**Index**

1. Method and Constructor

- Introduction (Page no- 2)

- Basic program in eclipse (Page no- 2)

- Some basic questions (Page no- 3)

1. Inheritance

- Introduction (Page no- 3)

- Basic program in eclipse (Page no- 3)

- Interface (Page no- 6)

- Super Keyword (Page no- 8)

1. Polymorphism

- Introduction (Page no- 9)

- Basic program in eclipse (Page no- 9)

1. String Handling

- Introduction (Page no- 10)

- Basic program in eclipse (Page no- 11)

1. References (Page no- 12)

**1. Method and Constructor**

*Introduction*

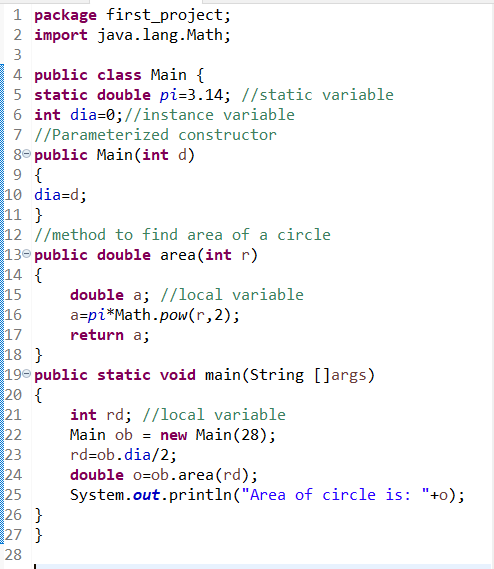
A method is a code block that contains a series of statements. There are three steps of making methods in a class:-

1. Method declaration (optional step)
2. Method definition
3. Method Calling

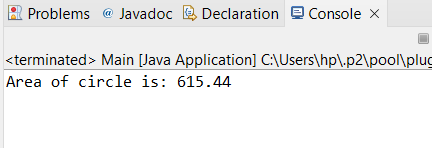
A constructor initializes an object when it is created. Have no explicit return type. Has the same name as its class. There are three types of constructors: -

1. Default
2. No-arg
3. Parameterized

*Basic program in java using method and constructor*



Output: -



*Some Basic Questions*

*What is the difference between main method and other methods inside a class or package?*

The Main method is the **entry point** for every java application and it is **called by the JVM** when the program is started.

*What are local, automatic, temporary and stack variables?*

All of these variables are one and the same thing. Variables that are defined inside a method are called local, automatic, temporary and stack variables.

*Difference between local, static and instance variables?*

**Local variables** require explicit (by user) initialization. Local variables are declared in a method, constructor, or block. Local variables are not visible outside the method. **Created** when method or constructor is entered and **Destroyed** on exit.

**Instance/member/field variables** are initialized automatically. They are **declared in a class**, but **outside a method.** An instance variable is created when an object is created and destroyed when the object is destroyed. Visible in all methods and constructors of the defining class, should generally be declared private, but may be given greater visibility. **Created** when instance of class is created with new. **Destroyed** when there are no more references to enclosing object (made available for garbage collection). often prefixed to clarify difference from local variables, Eg with my, m, or m\_ (for member) as in myLength, or this as in this.length.

**Class/static variables** are declared with the static keyword in a class, but outside a method. Must be declared static. Typically also final. **Created** when the program starts. **Destroyed** when the program stops. Same as instance variable, but are often declared public to make constants available to users of the class. static public final variables (constants) are all uppercase, otherwise normal naming conventions. Alternatively prefix the variable with "c\_" (for class) or something similar. here is only one copy per class, regardless of how many objects are created from it. They are stored in static memory. It is rare to use static variables other than declared final and used as either public or private constants.

*What is the meaning of word static?*

In the Java programming language, the keyword static indicates that the particular member belongs to a type itself, rather than to an instance of that type. This means that only one instance of that static member is created which is shared across all instances of the class.

*What is Pass by value and Pass by reference?*

Pass by value – when a value type is passed to a method, a copy is passed instead of the object itself.

Pass by reference – when an object of a reference type is passed to a method, a reference to the object is passed.

*What is this keyword?*

The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name.

**2. Inheritance**

*Introduction*

One class can derive the properties of another class.

Keyword: - extends

Eg:- Car extends Vehicle

Supported inheritance – Single, Multilevel, Hierarchical, Multiple (through interface) and Hybrid inheritance (through interface)

**Single Inheritance –** when class extends only single or one class.

**Multilevel Inheritance –** Derived class acts as a parent to another class.

**Hierarchical Inheritance –** One parent class and multiple subclasses

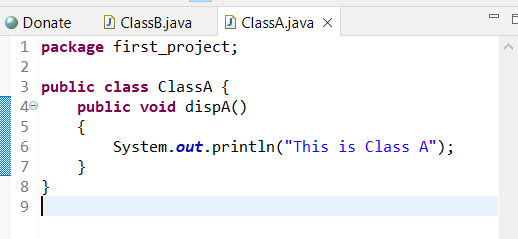
**Multiple Inheritance -** when class extends multiple classes. Java does not support multiple inheritance as child has to manage the dependency of multiple parents on it.

**Hybrid Inheritance –** combination of single and Multiple inheritance

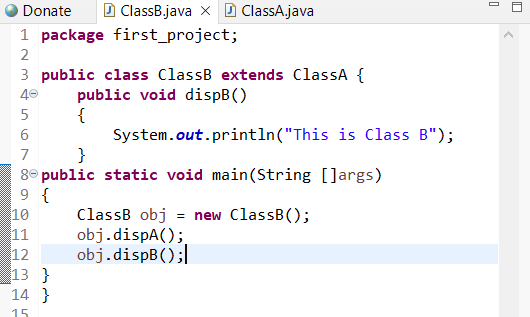
*Basic program in java for inheritance*

1. Single Inheritance

ClassA



ClassB

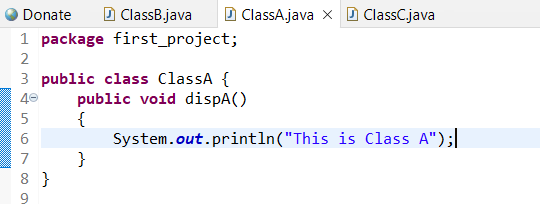


Output

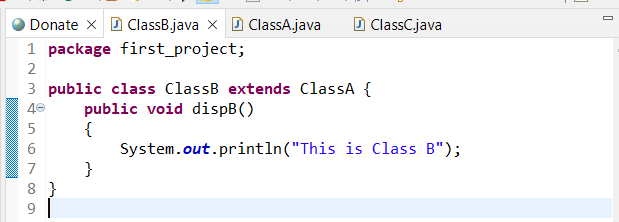


1. Multilevel Inheritance

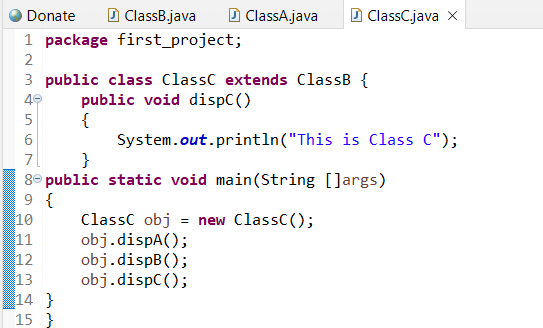
ClassA



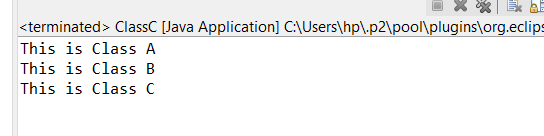
ClassB



ClassC

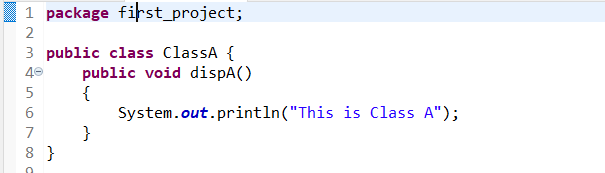


Output

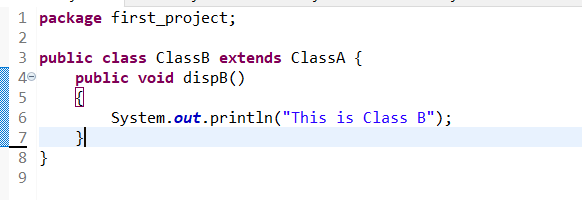


1. Hierarchical Inheritance

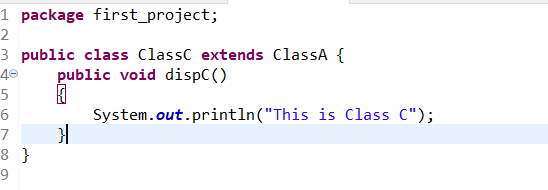
ClassA



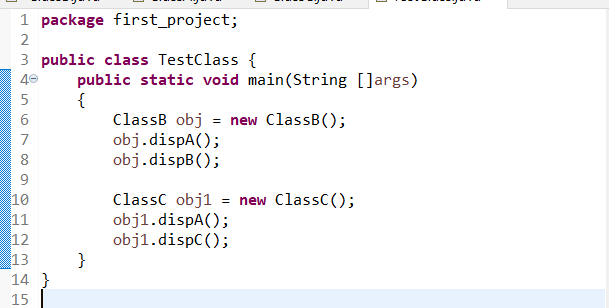
ClassB



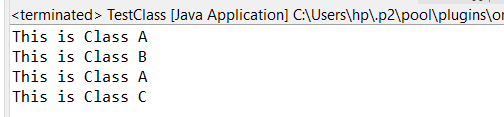
ClassC



TestClass



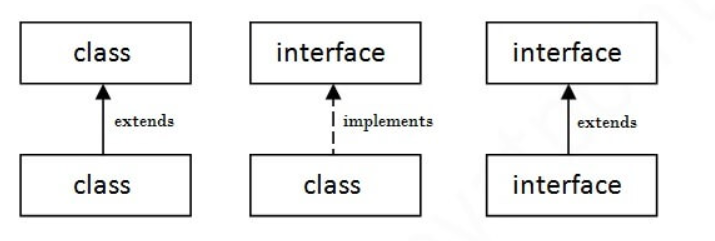
Output

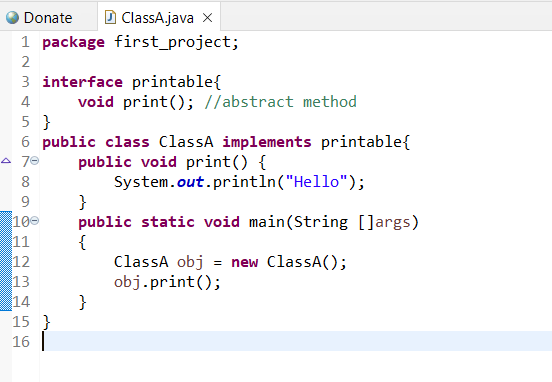


*Interface: - Keyword implements*

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this: abstract void moveTo(double deltaX, double deltaY);

An interface in Java is a **blueprint of a class**. It has static constants and abstract methods.The interface in Java is a mechanism to achieve abstraction.

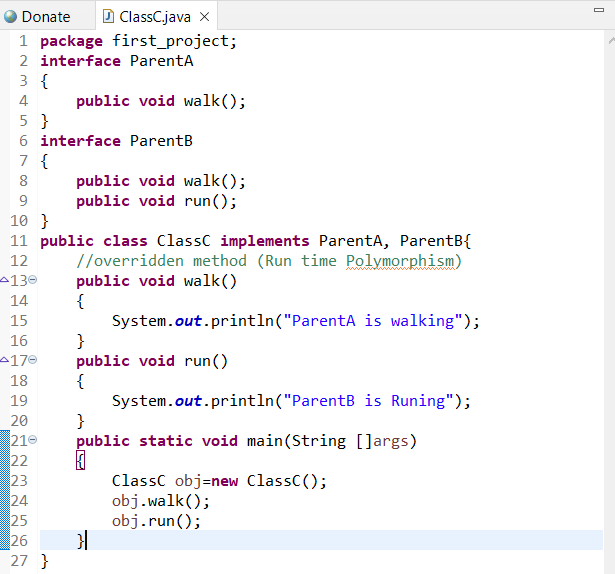




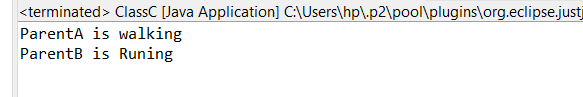
Output



1. Multiple Inheritance

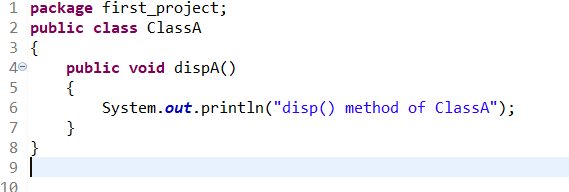


Output-

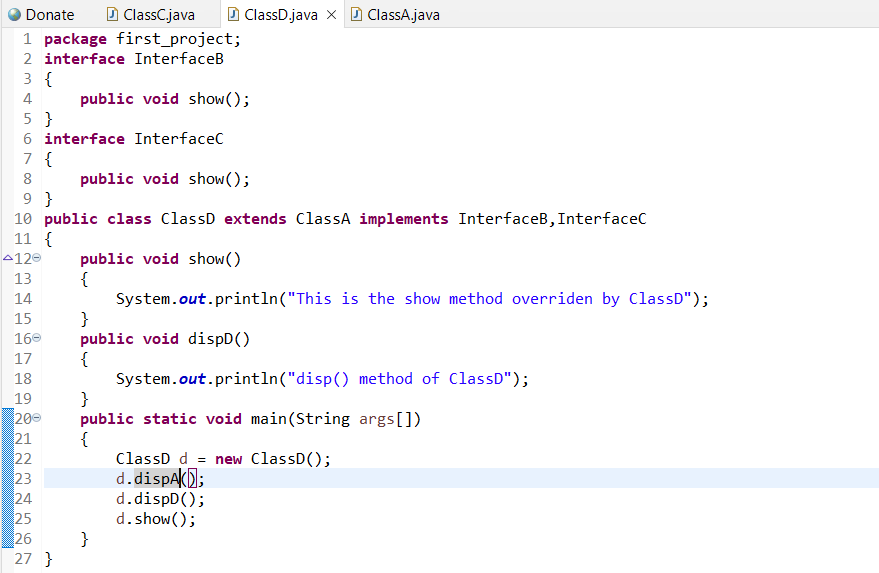


1. Hybrid Inheritance

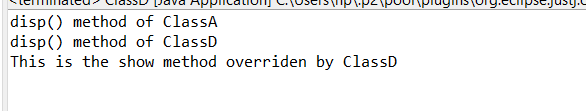
ClassA



Class D with interface B and C



Output –

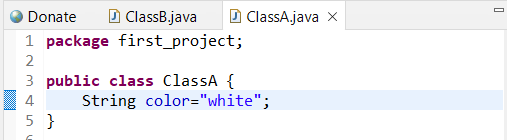


*Super keyword*

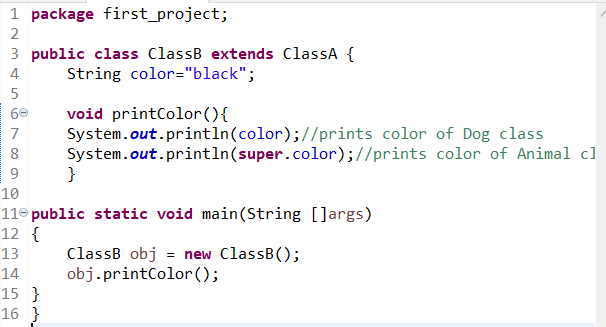
1. Invoke parent class method
2. Invoke parent class constructor
3. Refer to parent class instance variable

*Basic program in java for super keyword*

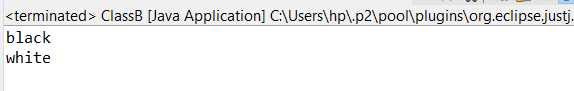
ClassA



ClassB



Output –



*Polymorphism*

It means multiple forms.

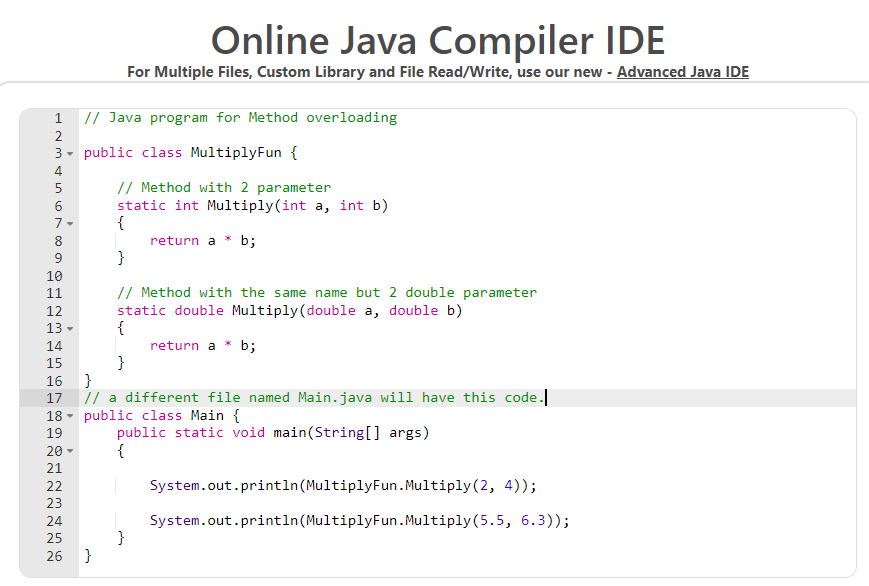
For example: - A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So, the same person possesses different behavior in different situations. This is called polymorphism.

In JAVA: - Polymorphism allows us to define one interface and have multiple implementations.

***Types of polymorphism***

1. **Compile time polymorphism** – Achieved by function overloading.

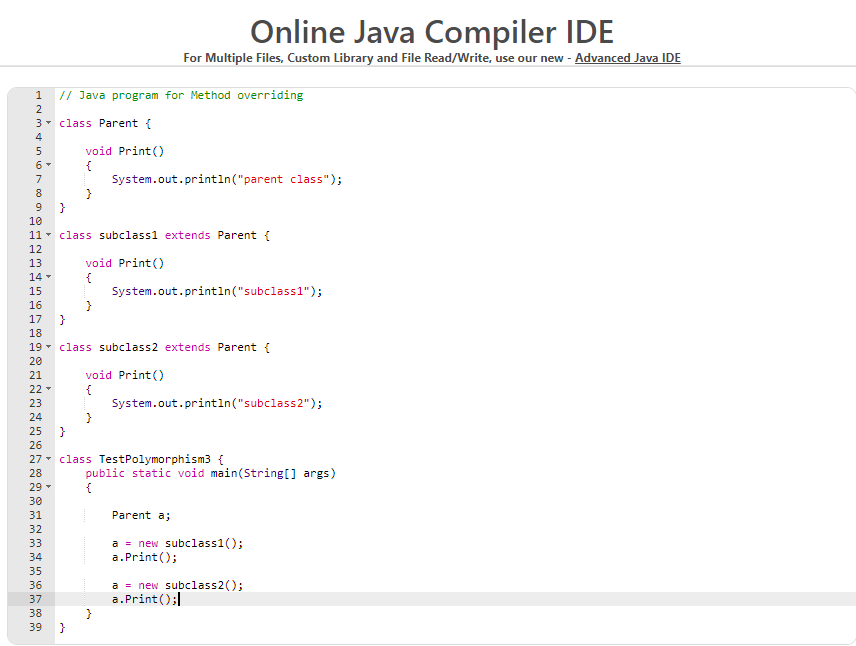
When there are multiple functions with same name but different parameters then these functions are said to be overloaded.



Output –



1. **Runtime polymorphism –** Method overriding. It occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden. In this inheritance is required and name of two to three functions are same. Child class redefines the parent class method known as overriding.



Output –



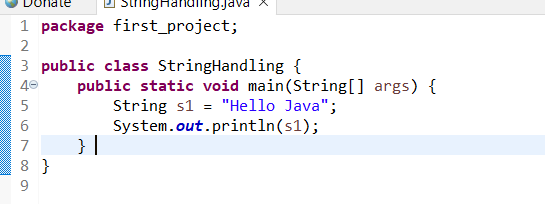
*String Handling*

String is an object that represents sequence of characters. In Java, String is represented by String class which is located into java.lang package.

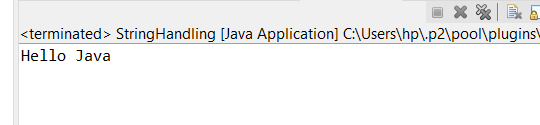
Notes: -

* String objects are immutable that means once a string object is created it cannot be changed.

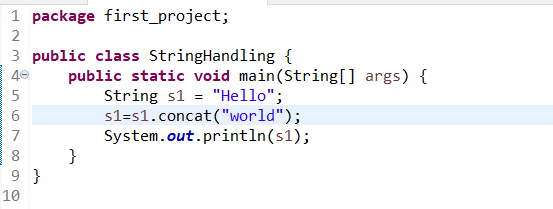
Program 1



Output –



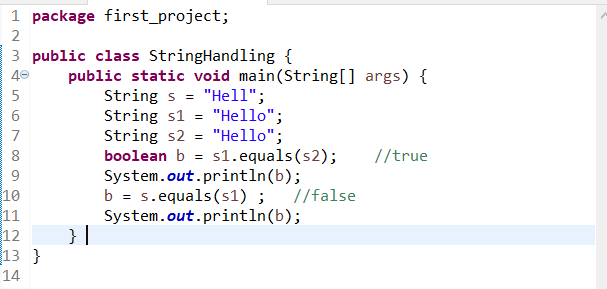
Program 2



Output –



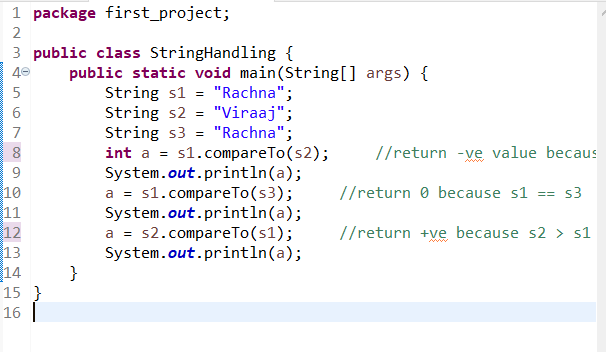
Program 3



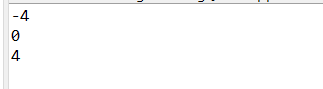
Output –



Program 4

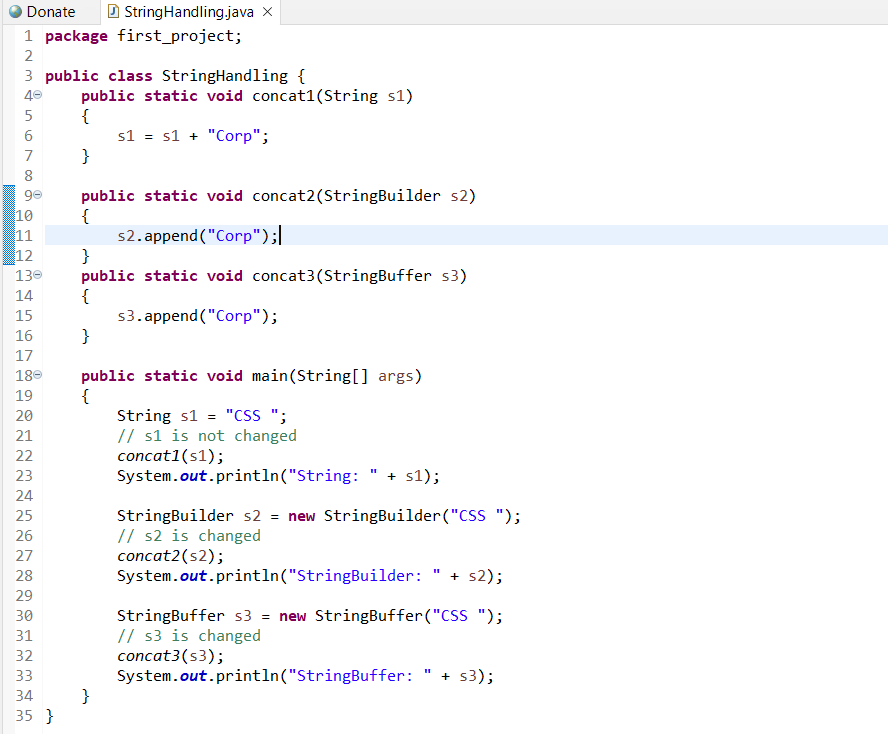


Output –

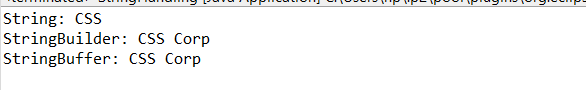


*String vs StringBuilder vs StringBuffer in Java*

Code: -



Output: -



Explanation:

Concat1: In this method, we pass a string “CSS” and perform “s1 = s1 + ”Corp”. The string passed from main() is not changed, this is due to the fact that String is immutable. Altering the value of string creates another object and s1 in concat1() stores reference of the new string. References s1 in main() and cocat1() refer to different strings.

Concat2: In this method, we pass a string “CSS” and perform “s2.append(“Corp”)” which changes the actual value of the string (in main) to “CSS Corp”. This is due to the simple fact that StringBuilder is mutable and hence changes its value.

Concat3: StringBuffer is similar to StringBuilder except for one difference that StringBuffer is **thread-safe**, i.e., **multiple threads can use it without any issue.** The thread-safety brings a penalty of performance.

References:-

1. <https://www.studytonight.com/java/string-handling-in-java.php>
2. <https://www.tutorialspoint.com/multiple-inheritance-by-interface-in-java>
3. <https://drive.google.com/drive/folders/1fWX5Oon59vpI_HFl8AkM2oYEBOb-6mKr>
4. <https://docs.oracle.com/javase/8/docs/api/>
5. <https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/>