

Q.1) Difference between deep copy and shallow copy.

Shallow Copy.

- Creates a new object.
- Copies primitive data types directly.
- Copies reference variables as references (not actual objects).
- Inner/nested objects are shared between original and copied object.
- Both objects point to the same memory location for nested objects.
- Changes made to shared objects reflect in both copies.
- Faster than deep copy.
- Uses less memory.
- Default behavior of `Object.clone()` in Java.

Deep Copy.

- Creates a new object.
- Copies primitive data types directly.
- Copies reference variables by creating new objects.
- Entire object hierarchy is duplicated.
- No shared references between original and copied object.
- Changes in copied object do not affect original.
- Slower than shallow copy.
- Uses more memory.
- Must be implemented manually in Java.

Q. 2) What are varargs in Java?

- Varargs means variable-length arguments.
- It allows a method to accept zero or more arguments of the same data type.
- Represented using ellipsis (...) after the data type.
- Syntax: `datatype... variableName`
- Internally, varargs are treated as an array.
- Introduced in Java 5.
- Improves code readability and reduces method overloading.
- Only one varargs parameter is allowed per method.
- Varargs parameter must be the last parameter in the method.
- Can be used with both primitive types and object types.
- The method can also be called with no arguments.
- Useful when the number of inputs is not fixed at compile time.
- Helps avoid writing multiple overloaded methods for different argument counts.

Q. 3) What is autoboxing and unboxing?

Autoboxing

- Autoboxing is the automatic conversion of a primitive data type into its corresponding wrapper class object.
- Done automatically by the Java compiler.
- Introduced in Java 5.
- Helps in working with Collections (ArrayList, HashMap, etc.), because collections store objects only, not primitives.
- Improves code readability by removing manual conversion.
- int a = 10;
- Integer obj = a; // Autoboxing

Unboxing

- Unboxing is the automatic conversion of a wrapper class object into its corresponding primitive type.
- Also done automatically by the compiler.
- Makes it easy to use wrapper objects in arithmetic operations.
- Integer obj = 20;
- int b = obj; // Unboxing

Q. 4) What are the differences between == and equals() for

Strings?

== Operator

- Compares references (memory address).
- Checks whether two references point to the same object in memory.
- Does not compare content.

-Ex:

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = new String("Java");  
        String s2 = new String("Java");  
        System.out.println(s1 == s2); // false  
    }  
}
```

-Because s1 and s2 are two different objects in memory.

2. equals() Method

- Compares content (values) of the Strings.
- Defined inside the String class.
- Checks whether characters inside the Strings are equal.

Ex:

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = new String("Java");  
        String s2 = new String("Java");  
  
        System.out.println(s1.equals(s2)); // true  
    }  
}
```

Q. 5) Explain String concatenation and interning.

String Concatenation in Java

String concatenation means joining two or more Strings into a single String.

```
Ex:-String s1 = "Hello";  
String s2 = "World";  
String result = s1 + " " + s2;  
System.out.println(result);
```

2) String Interning in Java

-String interning means storing only one copy of identical String literals in the String Constant Pool (SCP).

```
-Ex:  
String s1 = new String("Java");  
String s2 = s1.intern();
```

```
String s3 = "Java";
System.out.println(s2 == s3); // true
```

Q. 6) Write a program to find second largest element in an array.

```
import java.util.Scanner;

public class SecondLargest {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of elements:");
        int n = sc.nextInt();

        if (n < 2) {
            System.out.println("Array must have at least 2 elements.");
            return;
        }

        int[] arr = new int[n];
        System.out.println("Enter elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int largest = Integer.MIN_VALUE;
        int secondLargest = Integer.MIN_VALUE;

        for (int i = 0; i < n; i++) {
            if (arr[i] > largest) {
                secondLargest = largest;
                largest = arr[i];
            }
            else if (arr[i] > secondLargest && arr[i] != largest) {
                secondLargest = arr[i];
            }
        }

        if (secondLargest == Integer.MIN_VALUE) {
            System.out.println("No second largest element found.");
        } else {
            System.out.println("Second Largest Element: " + secondLargest);
        }
    }
}
```

Output:-

Enter number of elements:

Enter elements:

1

2

3

4

-1

Second Largest Element: 3

Q. 7) Write a program to sort an array in ascending order.

```
import java.util.Scanner;

public class ArraySort {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of elements:");
        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter elements:");

        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }

        System.out.println("Array in Ascending Order:");
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

Output:-

Enter number of elements:

5

Enter elements:

4

5

32

1

2

Array in Ascending Order:

1 2 4 5 32

Q .8) Write a program to sort an array in descending order.

```
import java.util.Scanner;

public class SortDescending {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of elements:");
        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {

                if (arr[j] < arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}
```

```
        System.out.println("Array in Descending Order:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
```

Output:-

Enter number of elements:

5

Enter elements:

1

2

3

4

5

Array in Descending Order:

5 4 3 2 1