**Object class-**

It is the parent class of all the classes in java.

It is called as topmost class of java which is present in java.lang package.

Every class in Java is directly or indirectly derived from the object class. If a Class does not extend any other class then it is direct child class of object**.**

There are different methods of object class are as follows.

* public final Class getClass()
* public int hashCode()
* public boolean equals(Object obj)
* protected Object clone() throws CloneNotSupportedException
* public String toString()
* public final void notify()
* public final void notifyAll()
* public final void wait()
* public final void wait(long timeout)
* public final void wait(long timeout, int nanos)
* protected void finalize()throws Throwable

1. Public final Class getClass()-

This class is used to get the metadata of class.

**package** com.test;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Test test = **new** Test();

System.***out***.println(test.getClass().getName());

System.***out***.println(test.getClass().getSimpleName());

}

}

Output-

com.test.Test

Test

1. public int hashCode()-

For every object unique number is generated by JVM called as hashcode. It does not represent object of address then what is the use of hashcode. it will store into bucket based on hashcode.

Note-

1. The most of java native method are written in c or c++ that why not able show the body.
2. if two objects are equal, their hashcode will be same.If two object hashcode are same, you cannot guaranty that objects are equal.

Example

**package** com.test;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

Test1 test2 = **new** Test1();

Test1 test3 = **new** Test1();

System.***out***.println(test2.hashCode());

System.***out***.println(test3.hashCode());

}

}

Output

2018699554

1311053135

1. public boolean equals (Object obj)-

It compares the given object to this object. There are two equals method, this equals method is used to check the address of string not contents.

Example-1

**package** com.test;

**public** **class** Test3 {

**int** empId;

String empName;

**public** **static** **void** main(String[] args) {

Test3 test3 = **new** Test3();

test3.empId = 1;

test3.empName = "ashok";

Test3 test4 = **new** Test3();

test4.empId = 2;

test4.empName = "ram";

System.***out***.println(test3.equals(test4));

}

}

Output

False

**package** com.test;

**public** **class** Test3 {

**int** empId;

String empName;

**public** **static** **void** main(String[] args) {

Test3 test3 = **new** Test3();

test3.empId = 1;

test3.empName = "ashok";

Test3 test4 = **new** Test3();

test4.empId = 2;

test4.empName = "ram";

test3 = test4;

System.***out***.println(test3.equals(test4));

}

}

Output

True

1. protected Object clone() throws CloneNotSupportedException-

It creates and returns the exact copy (clone) of this object.

Example-1

**package** com.test;

//clone method- create copy of objects

**public** **class** Test4 **implements** Cloneable {

**int** x;

**public** **static** **void** main(String[] args) **throws** CloneNotSupportedException {

Test4 test4 = **new** Test4();

test4.x = 50;

System.***out***.println("first object is>>" + test4.x);

Object test5 = test4.clone();

System.***out***.println("second object is>>" + test5.toString());

}

}

Output

first object is>>50

second object is>>com.test.Test4@7852e922

Here, second output line, we will get address instead of value. To solve this issue, we should override toString method.

Example-2

**package** com.test;

//clone method- create copy of objects

**public** **class** Test4 **implements** Cloneable {

**int** x;

**public** **static** **void** main(String[] args) **throws** CloneNotSupportedException {

Test4 test4 = **new** Test4();

test4.x = 50;

System.***out***.println("first object is>>" + test4.x);

Object test5 = test4.clone();

System.***out***.println("second object is>>" + test5.toString());

}

@Override

**public** String toString() {

**return** "Test4 [x=" + x + "]";

}

}

Output

first object is>>50

second object is>>Test4 [x=50]

1. public String toString() –

It returns the string representation of this object.

**package** com.test;

**public** **class** Employee {

**int** id;

String employeeName;

String employeeCity;

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", employeeName=" + employeeName + ", employeeCity=" + employeeCity + "]";

}

**public** **static** **void** main(String[] args) {

Employee employee = **new** Employee();

employee.id = 10;

employee.employeeName = "ajay";

employee.employeeCity = "pune";

System.***out***.println(employee);

}

}

Output

Employee [id=10, employeeName=ajay, employeeCity=pune]

public final void notify()-

It wakes up single thread, waiting on this object's monitor.

public final void notifyAll()-

It wakes up all the threads, waiting on this object's monitor.

public final void wait(long timeout)throws InterruptedException()-

It causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).

public final void wait(long timeout,int nanos)throws InterruptedException-

It causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method).

public final void wait()throws InterruptedException

It causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).

protected void finalize()throws Throwable

It is invoked by the garbage collector before object is being garbage collected.