**Problem No.**: 01

**Problem Name:** Multiple inheritance using interface.

What is Multiple inheritance and Interface?
:When there are more than one superclass for a single subclass, that is called multiple inheritance.
Java does not support multiple inheritance with classes.
One must implement interface to implement multiple inheritance.

Why do we need it?
At times it is useful to have separate parent interfaces to abstract functionalities but also needed to inherit from more than one interface.

## Code:

```
package lab16oct;

// Java program to demonstrate Multiple Inheritance
// through default methods

// Interface 1
interface PI1 {

	// Default method
	default void show()
	{

		// Print statement if method is called
		// from interface 1
		System.out.println("Default PI1");
	}
}

// Interface 2
interface PI2 {

	// Default method
	default void show()
	{

		// Print statement if method is called
		// from interface 2
		System.out.println("Default PI2");
	}
```

```
}

// Main class
// Implementation class code
class Lab16Oct implements PI1, PI2 {

	// Overriding default show method
	@Override
	public void show()
	{

		// Using super keyword to call the show
		// method of PI1 interface
		PI1.super.show();//Should not be used directly in the main method;

		// Using super keyword to call the show
		// method of PI2 interface
		PI2.super.show();//Should not be used directly in the main method;
	}

	//Method for only executing the show() of PI1
	public void showOfPI1() {
		PI1.super.show();//Should not be used directly in the main method;
	}

	//Method for only executing the show() of PI2
	public void showOfPI2() {
		PI2.super.show(); //Should not be used directly in the main method;
	}

	// Mai driver method
	public static void main(String args[])
	{

		// Creating object of this class in main() method
		Lab16Oct d = new Lab16Oct();
		d.show();
		System.out.println("Now Executing showOfPI1() showOfPI2()");
		d.showOfPI1();
		d.showOfPI2();
	}

}
```

## Output:

```
run:
Default PI1
Default PI2
Now Executing showOfPI1() showOfPI2()
Default PI1
Default PI2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig 1.1: Output on console.

**Problem No.**: 02

**Problem Name**: Scanner class constructor, Scanner class method.

What are Scanner class constructor, Scanner class method?
:Scanner class constructor is needed to specify the input stream.
Scanner class method is used to specify the format of the input.

Why do we need it?
It is needed to take input from user or file.

## Code:

```
package lab16oct2;

import java.util.Scanner;
class Person{
   Person(String name, int age){
      System.out.println(name);
      System.out.println(age);
   }
}
public class Lab16Oct2 {

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter your name:");
      String name = scanner.nextLine(); //Scanner class method nextLine()
      System.out.println("Enter your age:");
      int age = scanner.nextInt();        ////Scanner class method nextInt()
      Person person = new Person(name, age);
      System.out.println("Enter true or false:");
      boolean yn = scanner.nextBoolean();
      System.out.println(yn);

   }

}
```

## Output:

```
run:
Enter your name:
abc
Enter your age:
23
abc
23
Enter true or false:
true
true
BUILD SUCCESSFUL (total time: 8 seconds)
```

Fig 2.1: Output on console.

**Problem No.**: 03

**Problem Name**: String constructor, string literal vs string objects.

What are String constructor, string literal vs string object?
:String constructor is needed to create string objects.
String literals point to the same location if the content is same.
Whilst, string object creates a new instance in each call regardless of the content.

Why do we need it?
String constructor in needed to create string objects.

**Code:**

```
package lab16oct3;

public class Lab16Oct3 {
   public static void main(String[] args) {
      // Java code to illustrate different constructors and methods
      // String class.

      String s= "GeeksforGeeks";
      String sLit1= "GeeksforGeeks";
      String sLit2= "Geek";
      String sObj= new String ("GeeksforGeeks");
      String sObj1= new String ("GeeksforGeeks");
      String sObj2= new String ("Geek");

      // Returns the number of characters in the String.
      System.out.println("String length = " + s.length());

      // Returns the character at ith index.
      System.out.println("Character at 3rd position = "
                              + s.charAt(3));

      // Return the substring from the ith index character
      // to end of string
      System.out.println("Substring " + s.substring(3));

      // Returns the substring from i to j-1 index.
```

```
System.out.println("Substring = " + s.substring(2,5));

// Concatenates string2 to the end of string1.
String s1 = "Geeks";
String s2 = "forGeeks";
System.out.println("Concatenated string = " +
                                s1.concat(s2));

//proof of immutable string
System.out.println("Immutable: "+s1);

// Returns the index within the string
// of the first occurrence of the specified string.
String s4 = "Learn Share Learn";
System.out.println("Index of Share " +
                        s4.indexOf("Share"));

// Returns the index within the string of the
// first occurrence of the specified string,
// starting at the specified index.
System.out.println("Index of a = " +
                        s4.indexOf('a',3));

// Checking equality of Strings
Boolean out = "Geeks".equals("geeks");
System.out.println("Checking Equality " + out);
out = "Geeks".equals("Geeks");
System.out.println("Checking Equality " + out);

out = "Geeks".equalsIgnoreCase("gEeks ");
System.out.println("Checking Equality " + out);

//If ASCII difference is zero then the two strings are similar
int out1 = s1.compareTo(s2);
System.out.println("the difference between ASCII value is="+out1);
// Converting cases
String word1 = "GeeKyMe";
System.out.println("Changing to lower Case " +
                                word1.toLowerCase());


//proof of immutable string
System.out.println("Immutable: "+word1);
```

```
        // Converting cases
        String word2 = "GeekyME";
        System.out.println("Changing to UPPER Case " +
                                    word2.toUpperCase());

        //proof of immutable string
        System.out.println("Immutable: "+word2);

        // Trimming the word
        String word4 = " Learn Share Learn ";
        System.out.println("Trim the word " + word4.trim());


        //proof of immutable string
        System.out.println("Immutable: "+word4);

        // Replacing characters
        String str1 = "feeksforfeeks";
        System.out.println("Original String " + str1);
        String str2 = "feeksforfeeks".replace('f' ,'g') ;
        System.out.println("Replaced f with g -> " + str2);

        //proof of immutable string
        System.out.println("Immutable: "+str1);

    }

}
```

## Output:

```
run:
String length = 13
Character at 3rd position = k
Substring ksforGeeks
Substring = eks
Concatenated string = GeeksforGeeks
Immutable: Geeks
Index of Share 6
Index of a = 8
Checking Equality false
Checking Equality true
Checking Equality false
the difference between ASCII value is=-31
Changing to lower Case geekyme
Immutable: GeeKyMe
Changing to UPPER Case GEEKYME
Immutable: GeekyME
Trim the word Learn Share Learn
Immutable:  Learn Share Learn
Original String feeksforfeeks
Replaced f with g -> geeksgorgeeks
Immutable: feeksforfeeks
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig 3.1: Output on console.

**Problem No.**: 04

**Problem Name**: Java string operations and immutable string.

What are Java string operations and immutable string?
:Java strings are immutable. Performing operations on it does not change the actual content until rewritten on the same variable

Why do we need it?
We need the string methods to perform operations like getting the length of a string.

**Code:**

```
package lab16oct3;

public class Lab16Oct3 {
   public static void main(String[] args) {
      // Java code to illustrate different constructors and methods
      // String class.

      String s= "GeeksforGeeks";
      String sLit1= "GeeksforGeeks";
      String sLit2= "Geek";
      String sObj= new String ("GeeksforGeeks");
      String sObj1= new String ("GeeksforGeeks");
      String sObj2= new String ("Geek");

      // Returns the number of characters in the String.
      System.out.println("String length = " + s.length());

      // Returns the character at ith index.
      System.out.println("Character at 3rd position = "
                              + s.charAt(3));

      // Return the substring from the ith index character
      // to end of string
      System.out.println("Substring " + s.substring(3));
```

```
// Returns the substring from i to j-1 index.
System.out.println("Substring = " + s.substring(2,5));

// Concatenates string2 to the end of string1.
String s1 = "Geeks";
String s2 = "forGeeks";
System.out.println("Concatenated string = " +
                                s1.concat(s2));

//proof of immutable string
System.out.println("Immutable: "+s1);

// Returns the index within the string
// of the first occurrence of the specified string.
String s4 = "Learn Share Learn";
System.out.println("Index of Share " +
                        s4.indexOf("Share"));

// Returns the index within the string of the
// first occurrence of the specified string,
// starting at the specified index.
System.out.println("Index of a = " +
                        s4.indexOf('a',3));

// Checking equality of Strings
Boolean out = "Geeks".equals("geeks");
System.out.println("Checking Equality " + out);
out = "Geeks".equals("Geeks");
System.out.println("Checking Equality " + out);

out = "Geeks".equalsIgnoreCase("gEeks ");
System.out.println("Checking Equality " + out);

//If ASCII difference is zero then the two strings are similar
int out1 = s1.compareTo(s2);
System.out.println("the difference between ASCII value is="+out1);
// Converting cases
String word1 = "GeeKyMe";
System.out.println("Changing to lower Case " +
                                word1.toLowerCase());


//proof of immutable string
```

```
        System.out.println("Immutable: "+word1);

        // Converting cases
        String word2 = "GeekyME";
        System.out.println("Changing to UPPER Case " +
                                    word2.toUpperCase());

        //proof of immutable string
        System.out.println("Immutable: "+word2);

        // Trimming the word
        String word4 = " Learn Share Learn ";
        System.out.println("Trim the word " + word4.trim());


        //proof of immutable string
        System.out.println("Immutable: "+word4);

        // Replacing characters
        String str1 = "feeksforfeeks";
        System.out.println("Original String " + str1);
        String str2 = "feeksforfeeks".replace('f' ,'g') ;
        System.out.println("Replaced f with g -> " + str2);

        //proof of immutable string
        System.out.println("Immutable: "+str1);

    }

}
```

## Output:

```
run:
String length = 13
Character at 3rd position = k
Substring ksforGeeks
Substring = eks
Concatenated string = GeeksforGeeks
Immutable: Geeks
Index of Share 6
Index of a = 8
Checking Equality false
Checking Equality true
Checking Equality false
the difference between ASCII value is=-31
Changing to lower Case geekyme
Immutable: GeeKyMe
Changing to UPPER Case GEEKYME
Immutable: GeekyME
Trim the word Learn Share Learn
Immutable:  Learn Share Learn
Original String feeksforfeeks
Replaced f with g -> geeksgorgeeks
Immutable: feeksforfeeks
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig 4.1: Output on console.

**Problem No.**: 05

**Problem Name**: Equality test and equals() method.

**Code:**

```
package lab16oct4;

public class Lab16Oct4 {

    public static void main(String[] args) {
      // Java program for using == operator


        // Declaring reference value
        int[] a = { 1, 2, 3, 4 };
        int[] b = { 1, 2, 3, 4 };
        int[] c = b;


        String s1 = "GG";
        String s2 = "GG";
        String s3 = "GJ";
        String sO1 = new String("GG");
        String sO2 = new String("GG");
        String sO3 = new String("GJ");

        // Comparing a and b using == operator
        // Though they both have the same value
        // the output will be false because
        // they both have a different address in the memory
        System.out.println("Are a and b equal? "
                                  + (a == b));

        // Comparing b and c using == operator
        // Though they both have the same value
        // the output will be true because
        // they both have same address in the memory
        System.out.println("Are b and c equal? "
                                  + (b == c));
```

```
        System.out.println("Are s1 & s2 equal?"+(s1==s2));
        System.out.println("Are s1 & s2 equal?"+(s1.equals(s2)));

        System.out.println("Are s1 & s3 equal?"+(s1==s3));
        System.out.println("Are s1 & s3 equal?"+(s1.equals(s3)));

        System.out.println("Are sO1 & sO2 equal?"+(sO1==sO2));
        System.out.println("Are sO1 & sO2 equal?"+(sO1.equals(sO2)));

        System.out.println("Are sO1 & sO3 equal?"+(sO1==sO3));
        System.out.println("Are sO1 & sO3 equal?"+(sO1.equals(sO3)));


    }

}
```

## Output:

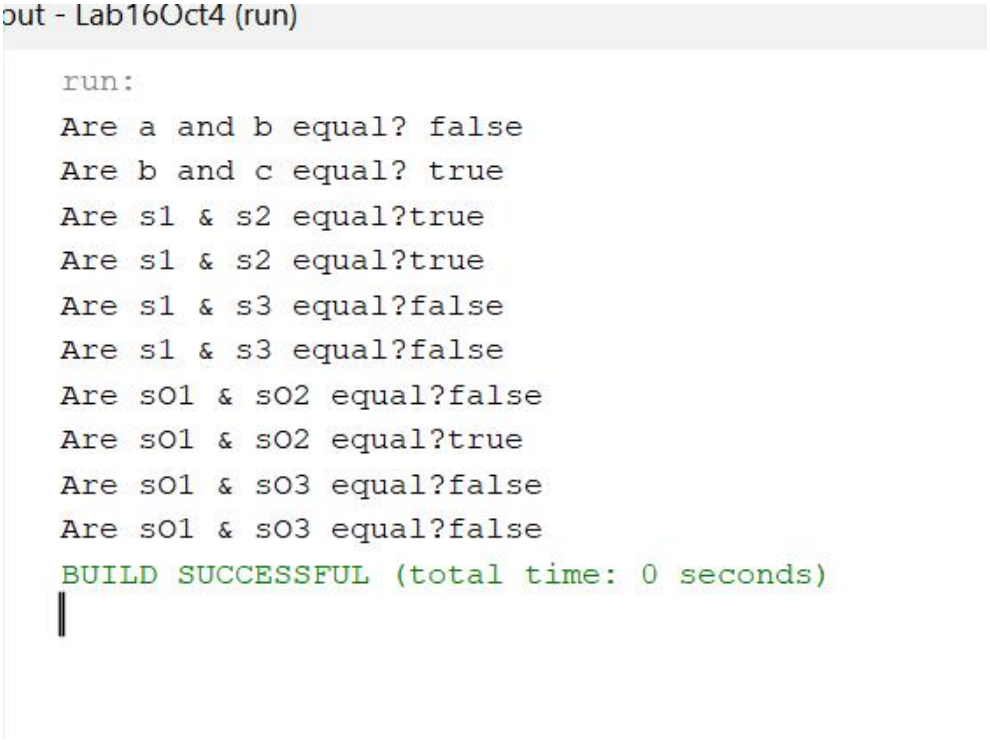


Fig 5.1: Output on console.

**Problem No.**: 06

**Problem Name**: String buffer.

**Code:**

```
package lab16oct5;

public class Lab16Oct5 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String sa = new String("ABC");
        //sa.append("Hello");  //throws an error
        StringBuffer sb = new StringBuffer();
        sb.append("Hello");
        sb.append(" ");
        sb.append("world");
        String message = sb.toString();
        System.out.println(message);
        System.out.println(sb);


    }

}
```

## Output:

```
run:
Hello world
Hello world
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig 6.1: Output on console.