students: pankaj, Swapnil…… sid, name, age, opted for some course: object or instance:**

**Student swapnil;**

**Student pankaj;**

FirstSL f1=new FirstSL();

**Global variable: these variables are not declared inside any method nor inside any block**

**They are declared in a class:**

Data types in Java:

1. Primitive Data types:which are used to hold simple values:
2. boolean true **or** false : 1 bit
3. char ‘c’ 2 bytes
4. byte 0 1 byte
5. short 1 2 bytes
6. int 12009 4 bytes
7. float 10.20f 4byte
8. long 1980382038 8 bytes
9. double 10.220 8 byte
10. Non primitive Data types: Array, class, interface….

Operators in Java:

1. Unary operator: operaotrs which works on a single operand: a++
2. Binary operator: 2 operands: :::: a + b, a - b
3. Ternary operator: more than 2 operands: a>b? a is bigger : b is bigger
4. Arithmetic operators: +, -, \*, /,%
5. Logical operators: a > b, a<b , a>=b, a<=b, a!=b Relational operators
6. Assignment operators: a=10, a=a+10; a+=10; a-=10; a=a-10; a\*=10; a=a\*10….a%=10, a/=10;
7. Bit wise operators: bit wise and &, or: |

a>b

syso(“a is bigger”);

syso(“b is bigger”);

Control statements

If-else:

If(condition1)

{

}

else if(condition2){

}

else if(condition3){

}

Else if(condition n){}

Switch case:

Switch(expression ) {boolean expression: evaluated : generate some value: 1

Case value1:

statement1;

statements2;

break;

case value2:

statement 1:

statement2:

break;

case3:

.

.

.

.

.

.

Default :

}

<< >>: bit wise operator

Binary bit:

10 decimal :

Binary are always having their base 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 |  |  |  |  | 0 | 0 | 0 |

For: “Hello All”; multiple:

for(initialization; condition/termination condition;inc/decr ){

syso(“Hello All”);

}

Syso()

While:

While(condition){

//code;

//code;

++,--

}

Do while:

do{

state1;

state2;

..

}

while(condition);

String: String : Sequence of character: more than : String: class

Data entry : max data : String

Int n=str1.CompareTo(str2)//

If(n==0){

Syso(“pw is correct”);

}

Else if(n>0){

Syso(“the first String is larger than second”)

}

Else

{

Syso(“Second string is larger”)

}

Concat()

equals()//true

==;

Split(): str1.split(‘’); Hello: H E L L O

length(): 5

replace()

substring():

Name: nilesh

Last name: magar

Username: nilesraghahv

Pw:

Based on declaration : memory:

2 :

1. String str=”Hello All”;//String literal

String constant pool

1. String str=new String(“Hello All”);// new //instantiating string

Heap memory:

PAN card:

10lakh:

City: Pune:

City: pune: poone

String in Java in Immutable: Its contents cannot be changed : once it is declared :

String str=new String(“SimplyLearn”);

Str=str.concat(“Lets Learn”);

StringBuffer: mutable string: String:

Thread safe: at a time StringBuffer: only one thread: only one user can access this String:

Stringbuffer class is bit slow:

1. StringBuffer sb=new StringBuffer();// initial capacity of SB is 16
2. StringBuffer sb1=new StringBuffer(20);
3. StringBuffer sb2=new StringBuffer(“Hello All”);

Public synchronized Stringbuffer append(String s):

Public synchronized StringBuffer replace(int startindex,int ending index, String str)

StringBuilder: multiple thrads can access this StringBuilder:

1. StringBuilder sb=new StringBuilder ();// initial capacity of SB is 16
2. StringBuilder sb1=new StringBuilder (20);
3. StringBuilder sb2=new StringBuilder (“Hello All”);

Object and classes in Java:

Which have some common Properties:

Student: Akash, ROhit, Rimi, Rincy:

Class Librarian{

Int lid;

String Lname;

String desi;

Fields ;

Mehods;

Issue(){}

calculateFine(){}

procurebooks(){}

constructor:

}

Constructor:

Constructor is special kind of method in java: It has special purpose: to initialize instance variables:

Rule:

1. Name must be same as that of your class name
2. Need not to have return type:
3. Abstract , static, final, synchronized:

Types of Constructors In java:

1. No argument constructor:
2. Parameterized constructor:

Constructor Overloading

Method:

static:

this:

inheritance:

Scanner class: