VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

Object class

Objectives

This session will give the knowledge about

- Object class
- Object Cloning

The Cosmic Class – The Object Class

- Java defines a special class called Object. It is available in java.lang package
- All other classes are subclasses of Object
- Object is a superclass of all other classes; i.e., Java's own classes, as well as user-defined classes
- This means that a reference variable of type Object can refer to an object of any other class

The Object Class: Methods

boolean equals(Object object) - Determines whether one object is equal to another.

```
class Base{
}
public class Main {
	public static void main(String[] args) {
		Base b1=new Base();
		Base b2=new Base();
		System.out.println(b1.equals(b2));
		System.out.println("Welcome".equals("Welcome"));
	}
}
```

The Object Class: Methods

void finalize() - Called before an unused object is reclaimed from the heap by the garbage collector

```
public class Main {
        public static void main(String args[]) {
                Main ob1 = new Main();
                Runtime r = Runtime.getRuntime();
                ob1 = null;
                r.gc();
        }
        protected void finalize() {
                System.out.println("I am finalize");
        }
}
```

The Object Class: Methods

final Class getClass() - Obtains the class of an object at runtime

```
public class Main {
        public static void main(String args[]) {
                Main ob1 = new Main();
                System.out.println(ob1.getClass());
                String name="test";
                System.out.println(name.getClass());
        }
}
```

The Object Class: Methods

final void notify() - Resumes execution of a thread waiting on the invoking object

final void notifyAll() - Resumes execution of all waiting threads on the invoking object.

Waits on another thread of execution.

- final void wait(long milliseconds)
- final void wait(long milliseconds, long nanoseconds)

The Object Class: Methods

String toString() - Returns a string that describes the object.

```
public class Main {
        public String toString(){
                return "i am to string";
        }
        public static void main(String args[]) {
                Main ob1 = new Main();
                System.out.println(ob1);
                System.out.println(ob1.toString());
        }
}
```

Object Cloning

The object cloning is a way to create exact copy of an object. The clone() method of Object class is used to clone an object.

The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create. If a class implement Cloneable interface, clone() method generates CloneNotSupportedException.

The clone() method is defined in the Object class. Syntax:

```
public class Main implements Cloneable {

}
```

Object Cloning

Object.clone() is protected, so we have to provide our own clone() and indirectly call Object.clone() from it.

If you want to write a clone method in a child class then all of its super classes should define the clone() method in them.

Object.clone() supports only shallow copying but we will need to override it if we need deep cloning.

Object Cloning: Example

```
public class Main implements Cloneable
{
      int id=1257;
      Main(){
            id=1247;
      }
    public static void main(String args[])
    {
    try{
        Main obj1=new Main();
        System.out.println(obj1.id);
        Main obj2=(Main) obj1.clone();
            System.out.println(obj2.id);
    }
    catch (CloneNotSupportedException e) {
    System.out.println(e);
    }
    }
}
```

Object Cloning: Example

```
public class Main
{
      int id=1257;
      Main(){
            id=1247;
      }
    public static void main(String args[])
    {
        Main obj1 = new Main();
        System.out.println(obj1.id);
        Main obj2 = obj1;
        System.out.println(obj2.id);
        }
}
```

Object Cloning

What is the difference between cloning the object with .clone() method and = sign?

- Object obj = new Object(); //creates a new object on the heap and links the reference obj to that object
- Object obj2 = obj; //there is only one object on the heap but now two references are pointing to it.
- Object obj2 = obj.clone(); //creates a new object on the heap, with same variables and values contained in obj and links the reference obj2 to it.

Quiz: Guess the output

```
public class Main implements Cloneable
{
    public static void main(String args[])
    {
        Main obj1=new Main();
        System.out.println(obj1);
    }
}
```

Object Cloning: Example

```
class Test1 {
        int x;
}

class Test2 extends Test1 implements Cloneable {
        int a;

        public Object clone() throws CloneNotSupportedException {
                return super.clone();
        }
}
```

Object Cloning: Example

```
public class Main {
        public static void main(String args[]) {
                try{
                Test2 t1 = new Test2();
                t1.a = 10;
                t1.x = 30;

                Test2 t2 = (Test2) t1.clone();
                t2.a = 100;
                t2.x = 300;
```

Object Cloning: Example

```
                System.out.println(t1.a+" "+t1.x);
                System.out.println(t2.a+" "+t2.x);
                }
                catch(CloneNotSupportedException e){
                        System.out.println(e);
                }
        }
}
```

Object Cloning: Quiz

```
class Student implements Cloneable {
        int rollno;
        String name;

        Student(int rollno, String name) {
                this.rollno = rollno;
                this.name = name;
        }

        public Object clone() throws CloneNotSupportedException {
                return super.clone();
        }
```

Object Cloning: Quiz

```
        public static void main(String args[]) {
                try {
                        Student s1 = new Student(101, "arun");
                        Student s2 = (Student) s1.clone();
                        System.out.println(s1.rollno + " " + s1.name);
                        System.out.println(s2.rollno + " " + s2.name);
                } catch (CloneNotSupportedException c) {
                }
        }
}
```

Summary

We have discussed about

- Object class
- Object Cloning