Every class in Java is child class of object directly or indirectly so that object class methods by default available to every java class.

Hence Object class is considered as root/parent of all java classes.

Note:

1)If our class doesn’t extend any other class then only our class is the direct child class of object.

Example:

Class A

{

}

Here A is the child of Object.

2)If our class extends any other class then our class is indirect child class of object.

Class A extends B

{

}

Here A is child of B

Conclusion: Either directly or indirectly Java wont provide support for Multiple inheritance with respect to classes.

Object class defines the foloowing 11 methods.

Public string toString()

Public native int hashCode()

Public boolean equals(Object o)

Protected native Object clone() throws CloneNotSupportedException

Protected void finalize() throws throwable

Public final class getClass()

Public final void wait throws interruptedException

Public final native void wait(long ms) throws InterruptedException

Public final void wait(long ms, long nanoseconds) throws InterruptedException

Public native final void notify()

Public native final void notifyAll()

The above 11 methods present inside Object class.

Strictly Object class contains 12 methods. The extra method is registerNatives().

Private static native void registerNatives();

This method internally required for Object class and not available to the child classes.hence we are not required to consider this method.

1)toString():

We can use toString() method to get String representation of Object.

String s=Obj.toString();

Whenever we are tryign to print Object reference internally toString() method will be called.

Student s=new Student()

Syso(s); internally it is Syso(s.toString())

If our class doesn’t contain toString() method then object class toString() method will be executed.

Class Student{

String name;

Int rollno;

Student(String name, int rollno)

{

This.name=name;

This.rollno=rollno;

}

Public static void main(String[] args)

{

Student s1=new Student(“Nari”,101);

Student s2=new Student(“Mani”,102);

Syso(s1);

Syso(s1.toString());

Syso(s2);

}

}

Output:

Student@1888759

Student@1888759

Student@6e1408

In the above example, Object class toString() method got executed which is implemented as follows.

Public String toString(){

Return getClass.getName() + “@” + Integer.toHexString(hashcode);

}

className@Hashcode in Hexadecimal form.

Based on our requirement we can override toString() method to provide our own String representation.

For example whenever we are trying to print Student object reference to print his name and roll no we have to override toString() method as follows.

Public String toString()

Return name + “…” + rollno;

}

In all wrapper classes, in all collection classes, string class , string buffer and stringbuilder classes toString() method is overridden for meaningful string representation. Hence it is highly recommended to override toString() method in our class also.

Class Test

{

Public String toString()

{

Return “test”;

}

Public static void main(String[] args)

{

String s=new String(“durga”);

Syso(s); //durga

Integer i=new Integer(10);

Syso(I); //10

ArrayList l=new ArrayList();

l.add(“A”);

l.add(“B”);

Syso(l); //[A,B]

Test t=new Test();

Syso(t); //Test