**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.

**Methods of object class:**

There are different methods of object class are as follows.

1. public final Class getClass()-

This class is used to get the metadata of class.

package com.wipro.jpmorgan;

public class Example {

public static void main(String[] args) {

Example example = new Example();

System.out.println(example.getClass().getName());

System.out.println(example.getClass().getSimpleName());

}

}

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.

HashCode in Java is a function that returns the hashcode value of an object on calling. It returns an integer or a 4 bytes value which is generated by the hashing algorithm.

The most of java native method are written in c or c++ that why not able show the body.

Note- 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.wipro.jpmorgan;

public class Example {

public static void main(String[] args) {

Example example1 = new Example();

Example example2 = new Example();

System.out.println(example1.hashCode());

System.out.println(example2.hashCode());

}

}

1. public Boolean equals (Object obj)-

It compare 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-

package com.velocity;

public class Employee {

int empId;

String empName;

public static void main(String[] args)

{

Employee emp1 = new Employee();

emp1.empId = 1;

emp1.empName = "Ashok";

Employee emp2 = new Employee();

emp2.empId = 2;

emp2.empName = "Sachin";

System.out.println(emp1.equals(emp2));

}

}

Output- false

1. protected Object clone() throws CloneNotSupportedException-

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

Example-

package com.wipro.jpmorgan;

public class Example implements Cloneable {

int x;

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

Example example1 = new Example();

example1.x = 50;

System.out.println("First Object data is>>" + example1.x);

Object example2 = example1.clone();

System.out.println("Second Object data is>>" + example2);

}

}

1. public String toString() –

It returns the string representation of this object.

package com.wipro.jpmorgan;

public class Example {

int x;

@Override

public String toString() {

return "Example [x=" + x + "]";

}

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

Example example1 = new Example();

example1.x = 50;

System.out.println("First Object data is>>" + example1);

}

}

1. public final void notify()-

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

1. public final void notifyAll()-

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

1. 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).

1. 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).

1. public final void wait()throws InterruptedException

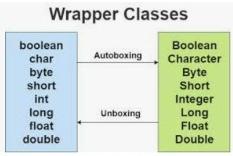
It causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).

1. protected void finalize()throws Throwable

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

**Wrapper class-**

It provides the mechanism to convert primitive’s data type into object and object into primitives data type called as wrapper class.



Process of converting primitive’s data type into object called as “Autoboxing.” And process of converting object into primitive’s data type called as “Unboxing.”

There are 8 classes of java.lang.package are known as wrapper classes in java.

* Integer
* Short
* Byte
* Long
* Double
* Character
* Boolean
* Float

**Autoboxing:**

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Example-

**package** com.object;

**public** **class** WrapperDemo {

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

//Converting int into Integer

**int** a=20;

Integer i=Integer.*valueOf*(a);//converting int into Integer explicitly

Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a) internally after 1.5V

  }

}

**Unboxing:**

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

e.g.

package com.object;

public class WrapperDemo {

public static void main(String[] args) {

//Converting Integer to int

Integer a=new Integer(3);

int i=a.intValue();//converting Integer to int explicitly

int j=a;//unboxing, now compiler will write a.intValue() internally

}

}

**Final keyword**-

We can apply final to variables, method and class.

**Final variable-**

A variable which is declared with final keyword called as final variables.

Once you assigned any value to that variables then it won’t be changed. It works like constants in java.

How to declare the final variables-

final int a=5;

Example-1

class FinalDemo{

public static void main(String args[]){

final int a=5;

System.out.println(a);

}

}

Output-

5

Example-2

class FinalDemo{

public static void main(String args[]){

final int a=5;

for (int a=5; a<=10;a++){

System.out.println(a);

}

}

}

In this example, we will get compile time error, final variable values does not changed.

**Final method-**

Method which is defined with final keyword called as final method.

How to declare the final method-

public final void test(){

//business logic here.

}

Note- Final method cannot be overridden.

Example-3

class X{

final void test(){

System.out.println(“This is x class-test method”);

}class Y extends X{

final void test(){

System.out.println(“This is y class-test method”);

}

public static void main(String args[]){

X x= new Y();

x.test();

}

}

Output-

Nothing

In this example, we will get compile time error final method cannot be override final method from X

Final class-

The class which is defined with final keyword called as final class.

How to declare the final class

final class Test{

//business logic

}

How you stop others from inheriting your class-

By making class as final.

final class X {

public final void test() {

System.out.println("x class-test method");

}

}

class Y extends X {

}

public class FinalKeyword {

public static void main(String[] args) {

Y y = new Y();

y.test();

}}