Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

Collections

- Object Class:
  - Object class is a root class for all classes in java.
  - Every class in java directly or indirectly inherit object class.
  -The properties (variables) & behaviours (methods) of object class used by JVM for manipulating objects.
  - This class provide 11 methods.

1) public String toString()
   - This method return String representation of the object.
   - This method is executed by JVM whenever program print reference variable.

Example:

```
class A{
        int x =10;
}
class Demo{
        public static void main (String arg[]){
                A obj = new A();
                System.out.println(obj);
        }
}
```

Source code in toSting():

```
Public String toString(){
        Return                  getClass().getname()+"@"+
        Integer.toString (hashcode());
}
```

   - toString method is override inorder to print content of object.

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

Example:

```
Class Employee{
    private int empno;
        private String ename;
        Employee(in tempno, String ename){
            this.empno = empno;
            this.ename = ename;
    }
    public String toString(){
        return empno+" " +ename+" "+ super.to
        string();
    }
}
class Demo2{
public static void main(String args[]){
    Employee emp1= new Employee (101, "Seema");
    Employee emp2 = new Emmployee (102, "Sita");

    System.out.println(emp1);
    System.out.println(emp2);
    }
}
```

2) Public Boolean equals(object):

- Indecates whether some other object is "Equal to " This one
- The equals method implements on equivalence relation on non-null object reference

It is reflexive:

For any non-null reference value x.equals(x) should return true.

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

a. It is Symmieric:
   for any non-null reference values x&y , x.equals(y) should return true iff y.equals(x) return true.

b. It is transitive:
   For any non null values x.y& 2. if x. equals(y) return true then x.equals(z) should return true.

c. It is consistent:
   - For any non-null reference values x& y, multiple invocation of x.equals (y) consistently return false, provided no information used in equals comparisons on the object is modified
   - For any non-null reference values x, x.equals(null) should return false.

Examples:

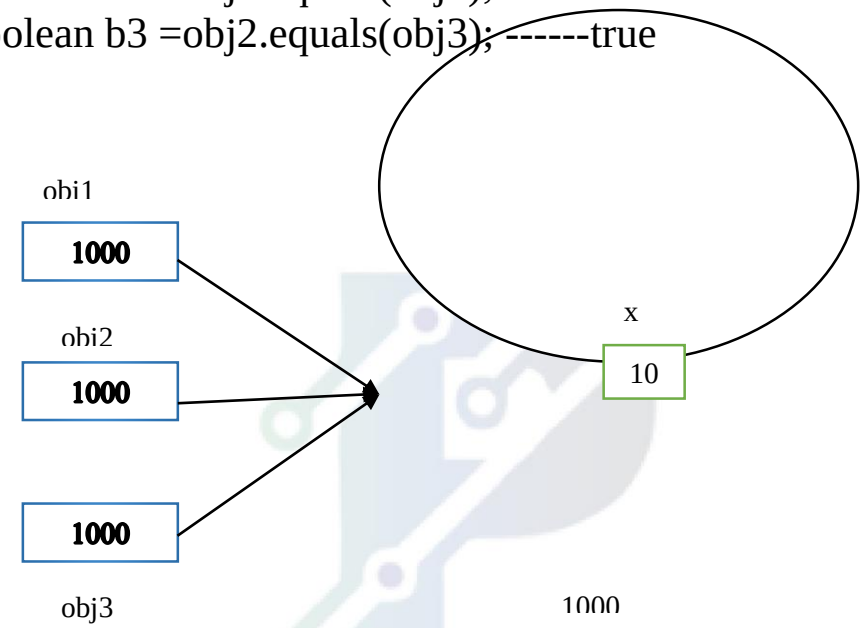
1)
```
class A{
        int x;
        A(int x){
                This.x=x;
        }
}

A obj1=new A(10);
boolean b1 = obj1.equals(obj1);
```

2)
```
A obj1=new A(10);
A obj2=new A(20);
boolean b1= obj1.equals(obj2); --------false
A obj3=obj1;
```

boolean b2=obj1.equals(obj3); ---------true

3) A obj1 = new A(10);
A obj2 = obj1;
A obj3 = obj2;
boolean b1 =obj1.equals(obj2); ------True
boolean b2 =obj1.equals(obj3); ------true
boolean b3 =obj2.equals(obj3); ------true



- equals method is override in order to compare contents of objects
- By default equals method method compare reference but not state.

Example:

```
class Student{
        private int mo;
        private String name ;
        Student(int mo, String name){
            this.mo=mo;
            this.name=name;
```

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

```
        }
        public boolean equals(object o){
        Student s = (Student)o; ----------
```

Reference narrowing type conversion

```
        if(mo== s.mo)
            return true;
        else
            return false;
        }
        }
class Demo3{
        public static void main(Stirng arg[]){
            Student std1=new Student(101,"Roma");
            Student std2=new Student (102, "Om");
            boolean b1 = stud1.equals(stud2);
            System.out.println(b1);
        }
}
```

equals(=)	double equals(==)
1) it is a method of object class	1) it is operator
2) It is used for comparing reference type	2) It is used for comparing primitive and reference types
3) This method by default compare object of references chashcode. It can be override inorder to compare state of objects	3) It compares only object references . java Does not support operator overlooding

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

3) Public int hashCode():

- Return a hash code value for object This method is supported for the benefits of hashtables such as those provides by java. util. hashtable
- the contract of hashcode is:

  Where it is invoked on same object more then once during on execution of Application , me hashcode method must consistently return the same integer ,provided to information used in equals comparisons on the object is modified . This integer is need not remain consistent from one execution of an application to another execution of same application.
- If two object are equals according to the equals (object) method, then calling the hashcode method then calling the hashcode method one each of two objects must produce the same integer result.
- It is not required that if two objects are unequal according to the equals(java.lang.object) method then calling the hashcode method on each of two objects must produce distance integer result. However the programmer should be aware that producing distinct integer result for unequal objects may improve the performance Of hashtables.

Example:

```
class A{
        private int x;
        A (int x){
            this.x=x;
        }
        public boolean equals(object o){
```

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

```
                A a = new (A) o;
                if (x==a.x)
                return true;
                else
            return false;
            }
            public int hashcode(){
                return x;
            }
}
class Demo1{
            public static void main(String[] args)
            {
                A obj1=new A(10);
                A obj2= new A(10);
                A obj3= new A(20);
                Boolean b1 = obj1.equals(obj2);
                boolean b2 =  on=bje1.equas(boj3);

                System.out.println(b1);
                System.out.println(b2);
                System.out.println(obj.hashcode());
                System.out.println(obj2.hashcode());
                System.out.println(obj3.hashcode());
                boolean b3= obj1== obj2;
                System.out.println(b3);
            }
}
```

4) Protected void Finalize() :

- called by the garbage collection on an object when garbage collection determines that are no more reference to the object.
- Garbage collection is identiflying the objectand garbage

Phone
+91 8767591462| +91 7822869695
Website
www.payastechnologies.com

collector is removing the object

Example:

```
class A{
        int x;
        protected void finalize(){
```