2.2. The Object class

Object class is a special class in java.If no inheritance is precise for the classes then all those classes are derived class of the **Object** class. We can consider ,**Object** is a superclass of all other classes by default. therefore

Public class A{.....}is equal to public class A extends Object{.....}

A reference variable of type Object can refer to any object of additional classes.

The package java.lang.Object includes below specified method

Method	**Purpose**
Object clone()	Creates a new object that is similar to object being cloned
boolean equals(Object object)	Concludes whether one object is similar to another
void finalize()	Called by an unused object is used again
class getclass()	Holds the class of an object at run time
inthashcode()	Returns the hash code associated with the invoking object
void notify()	Resumes execution of a thread waiting on the invoking object
void notifyall()	Resumes execution of all threads waiting on the invoking objecg
String toString()	Returns a string that describes the object
void wait() void wait(long milliseconds) void wait(long milliseconds,int nanoseconds)	Waits on another thread of execution

toString() Method of Object Class

The toString function returns the string type value.The syntax is**public String toString()**

If we call up the toStringmethod,by default then it gives a string which describes the objectThis returned string contains the character "@" and object's memory address in hexadecimal form.

We can identify with the idea of toString() method by using as it is and overriding it with appropriate string.

Example:Illustration1

```
class A extends Object
{
}
class B extends A
{
}
class ObjectClassDemo
{
public static void main(String args[])
{
A obj=new A();
System.outprintln("Obj:"+obj);
```

```
System.outprintln("obj.toString():"+obj.toString());
}
}
```

Output:
obj:A@3e25a5
obj.toString():A@3e25a5

Example:Illustration2

```
class A extends Object
{
public String toString() //method is overriden
{
String str=”Hello”;
returnstr;
}
}
class B extends A
{
}
classObjectClassDemo
{
public static void main(String args[])
{
A obj=new A();
System.outprmtln("Obj:”+obj);
System.outprmtln("obj.toStrmg():”+obj.toStrmg());
}
}
```

Output:
Obj:Hello
Obj.toString():Hello

Example2:

```
Import java.awt.*;
class StringDemo
{
public static void main(String args[])
{
Point c=new Point(10,20); //Explicitly call toString() on object as part of string concatenation
System.outprmtln("C=”+c.toString()); //Using the default object.toString() method
System.outprintln("C=”+c);     //Implicitly call toString() on object as part of string concatenation
String s=c+”testing”;
System.out.println(s);
}
}
```

Output:
C=java.awt.Point[x=10,y=20]

C=java.awt.Point[x=10,y=20]
java.awt.Point[x=10,y=20] testing

Equals method of Object class

The method equals is helpful for comparing values given by two objects.

Example1:

```
class A extends Object
{
int a=10;
public Boolean equals(Object obj)
{
if(objinstanceof B)
{
return a==((B)obj).b;
}
else
return false;
}
}

class B extends A
{
int b=10;
}
class ObjectClassDemo1
{
public static void main(String args[])
{
A obj1=new A();
B obj2=new B();
System.outprintin("The two values of a and b are equal:"+obj1.equals(obj2));
}
}
```

Output:
The two values of a and b are equal:true

2.3. ABSTRACT CLASS AND METHODS

The base class is supposed to be the most common or less particular.Sometimes base class is so common and less specific that it does not anything but lists out only general features of different classes.Then such a base class is referred as abstract class.

For example,In java program we have formed three classes.

- class A is a base class consists of two methods namely fun1() and fun2(),
- The class B and class C are derived from class A
- .The class A is an abstract class since it contains one abstract method fun1().
- We have defined this method as abstract because,itsdefinition of fun1() is overridden in the derived classes B and C.
- an another function of class A that is fun2() is a regular function.

Definition of function overridden:
When a method of base class and derived classes consists of similar return type ,function name and parameters are called as function overridden.

Example1:Abstract Class and Methods

```
abstract class A //abstract class
{
abstract void fun1(); //abstract method
void fun2() //normal method
{
System.outprmtin("A:In fun2”):
}
}

class B extends A
{
void fun1()                    //function overridden from abstract class
{
System.outprmtin("B:In funl”);
}
}

class C extends A
{
void fun1()                    //function overridden from abstract class
{
System.outprmtin("C:In funl”);
}
}
}

public class AbstractClsDemo
{
public static void main(String args[])
{
B b=new B();
C c=new C();
b.fun1();
b.fun2();
c.fun1();
c.fun2();
}
}
```

Output:

```
B:In fun1
A:In fun2
C:In fun1
A:In fun2
```

Example2:

```
abstract class Base
{
abstract void fun();
}
class Derived extends Base
{
void fun()
{
System.outprmtln("Derived fun() called”);
}
}
class Main()
{
public static void main(String args[])
{
Base b=new Derived();
b.fun();
}
}
```

Output:
Derived fun() called

Rules for writing abstract classes and abstract methods

1 .An abstract method must be there in an abstract class only.It should not be there in an nonabstract class.

2 .In all the non-abstract subclasses extended from an abstract base class all the abstract methods must be implemented.An un-implemented abstract method in the derived class is not acceptable.

3 .Abstract class cannot be instantiated by the new operator

4 .A constructor method of an abstract class can be defined and can be called by the derived classes.

5 .A class that consists of abstract method must be abstract but the abstract class may not include an abstract method.This class is simply used as a base class for defining new subclasses

6 .A derived class can be abstract but the derived class can be concrete.

7 .The abstract class cannot be instantiated by new operator but an abstract class can be used as a datatype.