

Object class and its methods

Object class is parent of every class in java and it is present in java.lang package and java.lang is default package of java means we not need to import in your project.

Why Object class is parent of every class in java?

Because Object class provide the some inbuilt method to us which is used for perform the operation on any class object like as check the equality of different object, convert the object to string, perform object cloning, perform garbage collection on object etc

Example:

Programmer	intentrally meaning
class A { }	class A extends java.lang.Object { }

Object class provide the some inbuilt method to us to perform operation on object just we have to override this methods in class on which object we want perform operations.

Object class methods given below

boolean equals(Object): this method is used for perform comparison between two different object using hash code and if objects equal then return true otherwise return false.

String toString(): this method is used for convert the object in to the string format.

Object clone(): this method is used for create the duplicated copy or shadow copy of object

void finalize(): this method normally we use for resource cleaning purpose then method call when we perform garbage collection on object.

void wait(): this method is used for perform unconditional wait with object in multi threading.

void wait(int milliseconds): this method perform the conditional wait with object in multi threading.

void notify(): this method is used for call the waited object in first in first out format or in queue format.

void notifyAll(): this method is used for call the all waited threads in last in first out format .

static block : static block get executed in program very first even before main method of class and static block call only once when we create the object of class.

Class getClass(): this method is used for create the run time instance of class using reflection api

Now we will discuss about the equals() method of Object class

Before learn the equals() method we will see the one example and we will use the equals method for cover the limitation of given example.

```
package org.techhub;
class Test
{
    int no;
    Test(int x)
    {
        no=x;
    }
}
```

```

}
public class ObjCompApplication {

    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);

        if(t1==t2)
        {
            System.out.println("Objects are equal");
        }
        else
        {
            System.out.println("Objects are not equal");
        }
    }
}

```

If we think about above code we have the two objects t1 and t2 we provide the same value to t1 as 100 and t2 as 100 but program generate the output to us Objects are not equal.

Q. why program generate the output to us objects are not equal even values of both objects are same?

Because in the case of java Objects not compare on basis of its value object get compare on basis of its hash code.

Q. what is the Hash Code ?

Hashcode is like as address object or hashcode is unique integer number provided by JVM to every object in Ram and JVM not generate the same hashcode for different objects.

If we want to see the hashcode of object we have the method name as `System.identityHashCode(object)`

Example

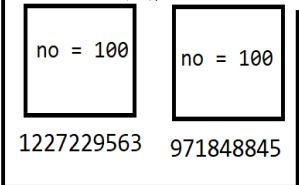
```
package org.techhub;
class Test{
    int no;
    Test(int x)
    {no=x;
    }
}
public class ObjCompApplication {
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);
        System.out.println("Hash code of t1 "+System.identityHashCode(t1));
        System.out.println("Hash code of t2 "+System.identityHashCode(t2));
        if(t1==t2)
        {System.out.println("Objects are equal");
        }
        else
        {System.out.println("Objects are not equal");
        }
    }
}
```

```

class Test
{
    int no;
    Test(int x)
    {
        no=x;
    }
}

Test t1 = new Test(100); Test t2 = new Test(100);

```



```

public class ObjCompApplication {
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);

        System.out.println("Hash code of t1 "+System.identityHashCode(t1));
        System.out.println("Hash code of t2 "+System.identityHashCode(t2));
        if(t1==t2) 1227229563== 971848845
        {
            System.out.println("Objects are equal");
        }
        else
        {
            System.out.println("Objects are not equal");
        }
    }
}

```

If we think about above diagram we have the object t1 with hash code 1227229563 and t2 with 971848845 and when we execute the statement `t1 == t2` then JVM perform comparison between hash code of object means JVM compare 1227229563 with 971848845 and these hash code are not equal so we get output Objects are not equal If we want to solve this problem Object class provide the two methods to us

boolean `equals(Object)` and int `hashCode()`;

How to override the `equals()` method of Object and Perform Object Comparison

Steps to work with `equals()` method

1) Declare the class and override the `equals ()` method from Object class

```

class Test
{
    int no;
    Test(int x)
    {

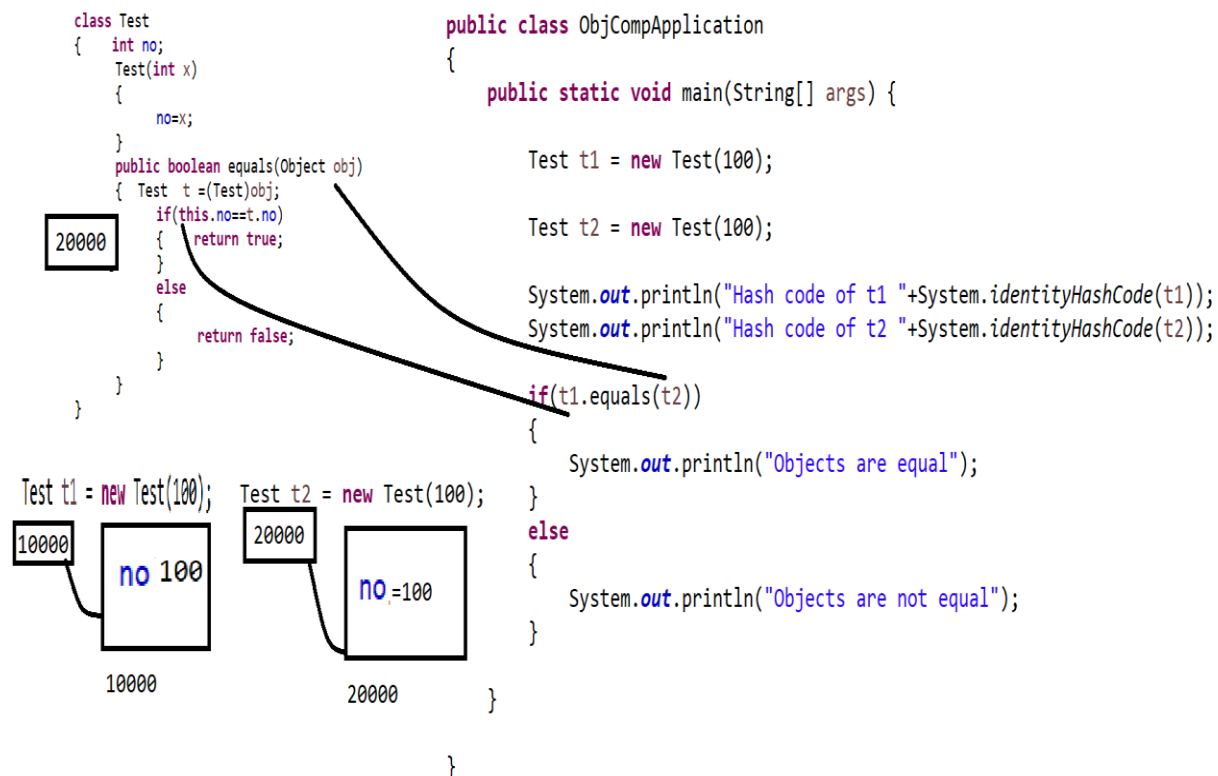
```

```
        no=x;
    }
    public boolean equals(Object obj)
    { Test t=(Test)obj;
        if(this.no==t.no)
        { return true;
        }
        else
        {
            return false;
        }
    }
}
```

2) Call the equals() method for Object Comparison purpose

Syntax: objectname1.equals (objectname2): when we call this method then objectname2 pass asparameter to equals() method and objectname1 is calling object means objectname1 is work as this reference.

We will discuss above concept with following diagram



If we think about above diagram then we have the two objects t1 and t2 so t1 and t2 contain same value i.e no=100 when we call the statement t1.equals(t2) then equals() method get executed then left hand object i.e calling object points to this reference and second object pass as parameter means as per our example t1 points to this and t2 points to t so here t1 and t2 having same value i.e 100 and 100 so equals() method return true value in if statement so we get output objects are equals.

Rules of object comparison in java

- 1) If two objects are equals then their hash code should be equal
- 2) If values two objects are equal then hash code should be equal and object should be equal.

Following example demonstrate the above concept

```

package org.techhub;
class Test
{
    int no;
    Test(int x)
    {
        no=x;
    }
    public boolean equals(Object obj)
    {
        Test t =(Test)obj;
        if(this.no==t.no)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
public class ObjCompApplication
{
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);
        System.out.println("Hash code of t1 "+System.identityHashCode(t1));
        System.out.println("Hash code of t2 "+System.identityHashCode(t2));
        if(t1.equals(t2))
        {
            System.out.println("Objects are equal");
        }
        else
        {
            System.out.println("Objects are not equal");
        }
    }
}

```



```
}
Hash code of t1 1227229563
Hash code of t2 971848845
Objects are equal
```

If we think about above code and output we have two objects are equals but their hashcode are different means above mention rules by java not satisfy here so if we want to satisfy this rule we have to override the hashcode method.

Note: JVM not generate the same hashcode for different object internally so if we want to satisfy above mention we have override the hashcode method and generate the hash code when two objects are equal and generate the two different hashcode when have two different objects.

Example

```
package org.techhub;
class Test {
    int no;

    Test(int x) {
        no = x;
    }

    public boolean equals(Object obj) {
        Test t = (Test) obj;
        if (this.no == t.no) {
            return true;
        } else {
            return false;
        }
    }
}
```

```

    public int hashCode() {
        return no * 1000;
    }
}

public class ObjCompApplication {
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(200);
        if (t1.equals(t2)) {
            System.out.println("Objects are equal");
            System.out.println("Hash code of t1 " + t1.hashCode());
            System.out.println("Hash code of t2 " + t2.hashCode());
        } else {
            System.out.println("Objects are not equal");
            System.out.println("Hash code of t1 " + t1.hashCode());
            System.out.println("Hash code of t2 " + t2.hashCode());
        }
    }
}

```

Q. Why we need to override hashCode for generate the dummy hashCode when we perform Object Comparision

Some time when we store the user defined objects in Set collection and the rule of Set Collection is cannot store the duplicated data but when we add the new element in Set Collection then Set Collection cross verify hashCode of object and in java every object having different hashCode generated by JVM even objects values are same in this case your set Collection may be store the duplicated data and if we want to avoid this problem we have to override the equals() and hashCode in class whose object want to store in Set Collection.

WAP to create the Set Collection and in Set Collection we want to store the Employee class objects but we want to employee detail should be unique cannot duplicate employee record.

```
package org.techhub;

public class Employee {

    private int id;
    private String name;
    private int sal;

    public Employee(String name,int id,int sal)
    {
        this.name=name;
        this.id=id;
        this.sal=sal;
    }
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSal() {
```

```

        return sal;
    }

    public void setSal(int sal) {
        this.sal = sal;
    }
    public boolean equals(Object obj)
    {
        Employee e=(Employee)obj;
        if(e.id==this.id && e.name.equals(this.name) && e.sal==this.sal)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public int hashCode()
    {
        return id*10;
    }
}

```

Mainmethod

```

package org.techhub;
import java.util.*;
public class EmployeeSetApplication {
    public static void main(String[] args) {
        LinkedHashSet employeeSet = new LinkedHashSet();
        Employee emp1 = new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,1000);
        Employee emp3 = new Employee("Dinesh",3,1000);
        Employee emp4 = new Employee("Ram",1,1000);
        employeeSet.add(emp1);
        employeeSet.add(emp2);
        employeeSet.add(emp3);
        employeeSet.add(emp4);
    }
}

```

```

        for(Object obj:employeeSet)
        {
            Employee e=(Employee)obj;
            System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal()+"\t"+e.hashCode());
        }
    }
}

```

toString() method of Object class

toString() method is used for convert the user defined object in to the string format

Normally we use the toString() in three cases

- 1) Convert the user objects in to the string format
- 2) When we want to display the object content using System.out.println()
- 3) Some time when we store the user defined objects in Collection and when print object then object data not visible in collection so we have to override the toString() and display it.

Example

```
package org.techhub;
```

```

class Student {
    private int id;
    private String name;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {

```

```

        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private float per;
}

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

        Student s1 = new Student();//s1 contain id name and percentage
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(50.5f);
        System.out.println(s1);
    }
}

```

If we think about the above code then we print the s1 object using System.out.println() then s1 not print its content we get output in following fashion.

org.techhub.Student@4926097b

But we want to display the content of s1 using System.out.println() then we have to convert the s1 in to the string by using toString() method of Object class means we have to override the toString() of Object class in Student class and return the student content in the form of String and capture and display it.

When we want to display the object content using System.out.println()

Example

```
package org.techhub;
```

```
class Student {  
    private int id;  
    private String name;  
  
    public int getId() {  
        return id;  
    }  
    public void setId(int id) {  
        this.id = id;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public float getPer() {  
        return per;  
    }  
    public void setPer(float per) {  
        this.per = per;  
    }  
    private float per;  
  
    public String toString()  
    {  
        return "["+name+","+id+","+per+"]";  
    }  
}  
  
public class StudentApplication {  
    public static void main(String[] args) {  
  
        Student s1 = new Student();//s1 contain id name and percentage  
        s1.setId(1);  
        s1.setName("Ram");  
    }  
}
```

```

        s1.setPer(50.5f);

        System.out.println(s1); //s1.toString()
    }
}

```

Some time when we store the user defined objects in Collection and when print object then object data not visible in collection so we have to override the toString() and display it

Example with Collection

```

package org.techhub;
import java.util.*;
class Student {
    private int id;
    private String name;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private float per;
}

```



```

    public String toString()
    {
        return "["+name+","+id+","+per+"]";
    }
}
public class StudentApplication {
    public static void main(String[] args) {

        Student s1 = new Student();//s1 contain id name and percentage
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(50.5f);

        Student s2 = new Student();//s1 contain id name and percentage
        s2.setId(2);
        s2.setName("Shyam");
        s2.setPer(60.5f);

        ArrayList al = new ArrayList();
        al.add(s1);
        al.add(s2);
        System.out.println(al);

    }
}

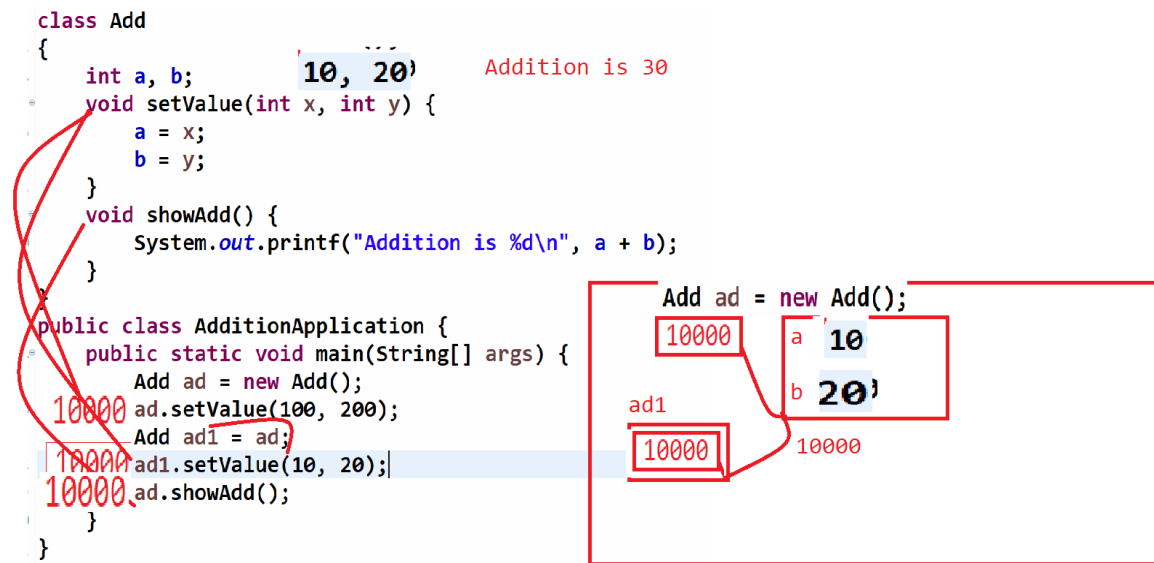
```

Object clone (): this method is used for create the duplicated or clone copy of object.

Q. Why we need to perform Object cloning in Java?

Some time if we apply more than one reference on single object and if we try to change the object by using any reference then object original state or variables or values get change and if we want to persist the object previous values in this case we can use the object cloning concept.

Given Diagram shows the need of object cloning concept



If we think about above diagram we create the statement name as Add
 ad=new Add() means we create the object in memory and we store the
 address of object in ad reference we consider the address of object is 10000
 means when we call the method name as ad.setValue(100,200) then value
 stored on 10000 address space means 100 and 200 stored on 10000 address
 space when we call the statement Add ad1=ad means we assign the reference
 of ad or address of ad in ad1 means address of ad1 is 10000 means ad1 points
 to the location where ad points means the location or address space of ad and
 ad1 is same means when we call the method name as ad1.setValue(10,20)
 then 10 and 20 get override in a and b on 10000 address space and when we
 call the ad.showAdd() function then we get answer Addition is 30.
 If we want to solve this problem we have to use the Object cloning concept
 And we want to perform object cloning concept we have the following steps.

Steps to work with Object cloning

1) Create the class and implement the Cloneable interface

```

class Add implements Cloneable
{
    int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
}

```

```

    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
}

```

2) Create the one user defined method and return the clone of object using clone() method and handle the CloneNotSupportedException

```

class Add implements Cloneable
{
    int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
    public Add getInstance() throws CloneNotSupportedException {
        Object obj = this.clone();
        Add ad=(Add)obj;
        return ad;
    }
}

```

3) Call the method from object clone get return

```

class Add implements Cloneable
{
    int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
    public Add getInstance() throws CloneNotSupportedException {
        Object obj = this.clone();
        Add ad=(Add)obj;
        return ad;
    }
}

```

```

    }
}
public class AdditionApplication {
    public static void main(String[] args)throws Exception {
        Add ad = new Add();
        ad.setValue(100, 200);
        Add ad1 = ad.getInstance(); //we get object clone
        ad1.setValue(10, 20);
        ad.showAdd();
    }
}

```

In the case of object cloning we have the two copies of object shadow copy and deep copy. Deep copy means original object created by using new keyword and shadow copy means duplicated copy of object created by clone method.

Source Code Example

```

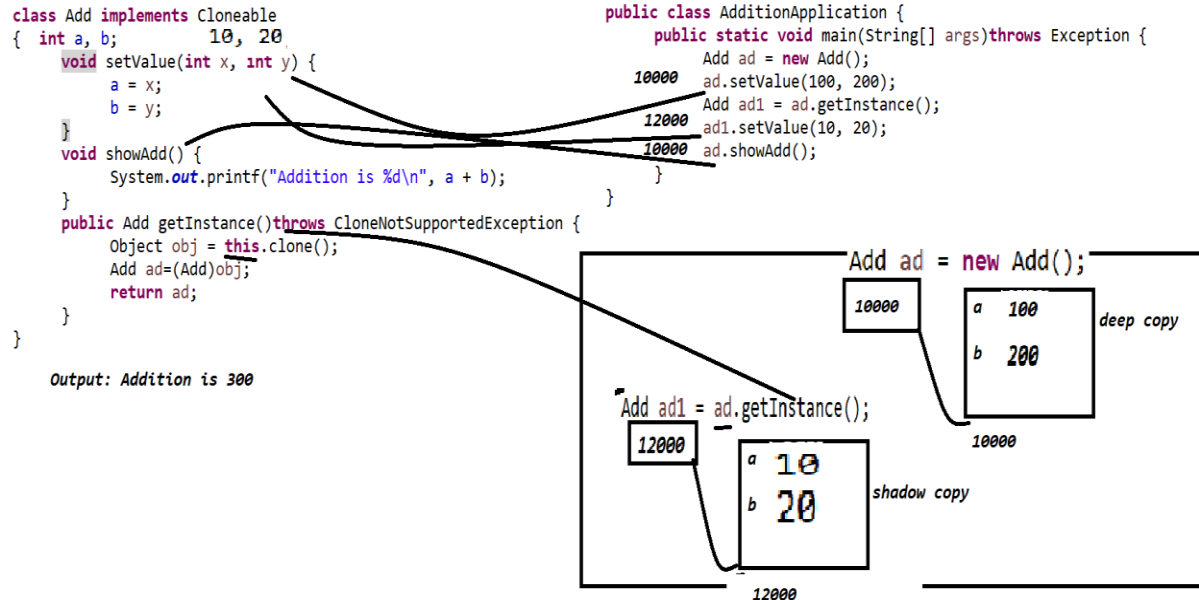
package org.techhub;
class Add implements Cloneable
{ int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
    public Add getInstance()throws CloneNotSupportedException {
        Object obj = this.clone();
        Add ad=(Add)obj;
        return ad;
    }
}
public class AdditionApplication {
    public static void main(String[] args)throws Exception {
        Add ad = new Add();
    }
}

```

```

        ad.setValue(100, 200);
        Add ad1 = ad.getInstance();
        ad1.setValue(10, 20);
        ad.showAdd();
    }
}

```



If we think about above code then we have the Statement `Add ad=new Add()` means we create object by using new keyword as per our diagram we have the address of object is 10000 and we store this object in ad reference when we call the method `ad.setValue(100,200)` these values are stored on 10000 address which is created by using `new Add()` means deep copy of object. when we call the statement `Add ad1=ad.getInstance()` so using this method JVM return the duplicated copy of object whose address stored in Add and return the new object address in ad1 means as per our example `this.clone()` statement return the duplicated of object whose address is 10000 and give new address to new object in our example 12000 means ad1 points to 12000 object called as shadow copy or object clone copy so when we call the statement `ad1.setValue(10,20)` it will override on 12000 address so when we execute statement `ad.showAdd()` then we get answer Addition is 300 because we not perform change on address space 10000

Q. what is the diff between object creation by using new keyword and by using clone() method.

When we create the object by using new keyword then constructor get executed and when we create the object by using clone () then constructor not executed. New object create the new constructed block in heap area of memory and clone() method create the object using previous constructor memory by using new keyword in heap.

Static block in Object class

We can write the two blocks or initializer in class

1) Static block or static initialize: static block is member of object class If we want to work with static block in java we have the some important points.

a) static block call very first in class even before main method of class

```
1 package org.techhub;
2
3 public class StaticBlockApp {
4
5     static
6     {
7         System.out.println("Hey I am static block");
8     }
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         System.out.println("Hey i am main method");
12     }
13 }
14
```



```
<terminated> StaticBlockApp [Java Application] C:\Users\Admin\
Hey I am static block
Hey i am main method
```

Static block call once means in program means if we declare the static block in particular class and if we create the multiple object of that class but static block run only once in program.

```

class A
{
    A()
    {
        System.out.println("I am constructor");
    }
    static
    {
        System.out.println("I am static block");
    }
}
public class StaticBlockApp {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
        A a3 = new A();
    }
}

```

```

I am static block
I am constructor
I am constructor
I am constructor

```

2) no static block or non static initializer :

Non static block call at the time of object creation means non static block call before constructor of class and non static block call every time when we create the object of class.

```

class A
{
    A()
    {
        System.out.println("I am constructor");
    }
    static //static initializer or static block
    {
        System.out.println("I am static block");
    }
    //instance initializer or non static block
    System.out.println("I am non static initializer");
}
public class StaticBlockApp {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
        A a3 = new A();
    }
}

```

```

I am static block
I am non static initializer
I am constructor
I am non static initializer
I am constructor
I am non static initializer
I am constructor

```

void finalize () method : this method is used for perform garbage collection or resource cleaning purpose this method call automatically when we

initialize the reference of class as null and call System.gc() method .
 System.gc() method destroy the object from memory which is not use by any reference called as garbage collection and when we call the System.gc() then finalize() method get executed means we can write the logic in finalize() for release the resource at the time of object destruction.

Source Code

```
package org.techhub;
class A
{
    A()
    { System.out.println("I am constructor");
    }
    static //static initializer or static block
    {System.out.println("I am static block");
    }
    { //instance initializer or non static block
        System.out.println("I am non static initializer");
    }
    public void finalize()
    { System.out.println("I can release resource");
    }
}
public class StaticBlockApp {
    public static void main(String[] args) {
        A a1 = new A();
        a1=null;
        System.gc();
    }
}
```

Output

```
I am static block
I am non static initializer
I am constructor
I can release resource
```