**JAVA**

**1. What is Java?**

-> Simple programming language

-> Writing, compiling and debugging is easy

-> It can be reused

**2. Java features (or) importance of java (or) why we go for java?**

-> Platform independent (JAVA compiler converts source code into byte code, can be executed in any platform.

-> Open source (free of cost)

-> Secured (virtual firewall between application and the computer)

-> Multi-threaded ( more task can be completed at a time and Multitasking - different task can be completed at a time)

-> Portable (write ­once run anywhere) WORA

**3. Java architecture:**

**JDK:** (JAVA DEVELOPEMENT KIT)

-> To develop JAVA code, JDK has to be install

-> It has development kit

-> JDK version 1.0 to 1.15

-> Stand version is 1.8

**JRE:** (JAVA RUNTIME ENVIRONMENT)

- it helps to run the program

-> it contains predefined library files (.class files)

**JVM:** (JAVA VIRTUAL MACHINE)

-> it is used for memory allocation

-> it is not physically exists.

**JDK= Development kit + JRE**

**JRE= Predefined files+ JVM**

**4. Coding standards:**

There are two types of coding standards:

* Pascal
* Camel

**PASCAL:** Every words First Letter must be in Capital

Eg: Employee\_Details- (used in Project name and class name)

**CAMEL:** Except first word remaining words First letter must be Capital.

Eg: employeeDetails- (used in Object ,method and variable)

**5. Access Specifier:**

* **Public:** Global level access (inside and outside the package)
* **Private:** Class level access (inside the class)
* **Protected:** same like public but use **“extends”** keyword
* **Default:** Package level access.

**6. Garbage collection:**

* Un-referenced object are deleted automatically.
* Un-wanted or un-used memory are deleted automatically.
* It is possible only in java not in c, c++ etc.

**7. Datatypes:**

Two types:

* Primitive : - 1. A variable can store only one value

2. Predefined datatype.

3. There is no additional methods.

4. Store in memory reference.

* Non Primitive :- 1. A variable can store group of values.

2. Based on class

3. It is reference variable (or) object reference

4. Store based on reference.

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA TYPE** | **MEMORY SIZE** | **DEFAULT VALUE** | **WRAPPER CLASS** |
| Byte | 1 | 0 | Byte |
| short | 2 | 0 | Short |
| Int | 4 | 0 | Integer |
| Long | 8 | 0 | Long |
| Float | 4 | 0.0f | Float |
| double | 8 | 0 | Double |
| Char | 2 | - | Character |
| boolean | True/false | false | Boolean |

**8. Wrapper Class:**

* Convert datatype into class object
* It is used in collection.

**9. Range of datatype:**

-2^(n-1) to 2^(n-1)-1

Eg:

1. byte : 8 bit= 1byte

n=8

-2^(8-1) to 2^(8-1)-1

-128 to 127

2. int : 32 bit= 4 byte

n=32

-2^(32-1) to 2^(32-1)-1

**10. OOPS:**

Method of implementation in our program are organized in the form of method, object and class.

**Class:** Combination of object and method.

**Method:** Set of actions to be performed.

**Object: ->** Instance of the class.

->helps to call the method

-> it allocates memory

->new keyword for creation/allocating memory.

Eg:

Public class StudentDetails{

Public void stdName(){

System.out.println(“Arun”);

}

Public void stdId(){

System.out.println(“123”);

}

Public static void main(string[] args){

StudentDetails sd= new StudentDetails();

sd.stdName();

sd.stdId();

}

**11. Inheritance:**

* Relationship between parent and child class.
* One class access the property of another class using **“extends”** keywords.

Types of Inheritance:

* Single
* Multiple
* Multilevel
* Hierarchical
* Hybrid

**Single Inheritance:**

* Child class (B) access the property of parent class (A).

[A]

[B]

**Multiple Inheritance:**

* It has more than one parent class.
* So there will be priority issue to access the parent class.
* It is not possible in JAVA.
* can overcome this disadvantage by using **“Interface”.**

**[A] [B]**

**[C]**

**Multilevel Inheritance:**

**[A]**

**[B]**

**[C]**

**Hierarchical Inheritance:**

**[A]**

**[B] [C] [D]**

**Hybrid Inheritance:**

**[A]**

**[B] [C]**

**[D]**