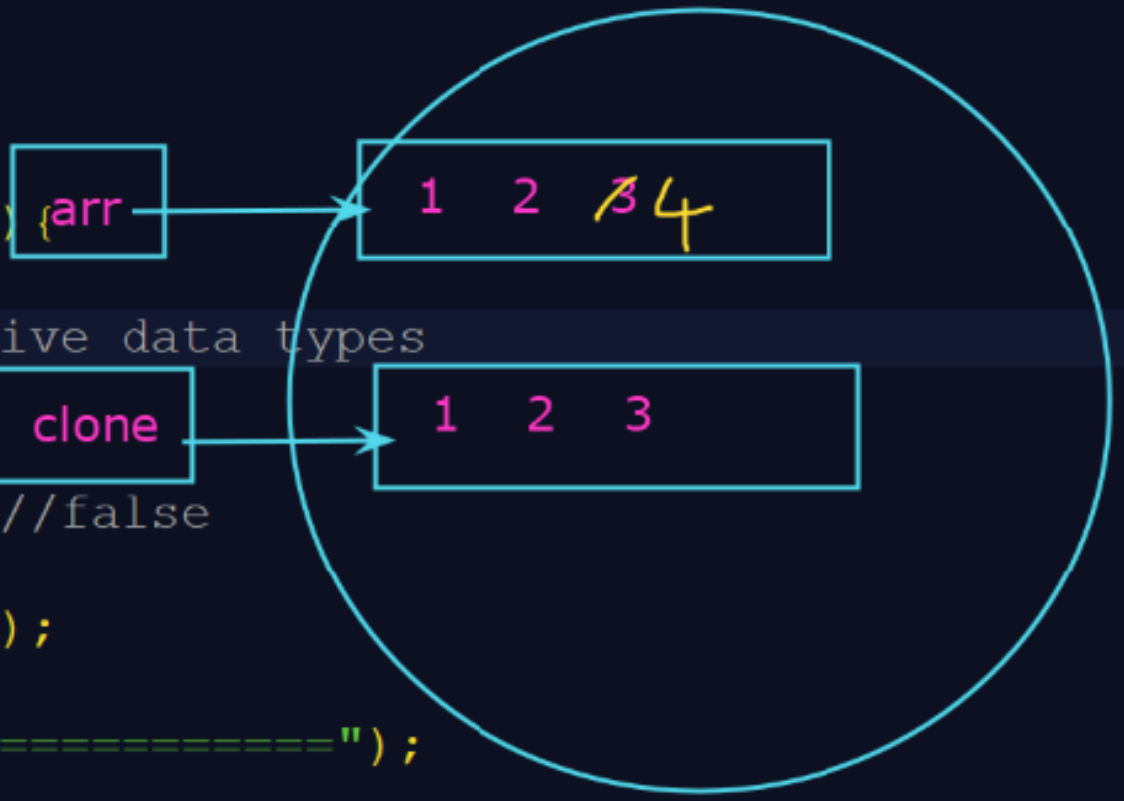
Object Oriented Programming with Java (OOPJ)

Session 7: OOPS Pillar

Kiran Waghmare

```java
//Cloning of an array
//2 ways: 1.Deep copy & 2. Shallow copy

class ArrayDemo11{

    public static void main(String args[]){

        //1d Array : Deep copy for primitive data types
        int[] arr = {1,2,3};
        int[] clone = arr.clone();
        System.out.println(arr == clone);//false
        System.out.println("arr="+arr);
        System.out.println("clone="+clone);

        System.out.println("==========================");

        //2d Array : Elements: Shallow copy
        int[][] matrix = {{1,2},{3,4}};
        int[][] copy = matrix.clone();
        System.out.println(matrix[0][0] == copy[0][0]); //true

    }
}
```

Date: 02/09/2025
Day 7 : OOPJ

Topics-
- Array
- OOPS pillars
- Abstraction
- Encapsulation

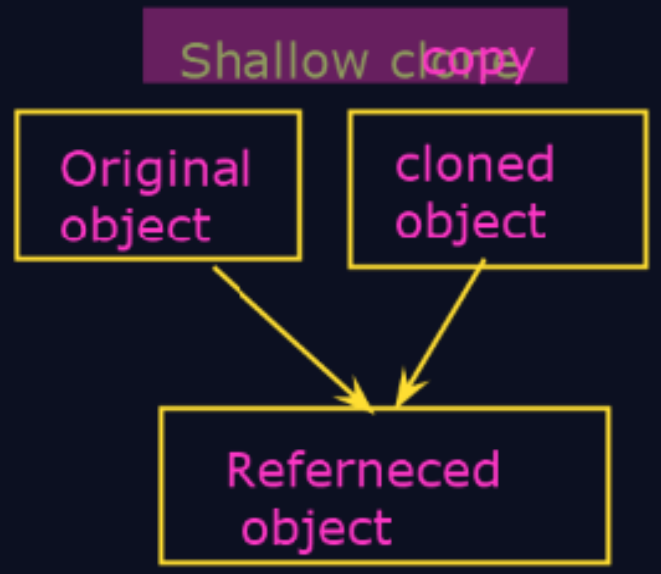
Deep coy and shallow copy:

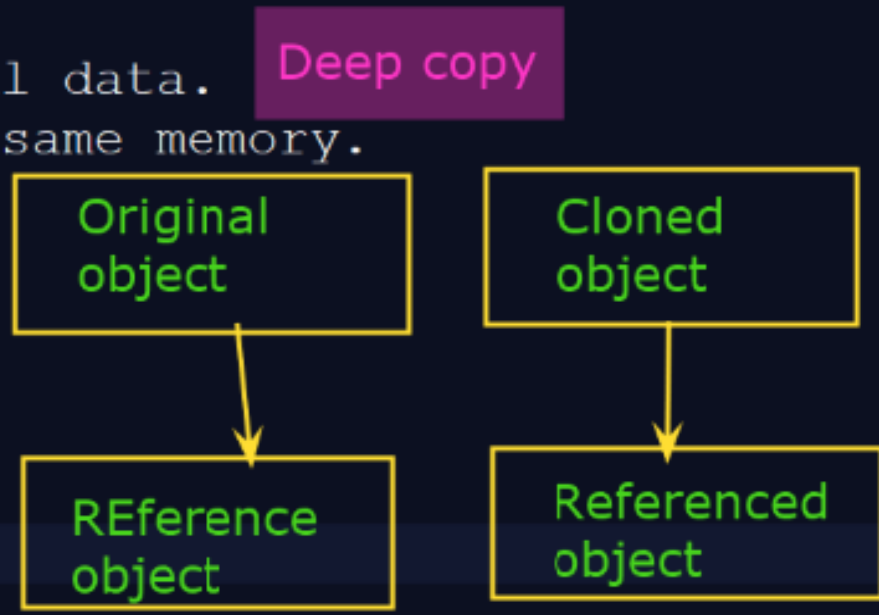
Shallow copy:

-Copies the reference of the object , not the actual data.
-Both the original and copied object point tot the same memory.
-Changes in one object reflect in the other object.

Deep Copy:

-Copies the actual data into a new object.
-The original and copied objects are independent.
-Changes in one object do not affect the other.

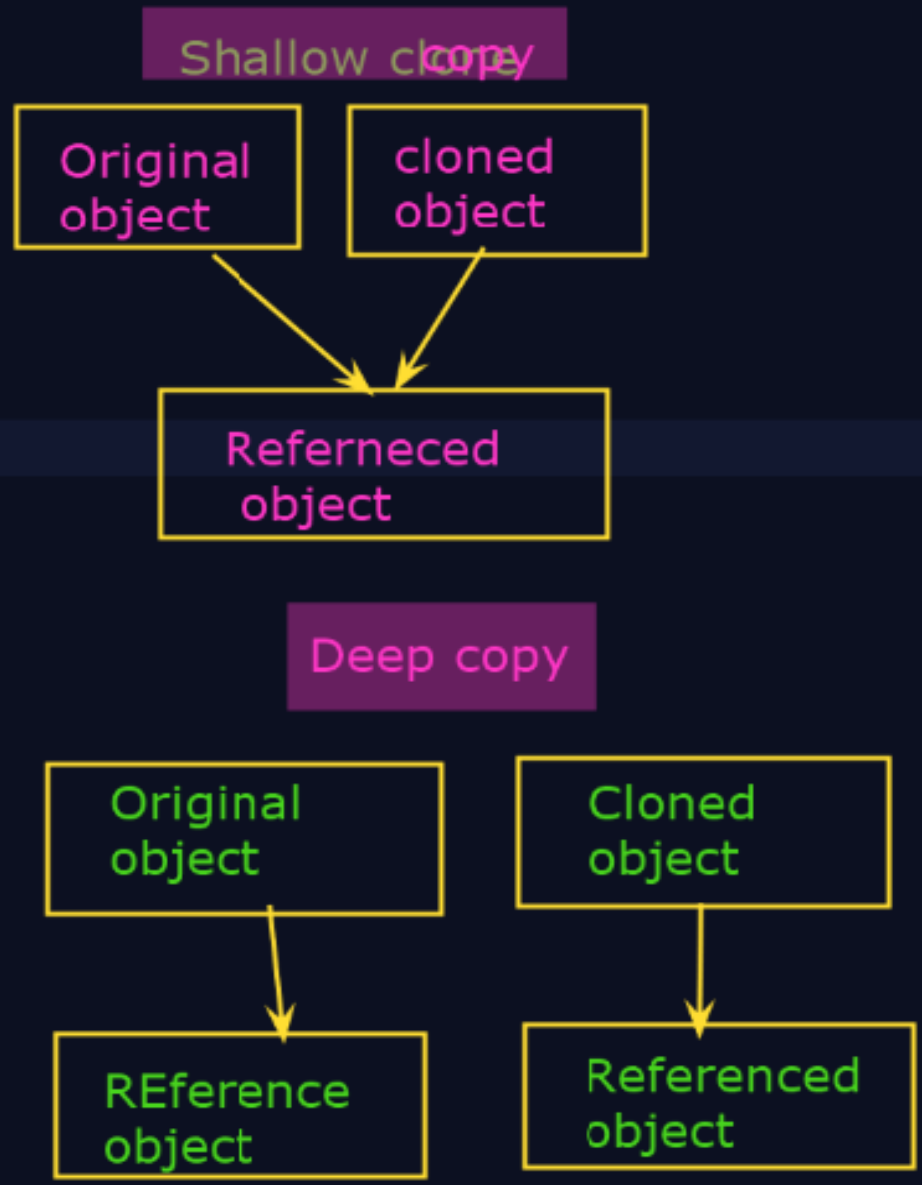




```java
//Cloning of an array
//2 ways: 1.Deep copy & 2. Shallow copy

class ArrayDemo11{

    public static void main(String args[]){


    }
}
```

Audio Video Participants Chat Share Pause Layout Annotate Remote contr

You are screen sharing HI Stop share

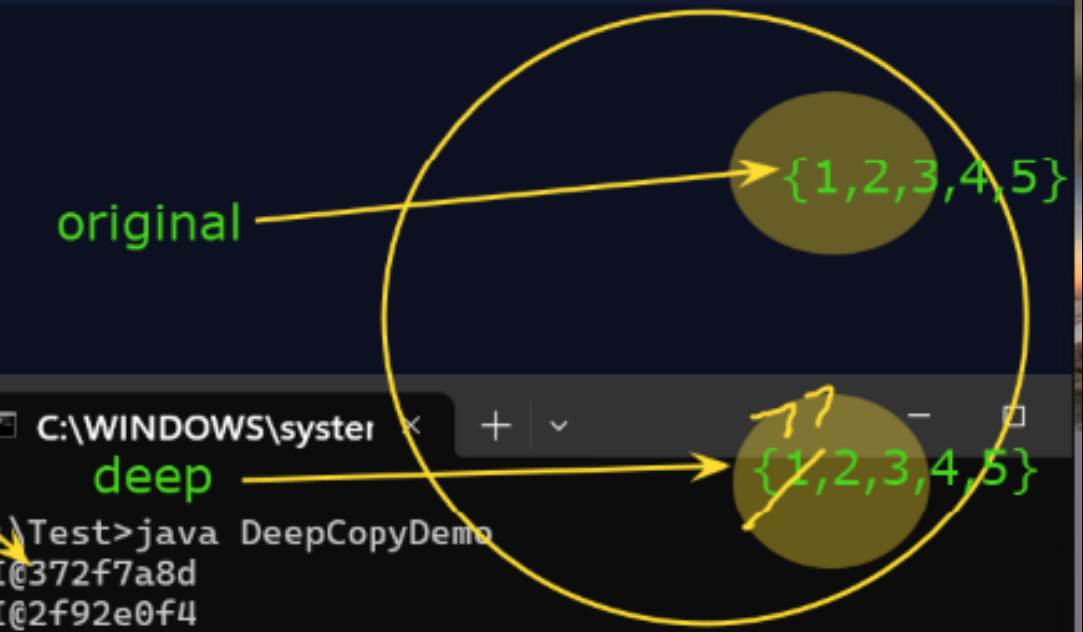


```java
class ShallowCopyDemo{

    public static void main(String args[]){
        int[] original = {1,2,3,4,5};
        int[] shallow = original;

        System.out.println(original);
        System.out.println(shallow);

        shallow[0] = 7;
        System.out.println("original");
        for(int i : original){
            System.out.println(i);
        }

        System.out.println("----------");
        System.out.println("shallow");
        for(int i : shallow){
            System.out.println(i);
        }
    }
}
```

```
C:\Test>javac ShallowCopyDemo.java

C:\Test>java ShallowCopyDemo
[I@372f7a8d
[I@372f7a8d
original
7
2
3
4
5
----------
shallow
7
2
3
4
5

C:\Test>
```

c1=c2

```java
class DeepCopyDemo{

    public static void main(String args[]){
        int[] original = {1,2,3,4,5};
        int[] deep = original.clone();

        System.out.println(original);
        System.out.println(deep);

        deep[0] = 77;
        System.out.println("original");
        for(int i : original){
            System.out.println(i);
        }

        System.out.println("----------");
        System.out.println("Deep");
        for(int i : deep){
            System.out.println(i);
        }
    }
}
```




```
C:\WINDOWS\syster
C:\Test>java DeepCopyDemo
[I@372f7a8d
[I@2f92e0f4
original
1
2
3
4
5
----------
Deep
77
2
3
4
5

C:\Test>
```

```java
interface Shape{//interface
    double calculateArea();// abstract method
}

class Rectangle implements Shape{
    double length, width;

    Rectangle(double length, double width){
        this.length = length;
        this.width = width;
    }

    @Override
    public double calculateArea(){
        return length * width;
    }

}

public class InterfaceDemo{

    public static void main(String args[]){

        //Rectangle r1 = new Rectangle(4.0,5.0);
        Shape r1 = new Rectangle(4.0,5.0);
        //Shape s1 = new Shape();//Error: object cannot be instantiated
        double res = r1.calculateArea();
        System.out.println("Result = "+ res);

    }
}
```

```
    }
}

class Demo implements Shape{

}

Encapsulation:
--------------

class: instance variable, method()
```