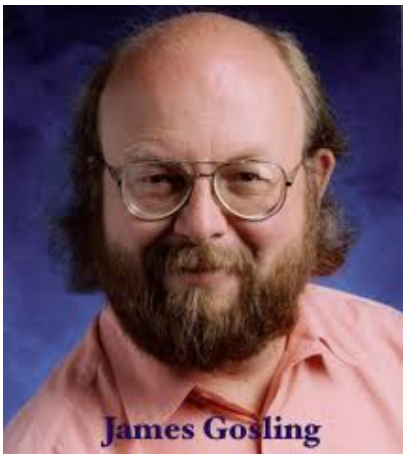
UNIT-1

Introduction

Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project(java) in the early '90s.



Java is the name of an island in Indonesia where the first coffee(named java coffee) was produced. this name was chosen by James Gosling while having coffee near his office.

Initially it was designed for small electronic appliances like set-top boxes.
Download java from https://www.oracle.com/java
It is an object-oriented language
example

- ★ Java,
- ★ C++,
- ★ C#,
- ★ Python,
- ★ R,
- ★ PHP,
- ★ JavaScript,
- ★ Ruby,
- ★ Perl,
- ★ Objective-C

- ★ Swift, Scala,
- ★ Kotlin,
- ★ Common Lisp,
- ★ MATLAB,
- ★ Smalltalk.

Syntax of java is borrowed from c & c++

Where java is used?

- Android apps are created using java language
  Popular Java-based mobile apps:
    - Netflix
    - Tinder(online dating app)
    - Google Earth
    - Uber
- popular mobile operating systems Android is developed using Java
- Desktop GUI Applications

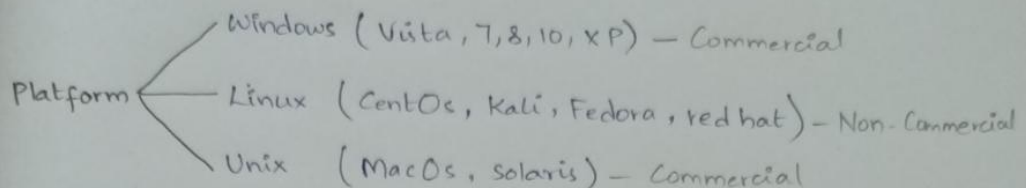
  e.g PDF Reader
- Popular web server “Tomcat” is developed using java
- For Developing games we use it
- Banking software uses It.
- Every SIM card uses Java
- More than 64,000 companies are using Java in the United States. For example, Google uses Java to build and develop Google Docs applications.

How a java program gets run

Due to the presence of Java Virtual Machine (JVM), JAVA language can run the code on any platform. That is why JAVA is called Platform - independent language.

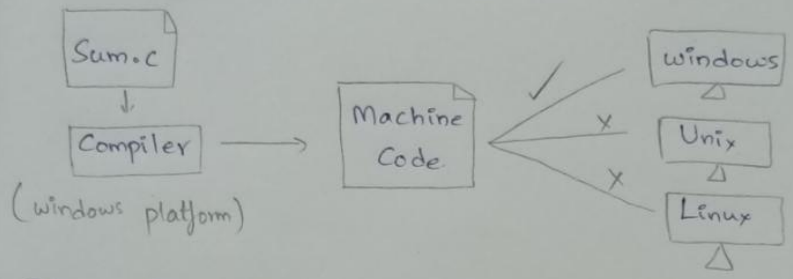
This JVM converts the Byte code to Machine code based on the platform being used and gives the required output.

* Operating systems are also called as System Software, platforms.

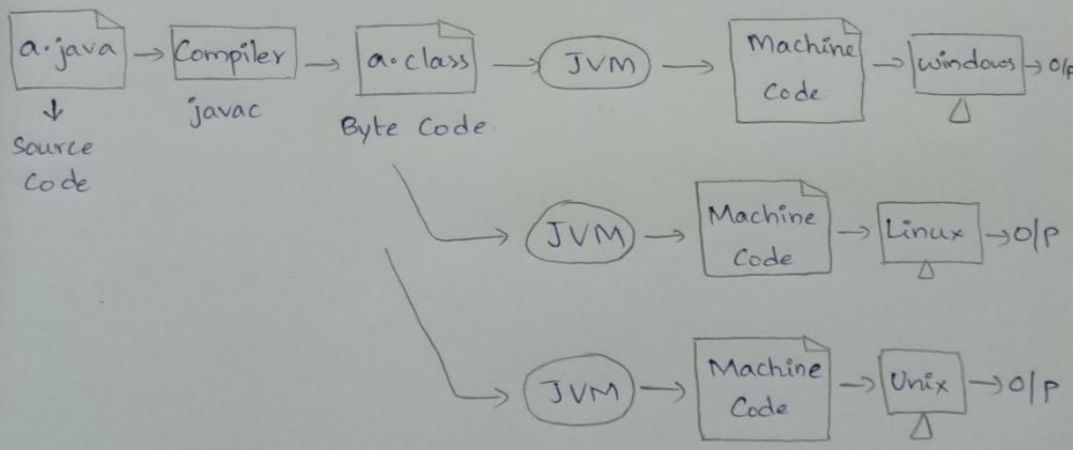


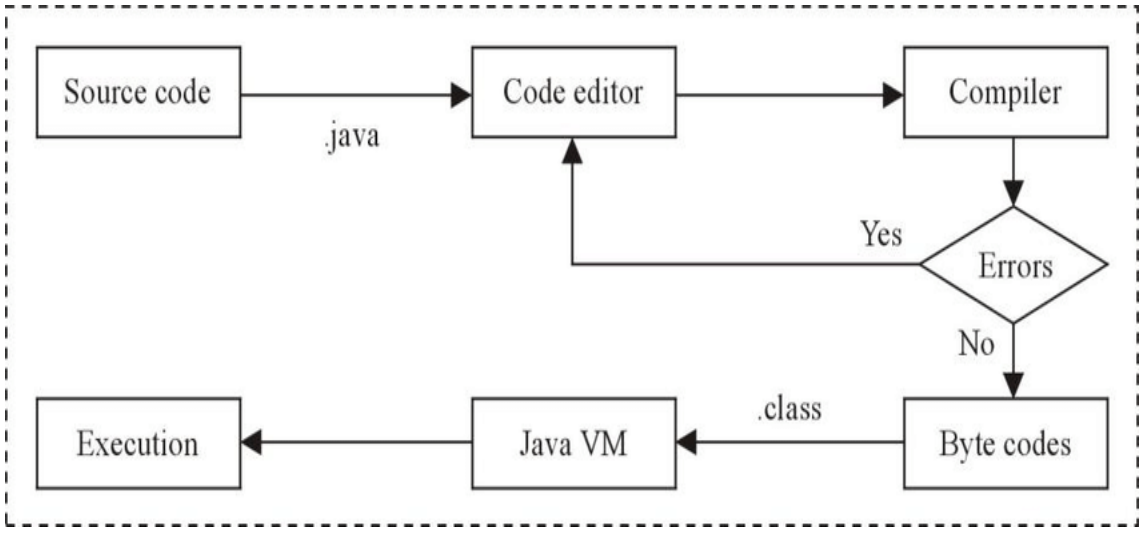
→ Machine code is platform dependent (Compiler)

→ C, C++ are platform dependant.



→ JAVA is called Platform-independant language.





Running ‘java programs’ Online without installing on you computer

- ★ Jdoodle
- ★ Programiz
- ★ codechef
- ★ w3schools.com
- ★ onecompiler.com
- ★ javatpoint
- ★ geeksforgeeks

Class and object

class	sbi
objects	ram,sam
attributes	accno,age,city
functions	credit(),debit(),showbal(),showme()

Members of a class

Variables and functions declared within a class,following are members of ‘sbi’ class

accno,age,city

credit(),debit(),showbal(),showme()

example 1

```
class sbi
{
     int age;
     String city;
     int accno;
     int bal=0;
     void credit(int c)
     {
        bal=bal+c;
     }

     void debit(int d)
     {
       bal=bal-d;
     }
     void showme()
     {
           System.out.println(accno+" "+age+" "+city+"\n");
     }
     void showbal()
     {
       System.out.println(bal+"\n\n");
     }
}
class bank
{
   public static void main(String args[])
   {
      sbi ram=new sbi();
```

```
        sbi sam=new sbi();
        ram.age=26;      // . operator is used to access members of a class with object
        ram.city="delhi";
        ram.accno=123;
        sam.age=29;
        sam.city="hyd";
        sam.accno=110;
        ram.credit(100);
        ram.debit(50);
        sam.credit(100);
        sam.debit(10);
        sam.showme();
        sam.showbal();
        ram.showme();
        ram.showbal();
    }

}
```

output

110 29 hyd

90

123 26 delhi

50

Explanation

Here ram,sam are objects of sbi class

Objects in java are created using ‘new’ operator

```
        Classname Objectname=new    Classname();
```

NOTE

Instantiating a class means creating objects for that class

---

example 2

```
public class book
{
   int price;
   String author;
   void show_book()
   {
       System.out.println(price+" "+author);
   }
   public static void main(String args[])
   {
      book java=new book();
      java.price=200;
      java.author="divya";
      java.show_book();
   }
}
```

output

200 divya

An object can be assigned to another object

```
class fruit
{
      float price_per_kilo;
      String color;
}
 class tasty
{
   public static void main(String args[])
```

```
   {
     fruit  apple=new fruit();
     fruit  kiwi=new fruit();
     apple.price_per_kilo=150.50f;
     apple.color="green";
     kiwi=apple;
     System.out.println("price: "+kiwi.price_per_kilo);
     System.out.println("colour: "+kiwi.color);
   }
}
```

output

```
price: 150.5
colour: green
```

Note:

Without using “class” you can't write a java program

Passing object reference to a function

```
class fruit
{
       float price_per_kilo;
       String color;
       void show_data(fruit f)
       {
              System.out.println("price: "+f.price_per_kilo);
              System.out.println("colour: "+f.color);
       }
}
 class tasty
{
   public static void main(String args[])
   {
     fruit  apple=new fruit();
```

```
      apple.price_per_kilo=150.50f;
      apple.color="green";
      apple.show_data(apple);

    }
}
```

output

price: 150.5

colour: green

Data Types

DATA	MEANING	DATA TYPE
1,2,3,55,789	integers	int
23.7,67.8,99.78	floating point	float OR double
'a','x','#','@'	characters	char
"sam","123","hello"	String values	String
true,false	boolean values	boolean

NOTE

IN JAVA “String" is data type as well as predefined "class"

Primitive types

Integer

byte (occupies 1 Byte in memory)

This sort of variables Stores whole numbers from -128 to 127

short (occupies 2 Bytes in memory)

This sort of variables Stores whole numbers from -32,768 to 32,767

int (occupies 4 Bytes in memory)

This sort of variables Stores whole numbers from -2,147,483,648 to 2,147,483,647

long (occupies 8 Bytes in memory)

This sort of variables Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

char (occupies 2 Byte in memory)

This sort of variables Stores a single character/letter or ASCII values

floating-point

float (occupies 4 Bytes in memory)

This sort of variables Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits

5.3456789

double(occupies 8 Bytes in memory)

This sort of variables Stores fractional numbers. Sufficient for storing 15 decimal digits

5.34567892689348

boolean

It has only two possible values true,false

This sort of variables Stores true or false values

example

```
class dtype
 {
         public static void main(String args[])

        {
             int x=7;
             float y=23.4545678923456f;
             double z=23.4545678923456;
             boolean p=true;
             char  q='#';
             String r="cse";
             System.out.println(x+"\n"+y+"\n"+z+"\n"+p+"\n"+q+"\n"+r);
         }
}
```

output

```
7
23.454567
23.4545678923456
true
#
cse
```

Reference types

- Class
- String
- Arrays

Reading input from keyboard

Most frequently used pre defined class to read data from keyboard is "Scanner"
"Scanner" class is present in "java.util" package

Use "import java.util.Scanner" at the beginning of program

Methods predefined in 'java.util.Scanner' class

nextBoolean()
: Reads a boolean value from the user

nextByte()
: Reads a byte value from the user

nextDouble()
: Reads a double value from the user

nextFloat()
: Reads a float value from the user

nextInt()
: Reads a int value from the user

nextLine()
: Reads a String value from the user

nextLong()
: Reads a long value from the user

nextShort()
: Reads a short value from the user

Example

```
import java.util.Scanner;
public class pqz
{
   public static void main(String args[])
   {
      String name;
      int age;
      float height;
      boolean indian;
      Scanner cse=new Scanner(System.in);
      System.out.println("enter ur name:");
      name=cse.nextLine();
      System.out.println("enter ur age:");
      age=cse.nextInt();
      System.out.println("enter ur height:");
      height=cse.nextFloat();
      System.out.println("you are indian?:");
      indian=cse.nextBoolean();
      System.out.println("Details u have entered are.....\n");
      System.out.println(name+" "+age+" "+height+" "+indian);
   }
}
```

Output

```
enter ur name:
sainath
enter ur age:
26
enter ur height:
5.3
you are indian?:
true

Details u have entered are.....
sainath 26 5.3 true
```

Type conversion

Implicit type conversion(automatic)

```
class automatic
{
   public static void main(String args[])
   {
      Int   r='A';//converting 'A' to 65
      System.out.println(r);
   }
}
```

output

```
A
```

```
public class MyClass
{
   public static void main(String args[])
   {
     char  e=65;
     System.out.println(e);
   }
}
```

output

```
A
```

```
public class MyClass
{
   public static void main(String args[])
   {
     float a;
     int b=98;
```

```
      a=b;
      System.out.println(a);
   }
}
```

Output

98.0

---

explicit type conversion(casting)

```
public class MyClass
{
   public static void main(String args[])
   {
      float a=34.56f;
      int b;
      b=(int)a;
      System.out.println(b);
   }
}
```

output

34

```
class div
{
        public static void main(String args[])
        {
                int x=7;
                int y=2;
                float z=x/y;
                System.out.println(z);
         }
}
```

output

3.0

```
class div
{
        public static void main(String args[])
        {
```

```
                int x=7;
                int y=2;
                float z=(float)x/y;
                System.out.println(z);
        }
}
```

output

3.5

Method(function) overloading

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

```
class over
{
    void many(int i,int j)
    {
       System.out.println(i+" "+j);
    }
    void many(String i,String j)
    {
       System.out.println(i+" "+j);
    }
    void many(int i,double j)
    {
       System.out.println(i+" "+j);
    }
}
class MyClass
{
   public static void main(String args[])
   {
      over o=new over();
      o.many(34,35);
      o.many("sai","ram");
      o.many(99,77.54);

   }
}
```

output

34 35
sai ram
99 77.54

Variable length arguments

```
class varargs
{
        void many(String...take)
        {
            System.out.println(take.length);
        }
}

class MyClass
{
    public static void main(String args[])
    {
        varargs v=new varargs();
        v.many("one","see","kiwi","i will meet you");
        v.many("one","see","kiwi");
        v.many("one","see");
    }
}
```

output

4

3

2

---

constructor

It is a function whose name is as same as class name

It is called automatically when object is created

It has no return type

```
class it
{
    it()
    {
            System.out.println("you are an it guy");
    }
}
class demo
{
```

```
    public static void main(String args[])
    {
        it namitha=new it();
        it kiran=new it();
        it sasi=new it();

    }
}
```

Output

```
you are an it guy
you are an it guy
you are an it guy
```

---

constructor overloading

Making single named constructor(method)to do more than one task

```
class it
{

    it()
    {
        System.out.println("i eat everything");
    }
    it(String i)
    {
        System.out.println("i like "+ i);
    }
    it(String i,String j)
    {
        System.out.println("i like "+ i+","+j);
    }
```

```
}
class food
{
    public static void main(String args[])
    {

        it kiran=new it();
        it sasi=new it("fish");
        it divya=new it("fish","prawns");

    }
}
```

Output

```
i eat everything
i like fish
i like fish,prawns
```

‘this’ keyword

It refers currently calling object

```
class it
{
    int age;
    int roll;
    it(int a,int r)
    {
        this.age=a;
        this.roll=r;
        System.out.println(this.age+" "+this.roll);
```

```
    }
}
class demo
{
    public static void main(String args[])
    {

        it kiran=new it(21,55);
        it sasi=new it(22,56);

    }
}
```

Output

```
21 55
22 56
```

Method(function) overloading

Making single named method to do more than one task

```
class it
{

    tasty()
    {
        System.out.println("i eat everything");
    }
    tasty(String i)
    {
        System.out.println("i like "+ i);
    }
    tasty(String i,String j)
    {
        System.out.println("i like "+ i+","+j);
    }
```

```
}
class food
{
    public static void main(String args[])
    {

        it kiran=new it();
           kiran.tasty();
        it sasi=new it();
           sasi.tasty(“fish”);
        it divya=new it();
           divya.tasty("fish","prawns");

    }
}
```

Output

```
i eat everything
i like fish
i like fish,prawns
---------------------------------------
```

Here,tasty() is overloaded

```
tasty()
tasty(String i)
tasty(String i,String j)
--------------------------------------------------------------------------
```

counting num of objects created

```
class feel
{
   static int c=0;
    feel()
    {
```

```
      c=c+1;
   }
}
class MyClass
{
   public static void main(String args[])
   {
      feel f1=new feel();
      feel f2=new feel();
      feel f3=new feel();
      System.out.println(feel.c);
   }
}
```

Output

3

Arrays

- ❖ Java array is an object which contains elements of a similar data type.
- ❖ Additionally, The elements of an array are stored in a contiguous memory location.
- ❖ It is a data structure where we store similar elements.

There are two types of array.

- Single Dimensional Array

  datatype arrayname[] =new datatype[size];

- Multidimensional Array

  datatype arrayname[][] =new datatype[rows][columns];
  datatype arrayname[][][] =new datatype[planes][rows][columns];

```
class Testarray
{
        public static void main(String args[])
        {
                int a[]=new int[5];//declaration and instantiation
                a[0]=10;//initialization
```

```
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;
        //traversing array
        for(int i=0;i<a.length;i++)//length is the property of array
        {
            System.out.println(a[i]);
        }
    }
}
```

Output

```
10
20
70
40
50
```

```
class Testarray
{
    public static void main(String args[])
    {
        int a[]={10,20,70,40,50};//declaration and initialization
        //traversing array
        for(int i=0;i<a.length;i++)//length is the property of array
        {
            System.out.println(a[i]);
        }
    }
}
```

Output

```
10
20
70
40
50
-----------------------------------
```

For-each Loop for Java Array

We can also print the Java array using for-each loop. The Java for-each loop prints the array elements one by one. It holds an array element in a variable, then executes the body of the loop.

The syntax of the for-each loop is given below:

```
for(data_type variable:array)
{
        //body of the loop
}
```

example

```
class Testarray
{
        public static void main(String args[])
        {
                int arr[]={33,3,4,5};
                //printing array using for-each loop
                for(int i:arr)
                {
                    System.out.println(i);
                }
        }
}
```

Output

33

3

4

5

Types of variables

- ★ Instance variables
  Variables declared within a class

- ★ Static variables
  Variables declared within a class with static keyword

- ★ Local variables
  Variables declared within a function OR block({ })

Static variables

Without static

```
class nostat
{
      int i=0;
}
class nostatdemo
{
   public static void main(String args[])
   {
       nostat n1=new nostat();
```

```
        n1.i=n1.i+1;
        nostat n2=new nostat();
        n2.i=n2.i+1;
        System.out.println("n1 i:"+n1.i);
        System.out.println("n2 i:"+n2.i);
    }

}
```

Output

```
n1 i:1
n2 i:1
```

With static

```
class stat
{
       static int i=0;
}
class statdemo
{
    public static void main(String args[])
    {
        nostat n1=new nostat();
        n1.i=n1.i+1;
        nostat n2=new nostat();
        n2.i=n2.i+1;
        System.out.println("n1 i:"+n1.i);
        System.out.println("n2 i:"+n2.i);
    }

}
```

Output

```
n1 i:2
n2 i:2
--------------------------------------------------------------------------------
```

static method(function)

static methods are called with class name

Without static

```
class nostat
{
       void see()
       {
           System.out.println("oops!");
       }
}
class nostatdemo
{
   public static void main(String args[])
   {
        nostat n1=new nostat();
        n1.see();
   }

}
```

Output

```
oops!
-------------------------------------
```

With static

```
class stat
{
      static  void see()
       {
          System.out.println("oops!");
       }
}
class statdemo
{
   public static void main(String args[])
   {
            nostat.see()
   }

}
```

Output

```
oops!
-------------------------------------
```

NOTE

A static method can directly access static data members ONLY

```
class abc
{
   int a=10;
   static int b=20;
   static void look()
   {
      System.out.println(a); //here ‘a’ is not static
   }

}
public class pqz
{
   public static void main(String args[])
   {
```

```
        abc n1=new abc();
        n1.look();
    }

}
```

Output

```
Error:   non-static variable ‘ a’ cannot be referenced from a static context
         System.out.println(a);
```

---

instanceof operator

```
class nostat
{

}
public class nostatdemo
{
    public static void main(String args[])
    {
        nostat n1=new nostat();
        System.out.println(n1 instanceof   nostat);
    }

}
```

Output

```
true
```

String handling

Generally, String is a sequence of characters.

But in Java, string is an object that represents a sequence of characters.

The ‘java.lang.String’ class is used to create a string object.

```
class a
{
```

```
}
class object
{
        public static void main(String ar[])
        {
                String s=new String();
                System.out.println(s.getClass());

        }
}
```

output

```
class  java.lang.String
-------------------------------------
```

```
Class  MyClass
{
   public static void main(String ram[])
   {
      String s=new String("sorry");//creating Java string by new keyword
      System.out.println(s);

   }
}
```

Output

sorry

'String' class methods

char charAt(int index)	It returns char value for the particular index
int length()	It returns string length
boolean equals(Object another)	It checks the equality of string with the given object.

boolean isEmpty()	It checks if string is empty.
String concat(String str)	It concatenates the specified string.
String toLowerCase()	It returns a string in lowercase.
String trim()	It removes beginning and ending spaces of this string.

Example

```
public class MyClass
{
   public static void main(String ram[])
   {
      String s="well";
      String t="well";
      String u="           well     said   ";
      String v="hi";
      String w="ramadas";
      System.out.println(s.length());
      System.out.println(s.charAt(1));
      System.out.println(s.equals(t));
      System.out.println(s.isEmpty());
      System.out.println(s.concat(t));
      System.out.println(u);
      System.out.println(u.trim());
      System.out.println(String.join("$",v,w));
      System.out.println(w.substring(4));
      System.out.println(w.substring(4,6));

   }
}
```

```
4
e
true
false
wellwell
           well     said
well     said
hi$ramadas
```

das

da

Control statements

Selection

- if
- if-else
- switch

Iteration

- for()
- while()
- do-while()
- for-each()

Jumping

- break
- continue
- return

Operators

Arithmetic

+

-

%

/

Logical

&&

||

!

Relational

<

>

==

!=
<=
>=
Bitwise
&
|
^
~
<<
>>
Assignment
=
+=
-=
/=
*=

Increment and decrement
++
--

Recursive function

```
import java.util.Scanner;
class recur
{
   public static void main(String args[])
   {
     long val;
     Scanner s=new Scanner(System.in);
     System.out.println("Enter a number:");
     int n=s.nextInt();
     val=fac(n);
     System.out.println(val);
   }
   static int fac(int z)
   {
      if(z==1)
      {
            return 1;
      }
      else
      {
         long result=z*fac(z-1);
         return result;
      }
   }
}
```

Output

```
Enter a number:
11
39916800

--------------------
```