12-05-2020

Applications of Java — Self driven automated electronic device's, internet programming.

Features Summarized as —

→ Java is a general purpose concrete object-oriented and class based programming language. Here we have a run-time environment for java known as JVM (Java Virtual Machine).

→ Java is highly popular and has dominated this field from early 2000 till present.

→ This programming language is definitely an ocean of opportunities.

→ Some technologies that make use of java as an essential core of their functionalities include the web development — then there is continuous testing for Android development.

→ JDK 1.0 — released in 1996. Many versions added new features such as :— JDK 1.1 (97), JDK 1.2 (98), JDK 1.3 (2000), JDK 1.4 (2002),

~~JDK 1.5 JDK 1.6 JDK 1.7 JDK 1.8 JDK 1.9 JDK 1.~~

J2SE 5.0 (2004), SE6 (2006), SE7 (2011), SE8 (2014's), SE9 (2017), SE10 (2018),

SE11 (2018), SE12 (2019), SE13 (2020), SE14 (Proposed in future but not released yet.)

(SE - Software Edition)

Java Environment

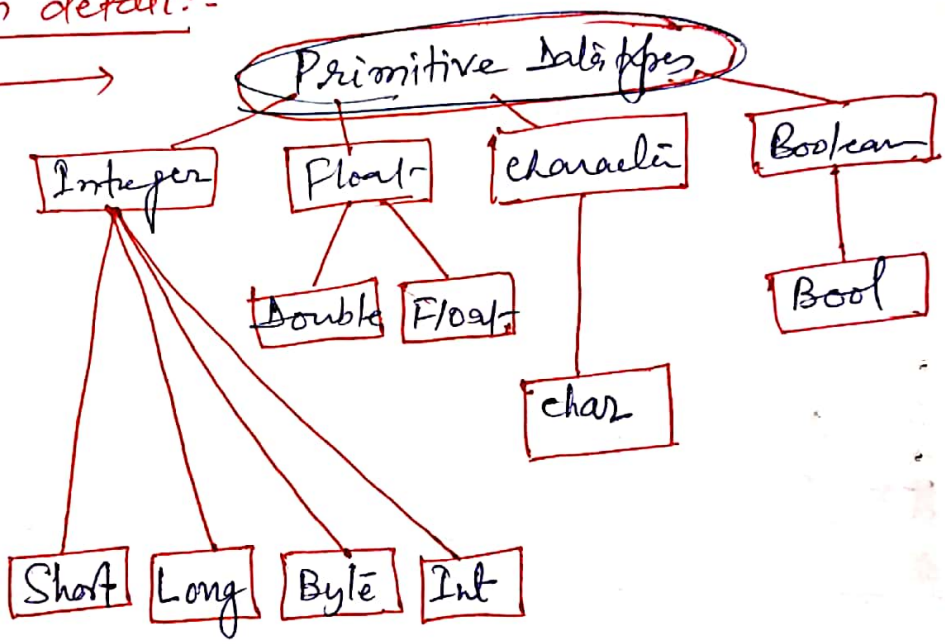
JIT — Just in time
JVM — Java Virtual Machine
JRE — Java Runtime Environment
JDK — Java Development Kit

Companies working on Java —

Google, Accenture, Infosys, Fipkart, Mantra, etc uses ~~their projects~~ java for various products, applications etc. Java might be facing tough competition right now but it can never be obsolete and its always the right time to begin your career ~~in~~ started in java programming language.

Basic concepts in detail:-

- Data types →
- Operators
- Functions
- Objects
- Class
- Encapsulation
- Polymorphism
- Abstraction
- Inheritance
- Loops



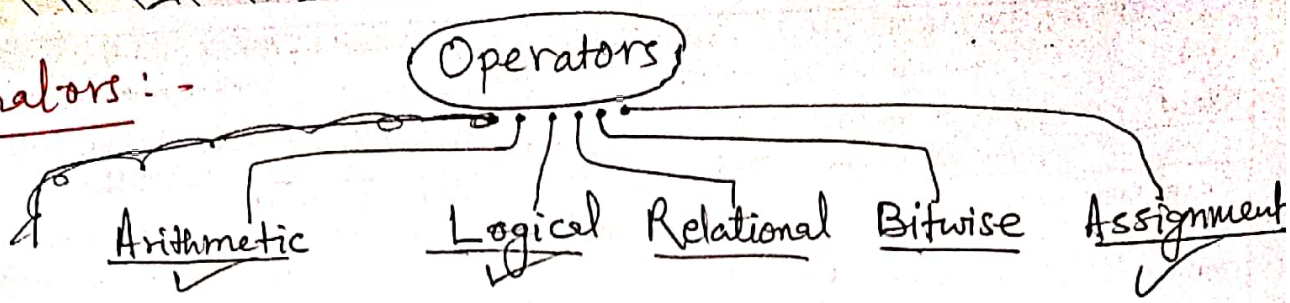

```
int myNum = 5;
float myfloat = 5.55f;
char myChar = "A";
boolean myBool = true;
System.out.println(Variable);
```

declaring types of variables.

```
long myLong = 10000L;
double myDouble = 10.55d;
~~double~~
float myfloat = 130.55f;
```

In above declarations, we have specified f, d, L at last, which indicates no extra numbers after decimal point will be get stored inside the variable.

→ Operators :-



Arithmetic - + - * / %

Assignment - = , += , -= , *= , /= , %= , <<= , >>= , &= , ^= , |= .

Logical - && || !

Relational - == != > < >= <=

Bitwise - & | ^ ~ (left shift) << (Right Shift) >>

(bitwise and) (bitwise or) (Bitwise XOR) Bitwise Complement

>>> (Zero Fill Right Shift)

→ Functions:- Now Functions are new in java. Now functional interface has to generate T and R

where T is type of the input argument
R is the return type of the function.

First we have to import java.util.function.Function

Example:-

```
import java.util.function.Function;
public class Demo{
    public static void main(String args[]){
Function<Integer, Double> half = a -> a/2.0;
System.out.println(half.apply(100));
    }
}
```

T R

name of the function half

output:-

50.0

Objects :-

Example:-

```
public class demo
{
    int x=50;
    public static void main(String args[])
    { demo obj = new demo();
    System.out.println(obj.x);
    }
}
```

In above program we have created object named- obj of demo class by -

```
demo obj=new demo();
```

This object obj is used to access the value of instance variable x as — obj.x

→ Encapsulation :-

Person.java

```
public class Person{
    private String name;
    public String getName(){ return name; }
    public void setName(String newName)
        {
            this.name = newName;
        }
}
```

Let us design another file as :— demo.java

```
public class demo
{ public static void main(String args[]){
Person p = new Person();
p.setName("Ram");
System.out.println(p.getName());
    } }
```

In above program :- Since we have declared name as private variable, hence in order to access it from outside the class we have to use a public method like — getName() and setName() as used in the above two programs. This is known as Encapsulation (Security of data members inside the class).

→ Polymorphism

The ability of an object to take many forms. Let us see with the example :-

```
class Animal {
    public void animalSound(){
      System.out.println("The animal makes a sound.");
      }
  }
class Pig extends Animal{
    public void animalSound(){
  System.out.println("The pig says : wee wee");
    }
  }
class Dog extends Animal{
   public void animalSound(){
   System.out.println("The dog says: bow wow");
}}
}
    class demo{
          public static void main(String args[])
          {
```

```
Animal myanimal = new Animal();
Animal mypig = new Pig();
Animal mydog = new Dog();

myanimal.animalSound();
mypig.animalSound();
mydog.animalSound();
} //close of main function.
} //close of demo class
```

Save this with **demo.java**

Compile & Run:- Output :-

```
The animal makes a sound.
The pig says: wee wee
The dog says: bow wow
```

This is how polymorphism works — because we have make one function — animalSound() but it works differently to the derived classes - and gives different outputs.

— END —