I - ASSIGNMENT
(Start Writing From Here)

1. Demonstrate the String class methods with suitable examples.

The Java String class provides a lot of built-in methods that are used to manipulate string in Java.

String Class Methods:-

1. Length - to find the length of the string

Ex: String myString = "Hello, World!";
    int length = myString.length();
    System.out.println("Length of the String: "+ length);
    o/p:- 13

2. Concatenation: to concatenate or combine strings:

Ex String firstString = "Hello, ";
   String secondString = "World!";
   String combinedString = firstString.concat(secondString);
   System.out.println("Combined string: "+ combinedString);
   O/p: Hello, World!

3. Substring: to extract a part of a string:

Ex String longString = "This is a long string";
   String subString = longString.substring (5,10);
   System.out.println("Substring: "+ substring);
   o/p: is a

4. Replace: to replace characters or strings within a string:

Ex String originalString = "I like cats.";
   String newString = originalString.replace("cats", "dogs");
   System.out.println("New string:" +newString);
   o/p:- I like dogs.

5. UpperCase/LowerCase - to convert the case of the string

Ex: String mixedCase = "Hello, World!";

```
String lowerCase = mixedCase.toLowerCase();
String upperCase = mixedCase.toUpperCase();
System.out.println("Lowercase: " + lowerCase);   o/p: hello, world!
System.out.println("Upper case: "+ UpperCase);   o/p: HELLO, WORLD!
```

6. CharAt :- method returns the character at the specified index in a string.

Ex: String myStr = "Hello";

```
char result = myStr.charAt(0);
System.out.println(result);        o/p: H
```

7. Trim :- method removes whitespace from both ends of a string

Ex String myStr = "   Hello World!   ";

```
System.out.println(myStr);
System.out.println(myStr.trim());
```

o/p:

```
            Hello World!
     Hello World!
```

8. ValueOf( ) :- return the string representation of the specified value

Ex: int a = 10;

```
String s = String.valueOf(a);
System.out.println(s+10);
```

o/p :- 1010

2 Contrast: Method overriding Vs. Method overloading. Which is necessary for runtime polymorphism? Justify.

Method Overloading	Method Overriding
1 Method overloading is a compile time polymorphism.	1. Method overriding is a run time polymorphism
2. Method overloading helps to increase the readability of the program	2. Method overriding is used to grant the specific implementation of the method which is already provided by it's parent class or superclass.
it occurs within the class	it is performed in two classes with inheritance relationships.
Method overloading may or may not require inheritance	Method overriding always needs inheritance
In method overloading, methods must have the same name and different signatures	In method overriding, methods must have the same name and same signature
In method overloading, the return type can or can not be the same. but we just have to change the parameter	In method overriding, the return type must be the same or co-variant.
Static binding is being used for overloaded methods.	Dynamic binding is being used for overriding methods.

Poor performance due to compile time polymorphism.	it gives better performance. The reason behind this is that the binding of overhidden methods is being done at runtime.
Private and final methods can be overloaded	Private and final methods can't be ~~overloaded~~ ridden.
The argument list should be different while doing method overloading	The argument list should be the same in method overriding.

Method Overloading in Java.

Method Overloading is a compile time polymorphism. In method, overloading, more than one method shares the same method name with a different signature in the class. In method overloading, the return type can or can not be the same, but we have to change the parameter because, in java, we can not achieve method overloa-ding by changing only the return type of the method.

Method Overriding in Java:

Method Overriding is a runtime polymorphism. In method overriding, the derived class provides the specific implementation of the method that is already provided by the base class or parent class. In method overriding, the return type must be the same or co-variant (return type may vary in the same direction as the derived class)

3. Write a Java program to demonstrate the use of Interfaces in achieving multiple inheritances.

```
interface Printable
{
   void print(); }
   Interface showable {
    void show();
   }
   Class Multiple_Inheritance implements Printable, showable {
   public void print()
   {
     System.out.println("Hello");
    }
    public void show()
    {
     System.out.println("Welcome");
    }
    Public static void main(String args[])
    {
    Multiple_Inheritance obj = new Multiple_Inheritance();
    Obj.print();
    obj.show();
     }
 }
```

Output:

```
Hello
Welcome.
```

4. Explain various methods in Object class.

Object class in Java

This class is present in java.lang package. It's direct or indirectly used to be derived in each java class. If any class extended any other class then it will be indirectly derived else direct.

Methods in Object Class:-

1. toString():- It's provide string representation or convert object to string form. you can overide toString() method to get your own String representation of objects.
2. hashCode(): It's generate unique hashcode for each object. The main advantage of saving objects. It's used to override for user defined objects for better perfomance like searching
3. equals(Object obj):- It's used to compare the two objects dynamically
4. getClass():- It return runtime class object and used to get metdata information as well.
5. finalize(): This method call required to perform garbage collector
6. Clone(): It used to create the copy or clone of object
7. notify(), notifyAll() and wait():- Used for concurrent programm-ing to manage thread synchonization.