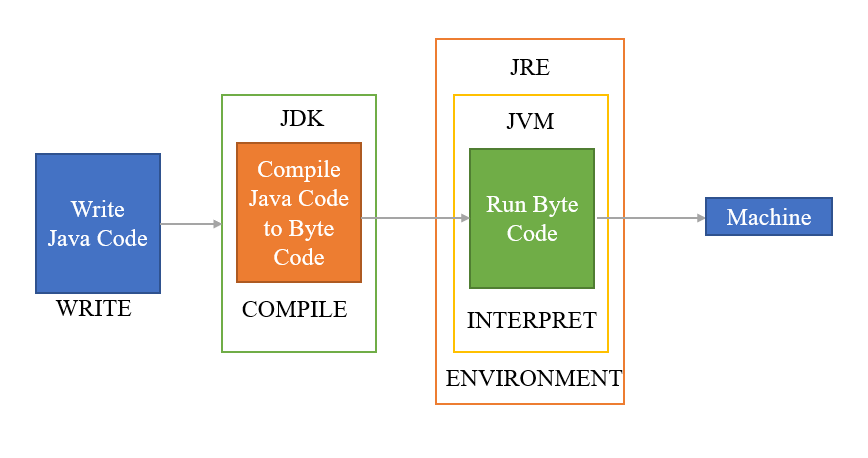
**Core Java Index-version1**

**Table Of Contents**

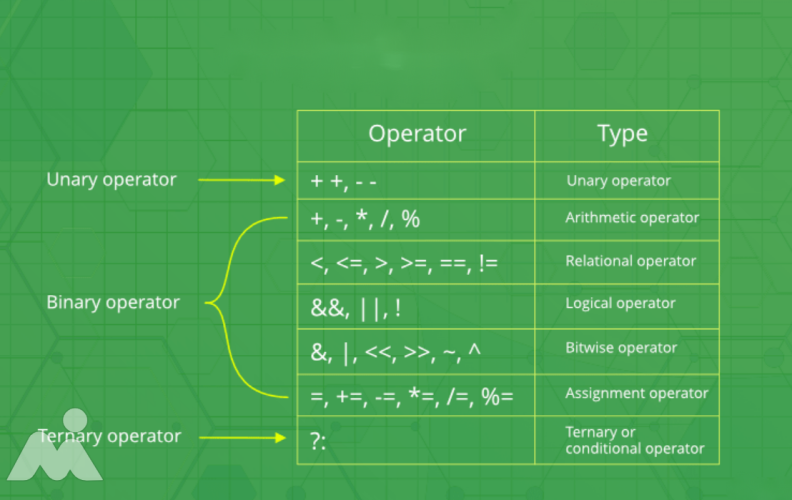
|  |  |
| --- | --- |
| **Table of Contents** | **Page No** |
| **Different Type’s of Language’s** | **1** |
| 1. **Introduction to Java** |
| 1. **Java Feature’s** |
| 1. **JVM,JDK,JRE** |
| 1. **JDK Version’s** |
| 1. **operator’s** |
| 1. **Condition Statement’s** | **2** |
| 1. **Iteration Statement’s** |
| 1. **Jump Statement’s** |
| 1. **Data Type’s** |
| 1. **Nested Loop’s** | **3** |
| 1. **Array’s** |
| 1. **Package’s** | **4** |
| 1. **Switch Case Statement’s** | **5** |
| 1. **enum** |
| 1. **Scanner Class** | **6** |
| 1. **OOP’s Concepts** | **6-9** |
| 1. **Constructor’s** | **10-12** |
| 1. **static keyword** | **12-14** |
| 1. **final keyword** | **15-16** |
| 1. **String’s** | **16-18** |
| 1. **StringBuffer** | **18** |
| 1. **StringBuilder** | **19** |
| 1. **Difference between String,StringBuffer and StringBuilder** |
| 1. **String Class Performance** | **20** |
| 1. **Creating our Immutable class** | **21-22** |
| 1. **Interfaces-In Depth** | **22-25** |
| 1. **Abstract Classes and Methods** | **25-26** |
| 1. **Exceptional Handling** |  |

Different Language's Like C,C++, .net, Java there Intro and Differences between them **5-8-24 Day-1**

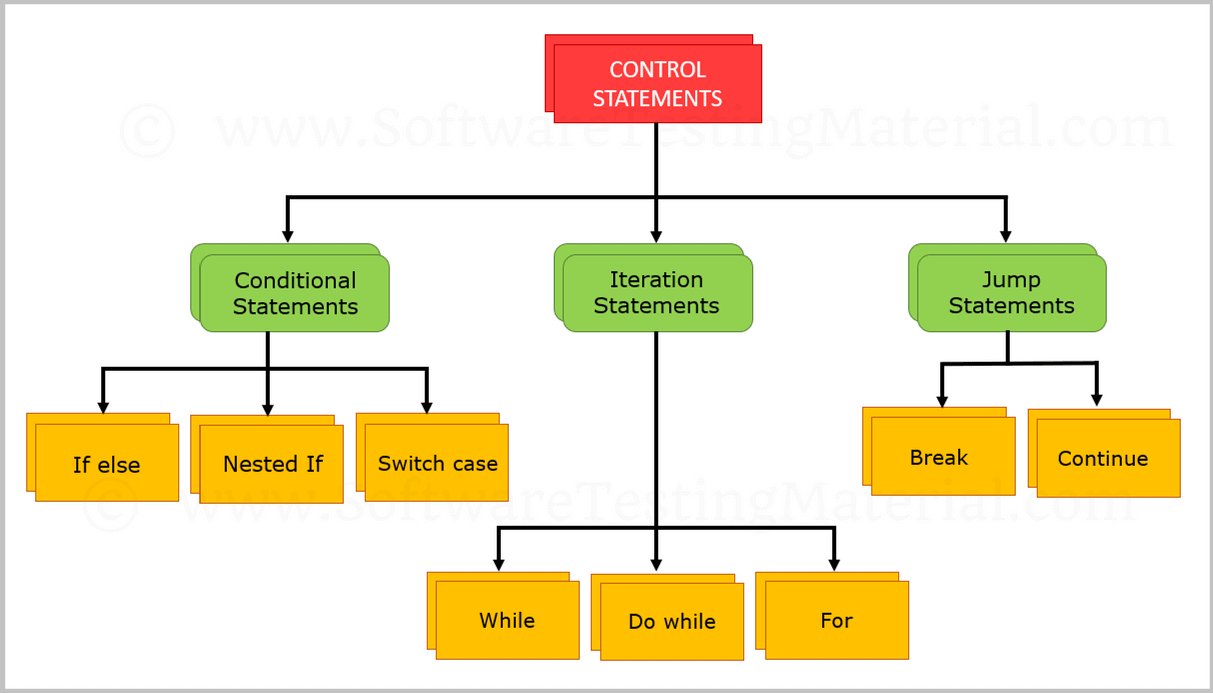
1. **Introduction to Java’**
2. **Java Feature's**
   1. Platform Independent
   2. Open Source
   3. OOPS Concept
   4. Garbage Collection(Memory Management)
   5. Exception Handling
   6. Multi-Threading
   7. Secure(Security)
3. **JDK,JRE,JVM**
   1. Difference Between JRE and JVM



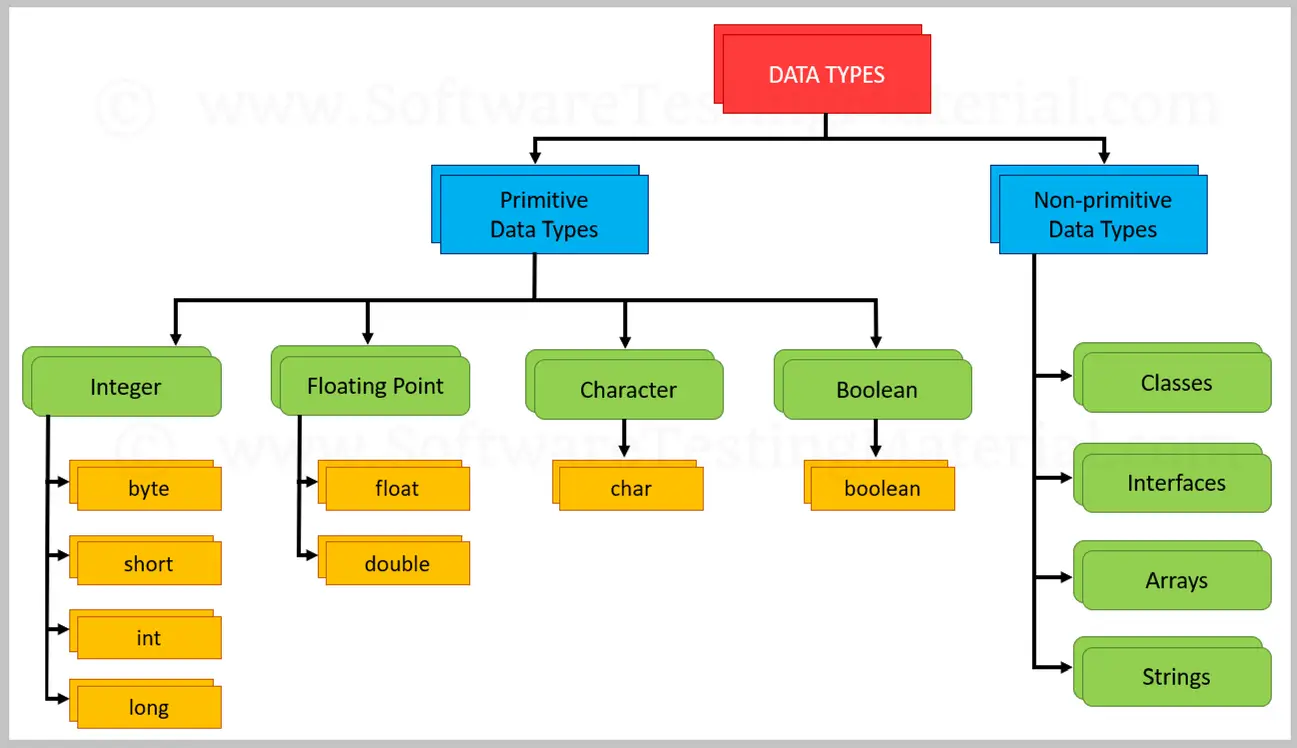
1. **JDK Version’s**
   1. 1.0 Release date 23rd January 1996
   2. 1.1 Release date 18th February 1997
   3. 1.5 or 5(StringBuilder,for-each loop,enum,Annotation)
   4. 1.8 or 8
2. **Operator's**
   1. Arithmetic Operator’s(**+,-,\*,%,/**)
   2. Relational Operator’s**(<,>,<=,>=,==,!=)**
   3. Logical Operator’s**( &&, ||,!)**



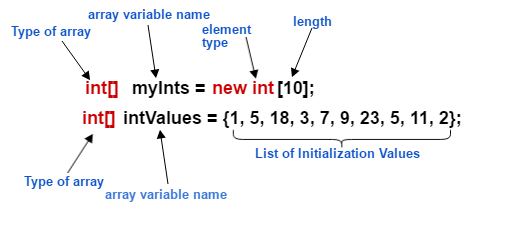
1. **Condition Statements** 
   1. If
   2. If else
   3. else if
   4. Nested if
   5. Switch case
2. **Iteration Statements**
   1. While
   2. Do while
   3. For
3. **Jump Statements**
   1. Continue
   2. Break



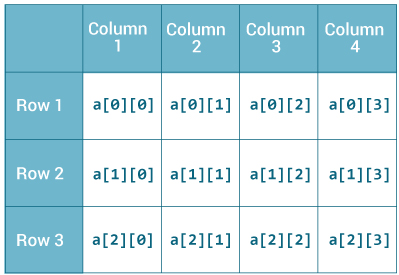
1. **Data Type’s**
   1. Primitive
      1. byte,Short,Int ,long,float,double
   2. Non-Primitive
      1. Arrays
      2. Strings
      3. Interface’s
      4. enums..etc



1. **Nested Loop’s**  **5-8-24 Day-2**
   1. For loop’s
2. **Array’s**
   1. Single Dimensional Array’s



* 1. 2D Array’s



**Example:-**

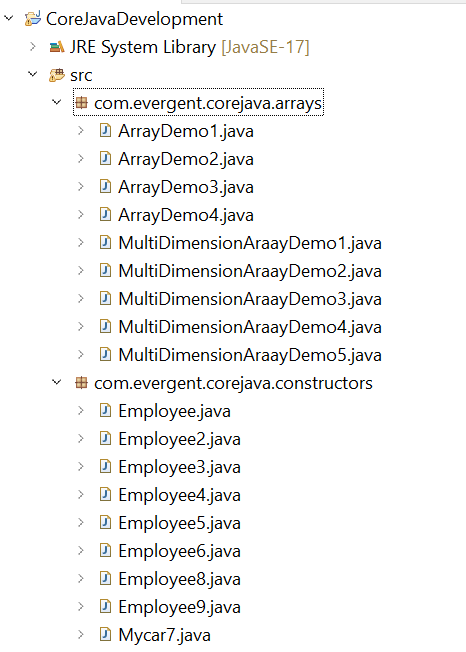




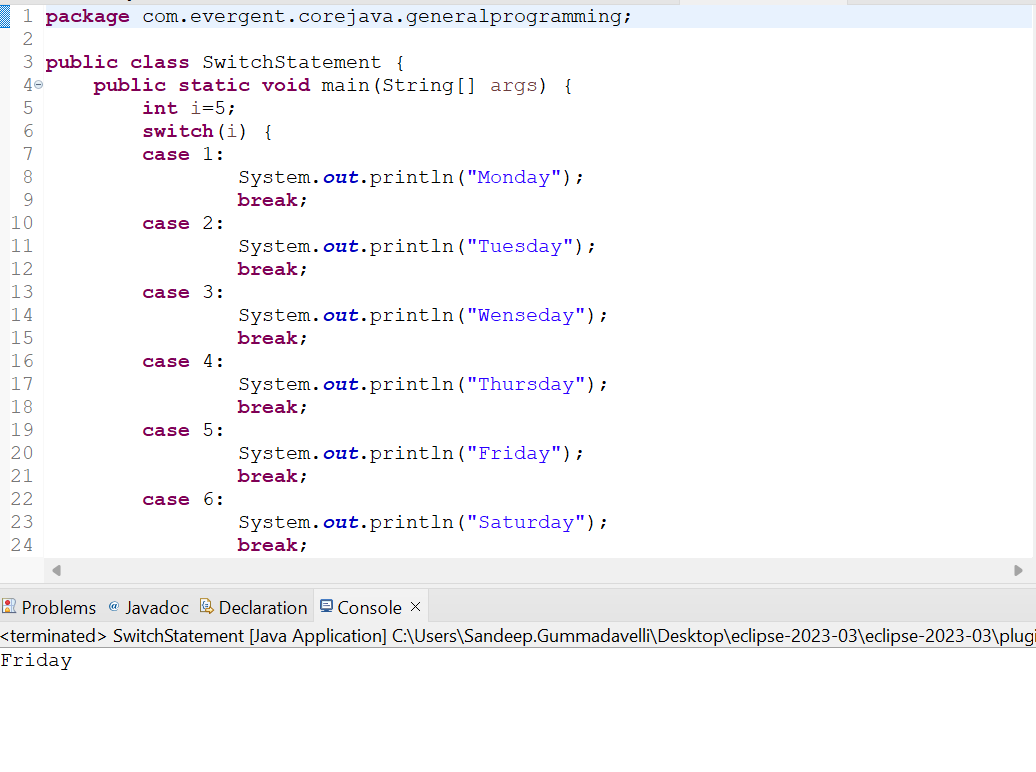
1. **Package’s :-**Packages contains classes and interfaces in java **6-8-24 Day-3**
   1. Pre-defined
2. Java.lang;

* System.stream
* String Class
* StringBuffer

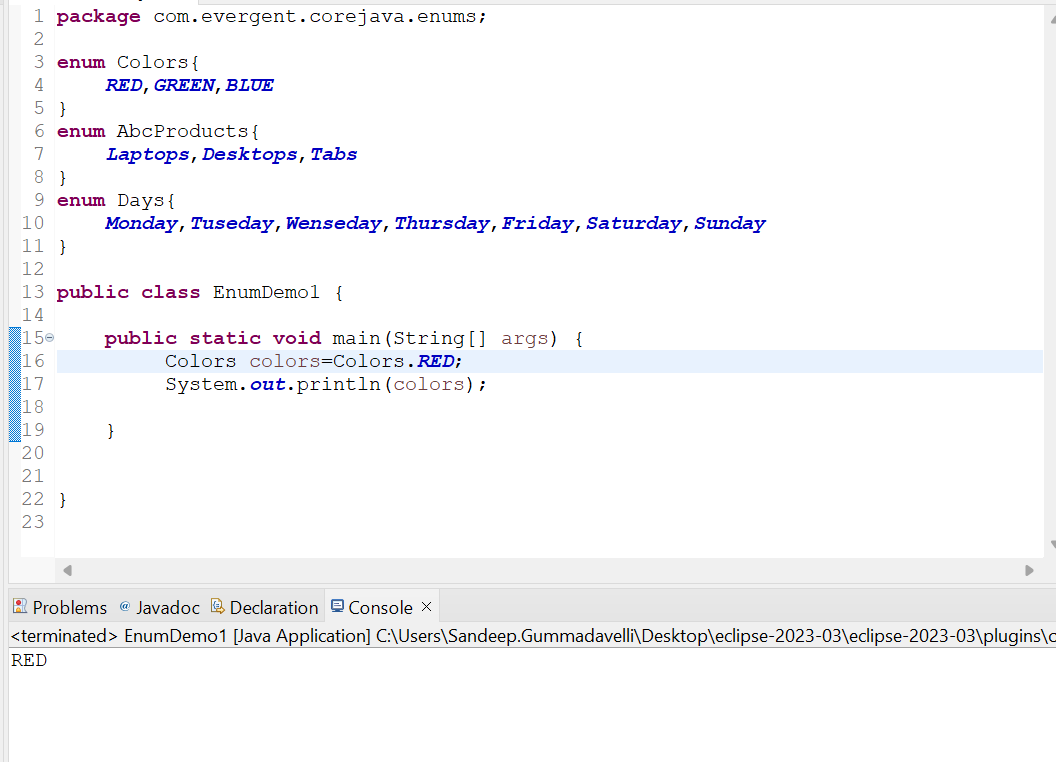
1. StringBuilder
2. Java.util;
3. Java.sql;
   1. User-defined
      1. The User-defined packages are the those packages which are created by the user explicitly in order to maintain Structure of the Application by placing the similar class into one package
      2. Example:-com.evergent.corejava.arrays;



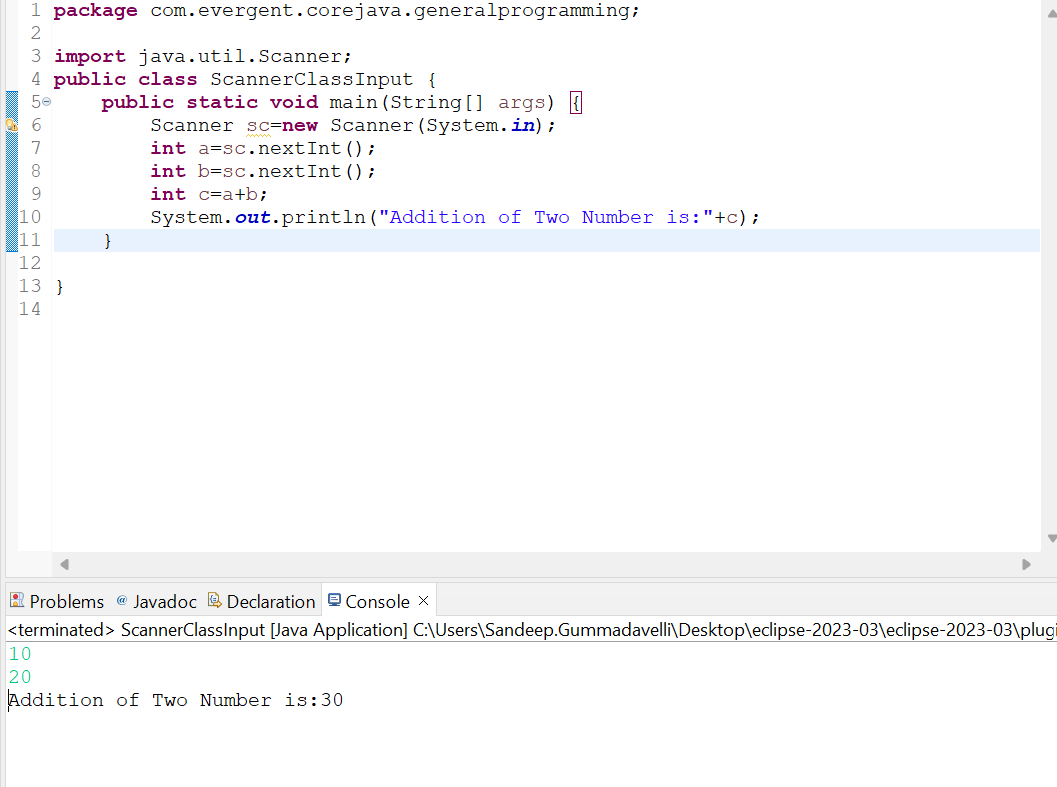
1. **Switch Case Statement:-**Switch statement in java is a muti-way branch statement.



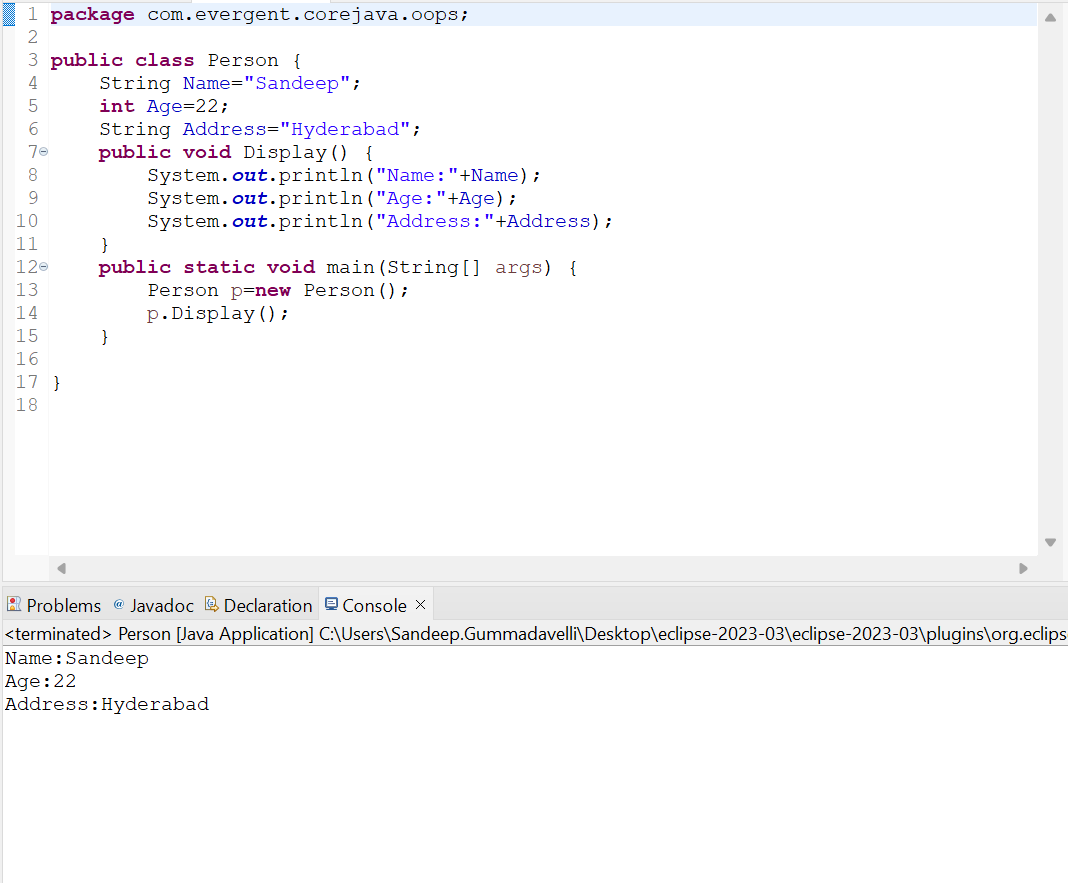
1. **Enum:-**enum is a special class that represents the group of constants



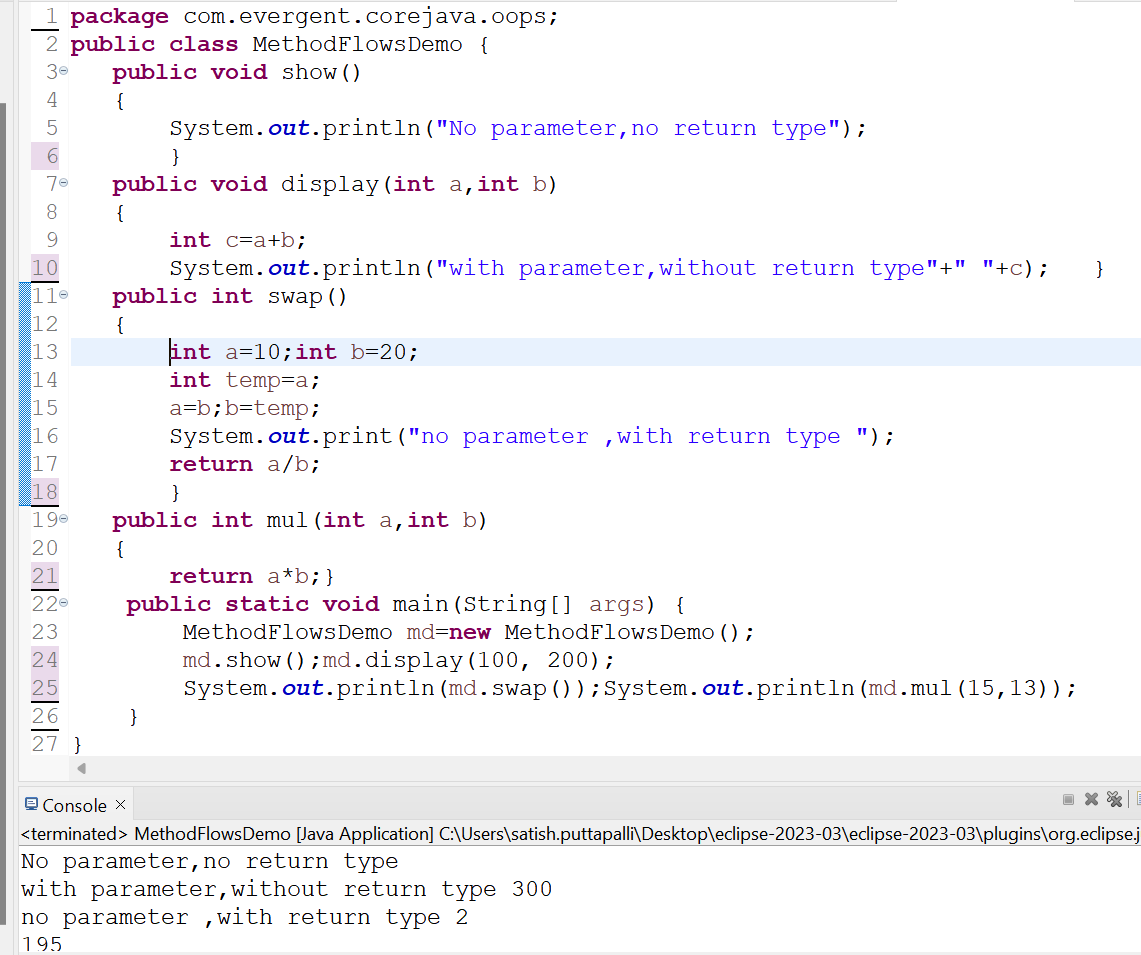
1. **Scanner Class:-**Scanner Class in java is used to take the input values from the keyboard

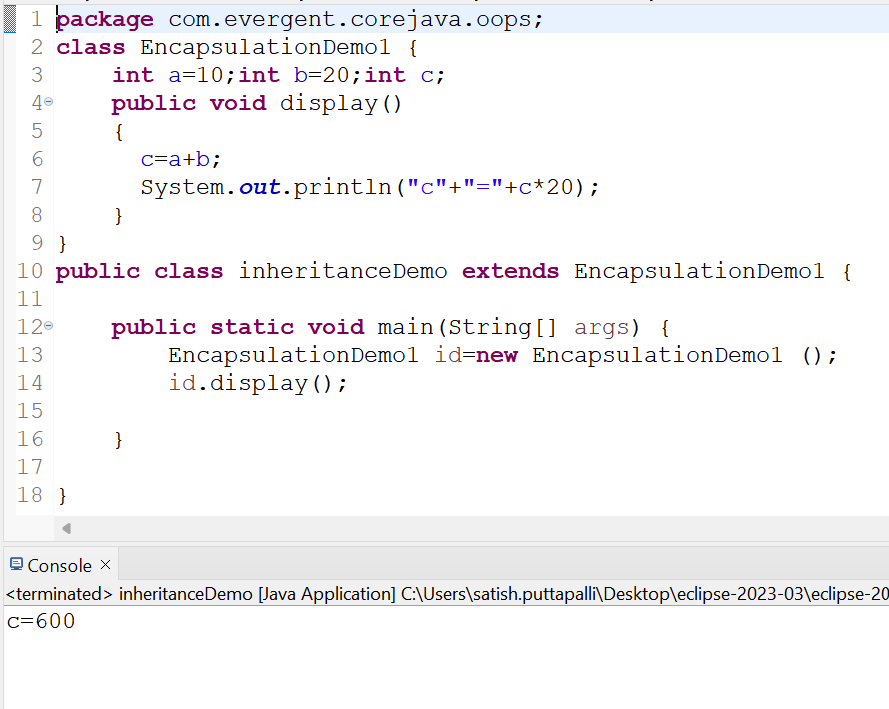


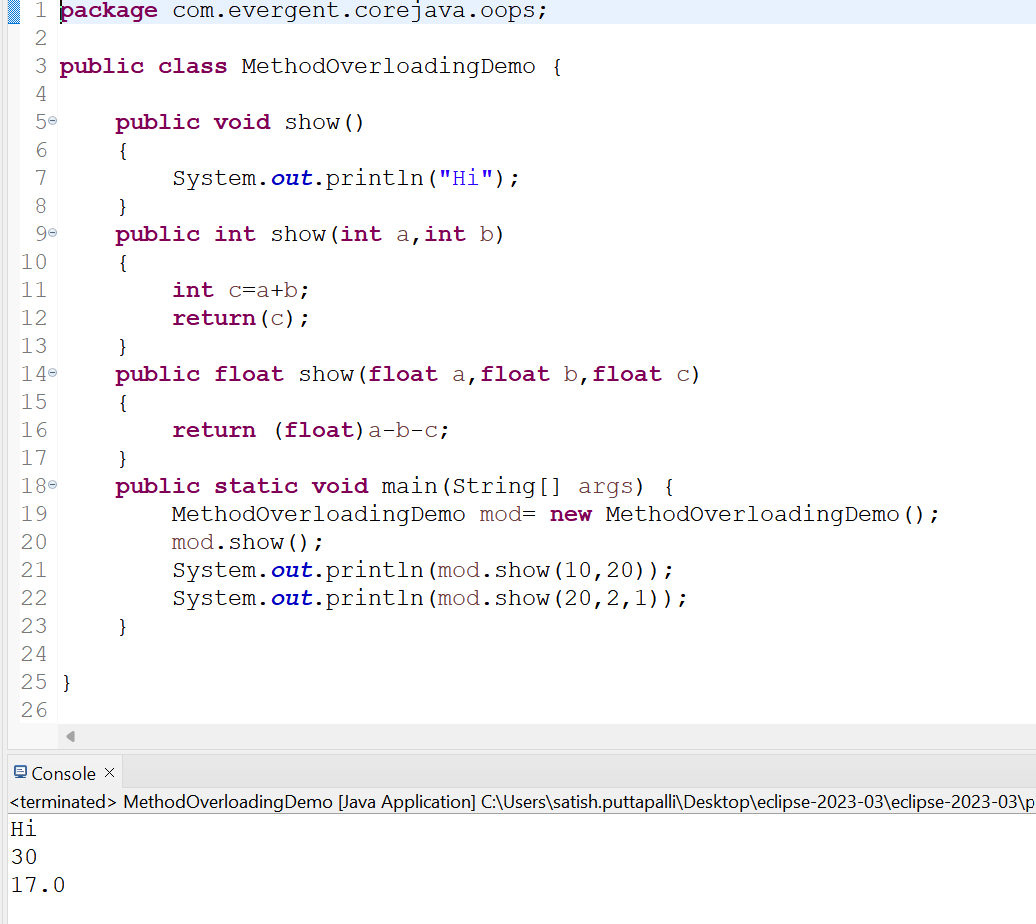
1. **Oops Concept’s** **8-8-24 Day-4**
   1. Encapsulation
      1. Object Creation
      2. Loosely coupled Example:-



* + 1. Method Types or Method Flows
       1. No Parameters with no return type
       2. Parameters with no return type
       3. Parameters with return type
       4. No Parameters with return type



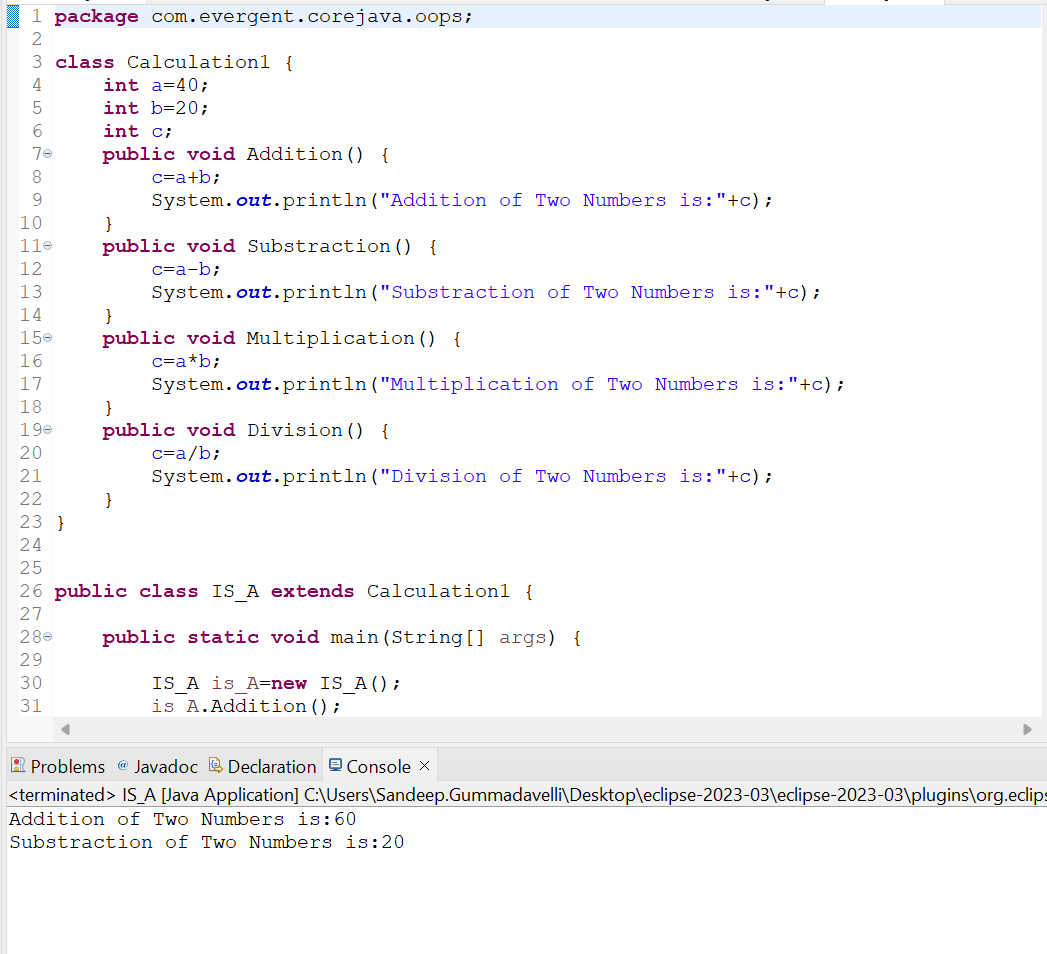
* 1. Inheritance 
  2. Polymorphism
     1. Method Overloading



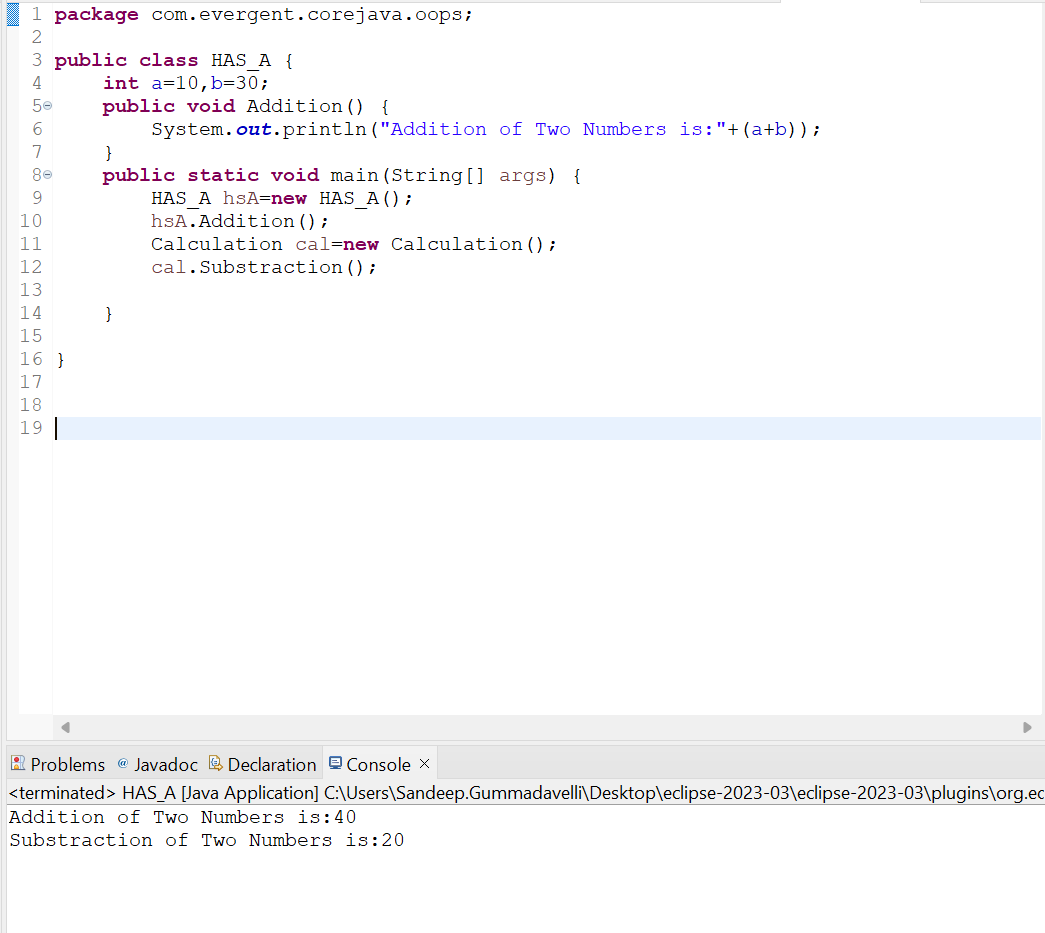
* + 1. Method Overriding



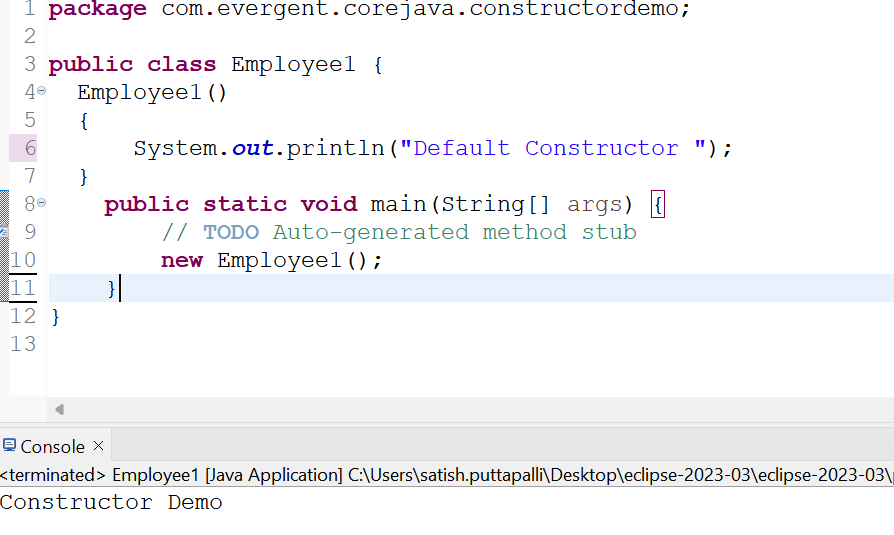
* 1. Abstraction
  2. IS-A



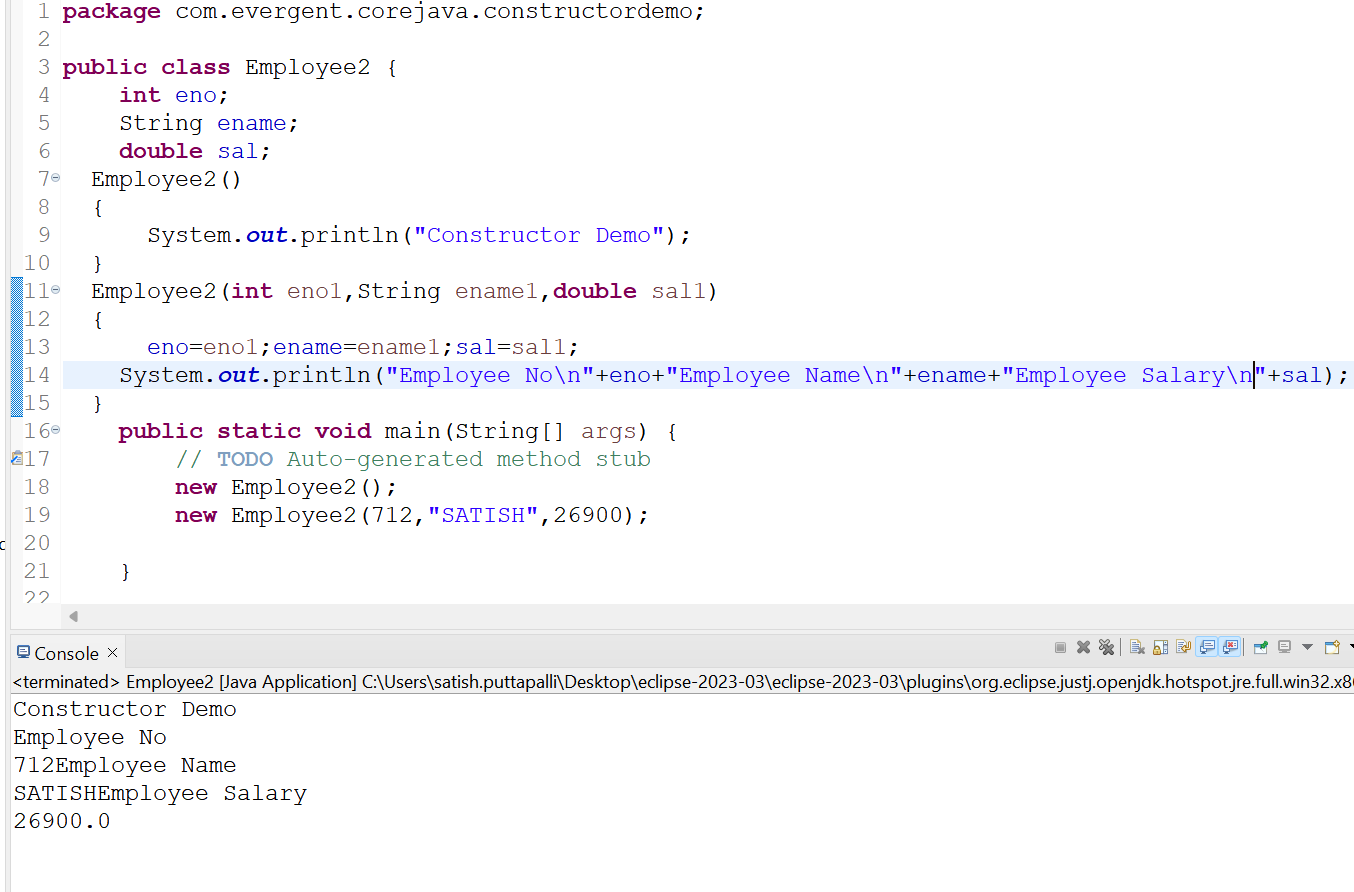
* 1. HAS-A



1. **Constructors -In Depth**
2. The Class name and the Constructor name should be same
3. There are two types of Constructors
4. Default Constructors
5. Parameterized Constructor
6. Object Copy Constructor
7. We can access the constructor while creation of object

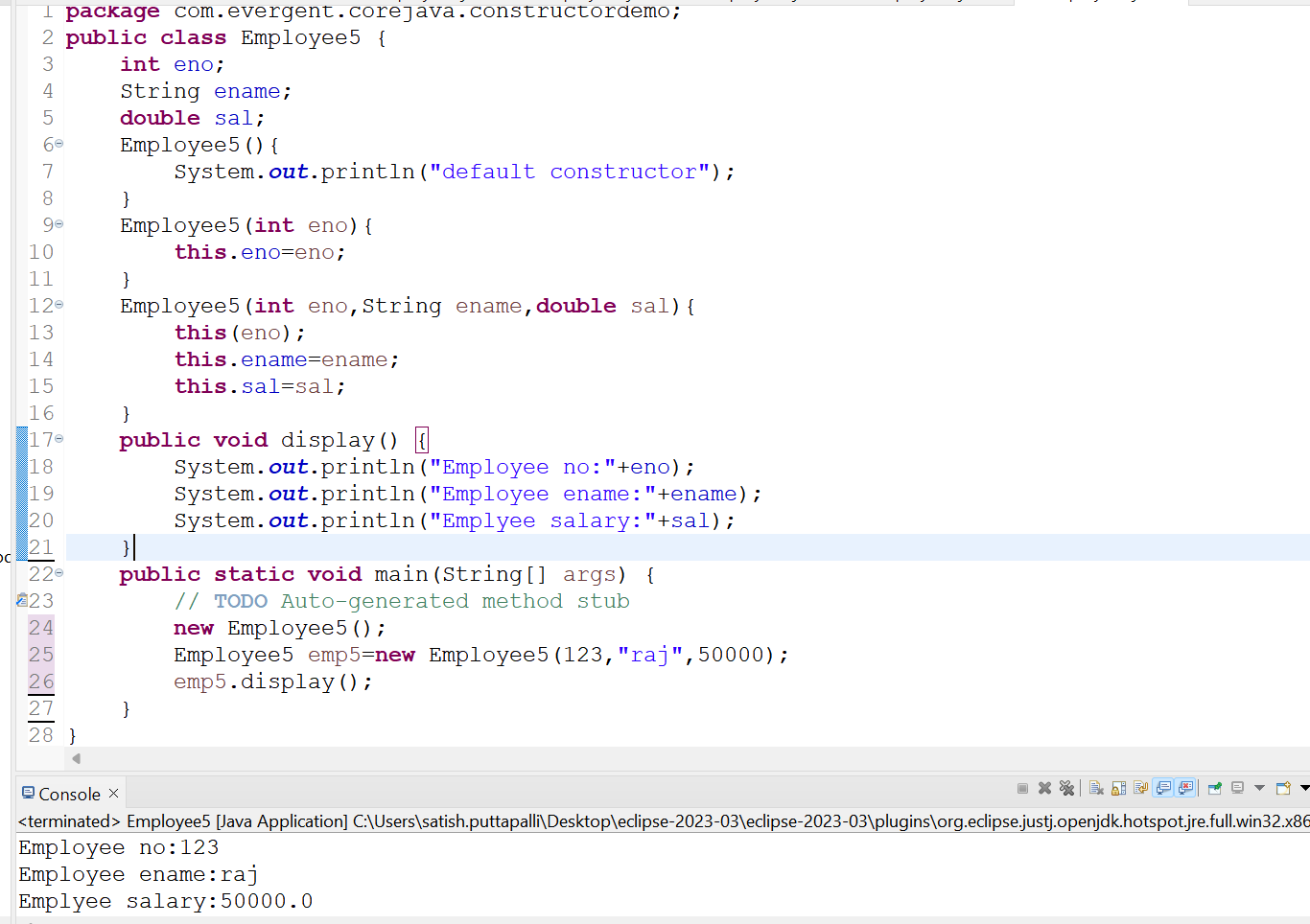


1. Constructors are mainly for initialization



1. Constructors doesn’t have any return type and void
2. If we Declare as a void it will consider as normal method not a constructor
3. Every class need’s at least on default constructor either the user will create or default constructor
4. Always Constructors are Overloaded
5. We can call one constructor in another constructor

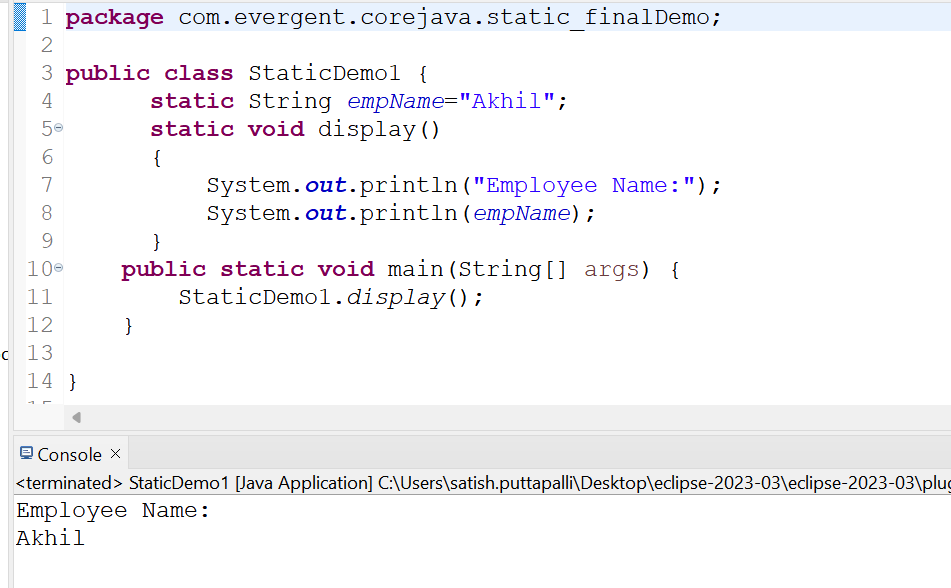
* **This Keyword:-**This is a keyword used to refer to the current instance variables and this keyword is used to invoke the one constructor to the other constructor in the same class



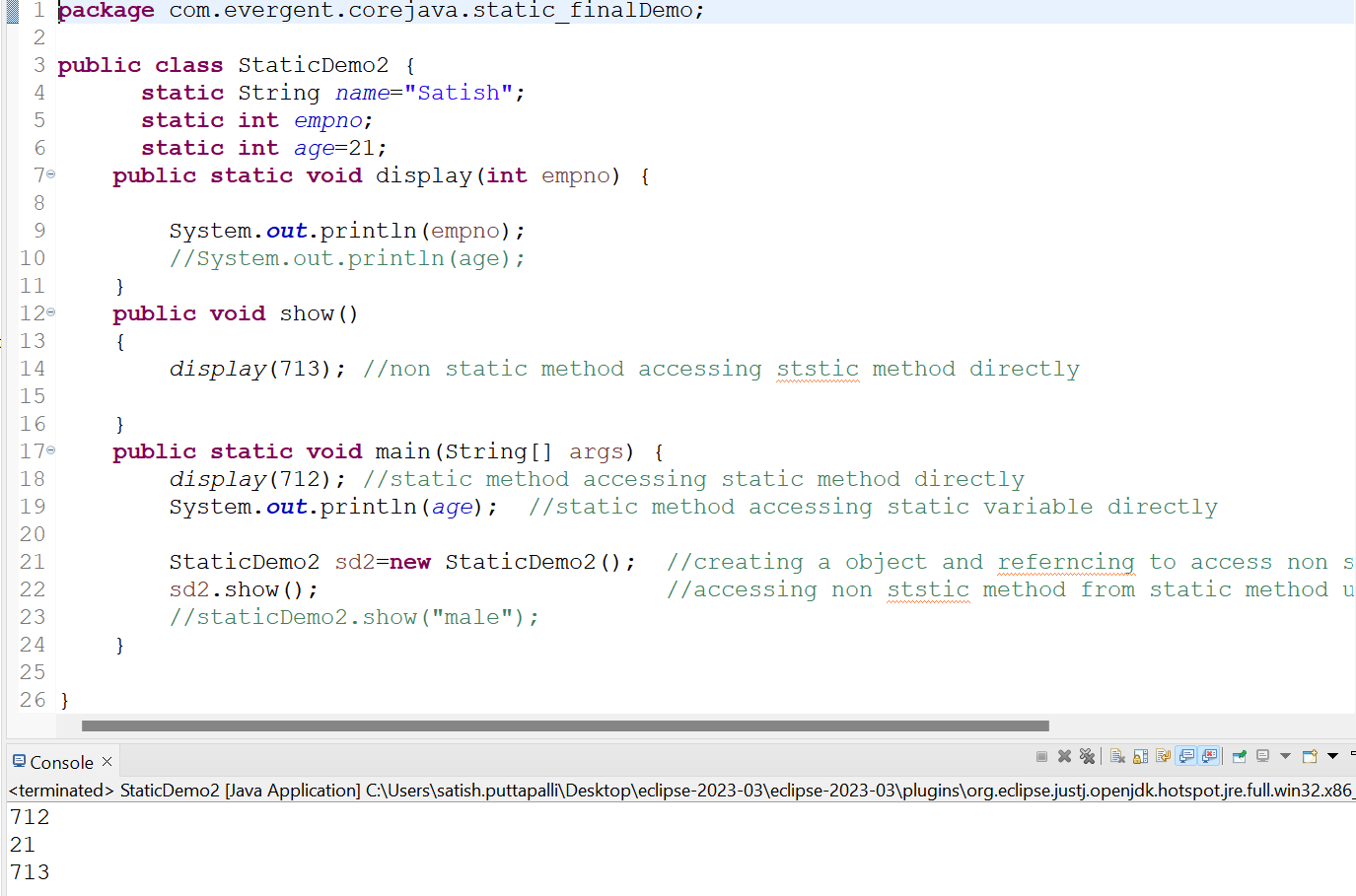
* **Super Keyword:-**Super is a keyword which is used to call the super class constructor from the base class constructors



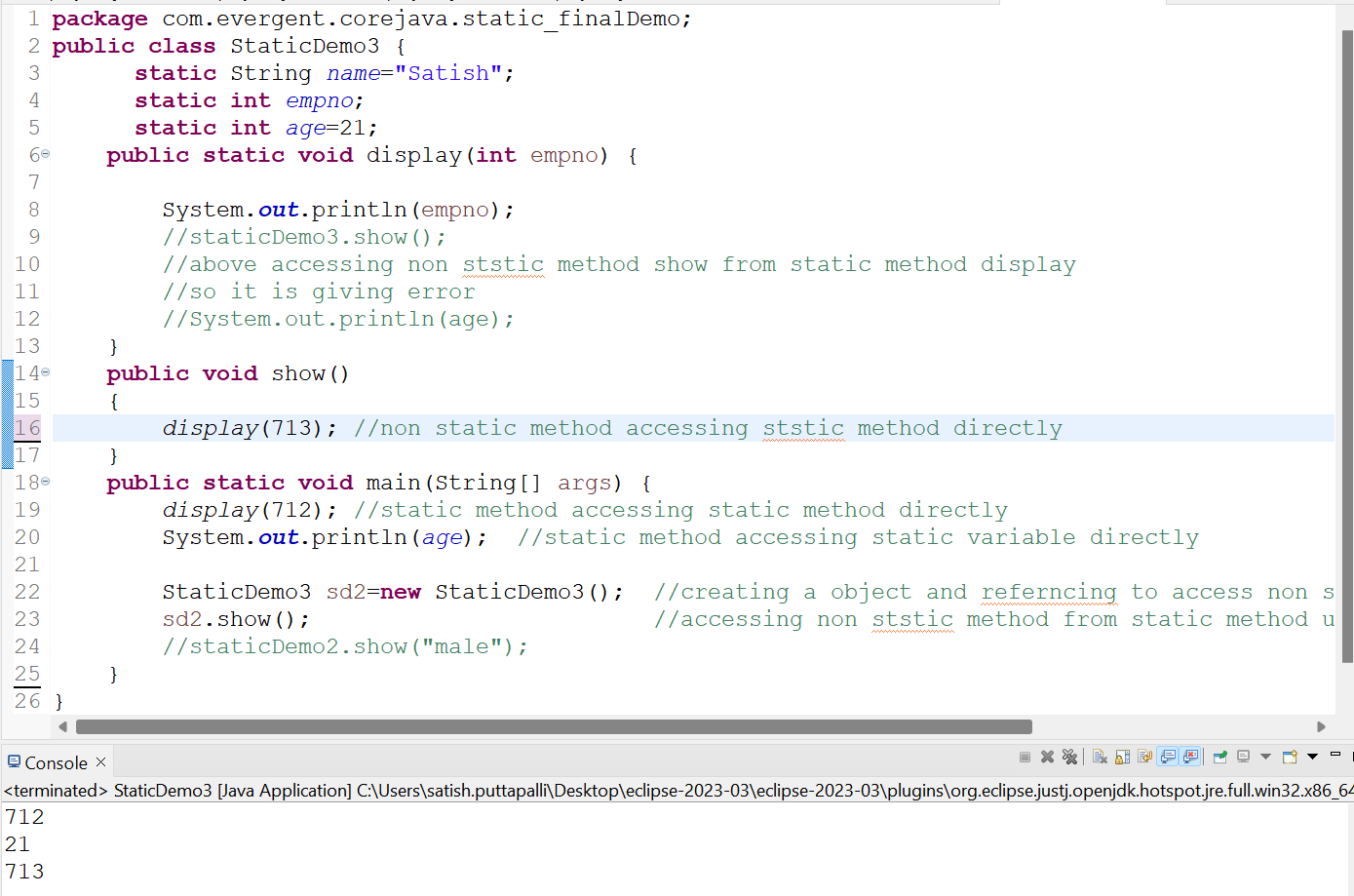
1. **static Keyword In Depth** **9-8-24 Day-5**
   1. static is keyword
   2. We can declare the static as variables and methods
   3. We can access the static methods and variables directly through className..method or className.variable



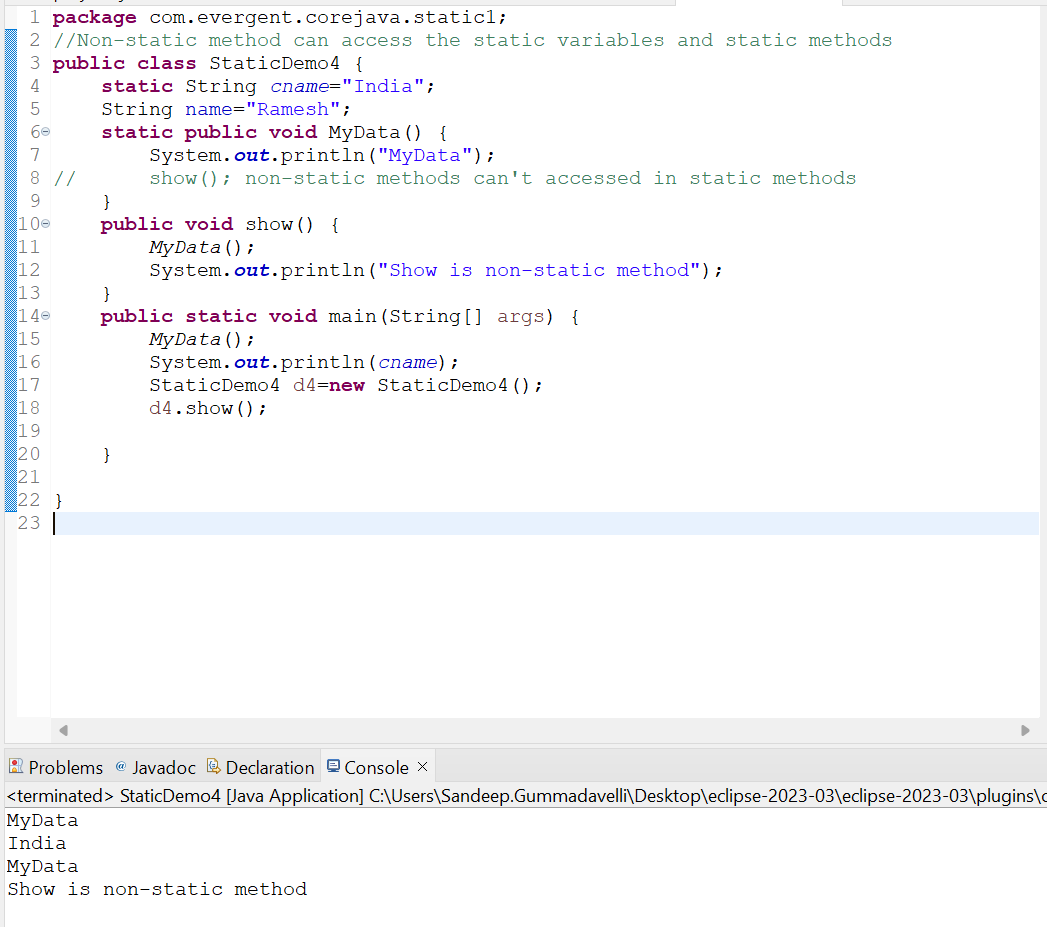
* 1. static methods can access the static variables and static methods



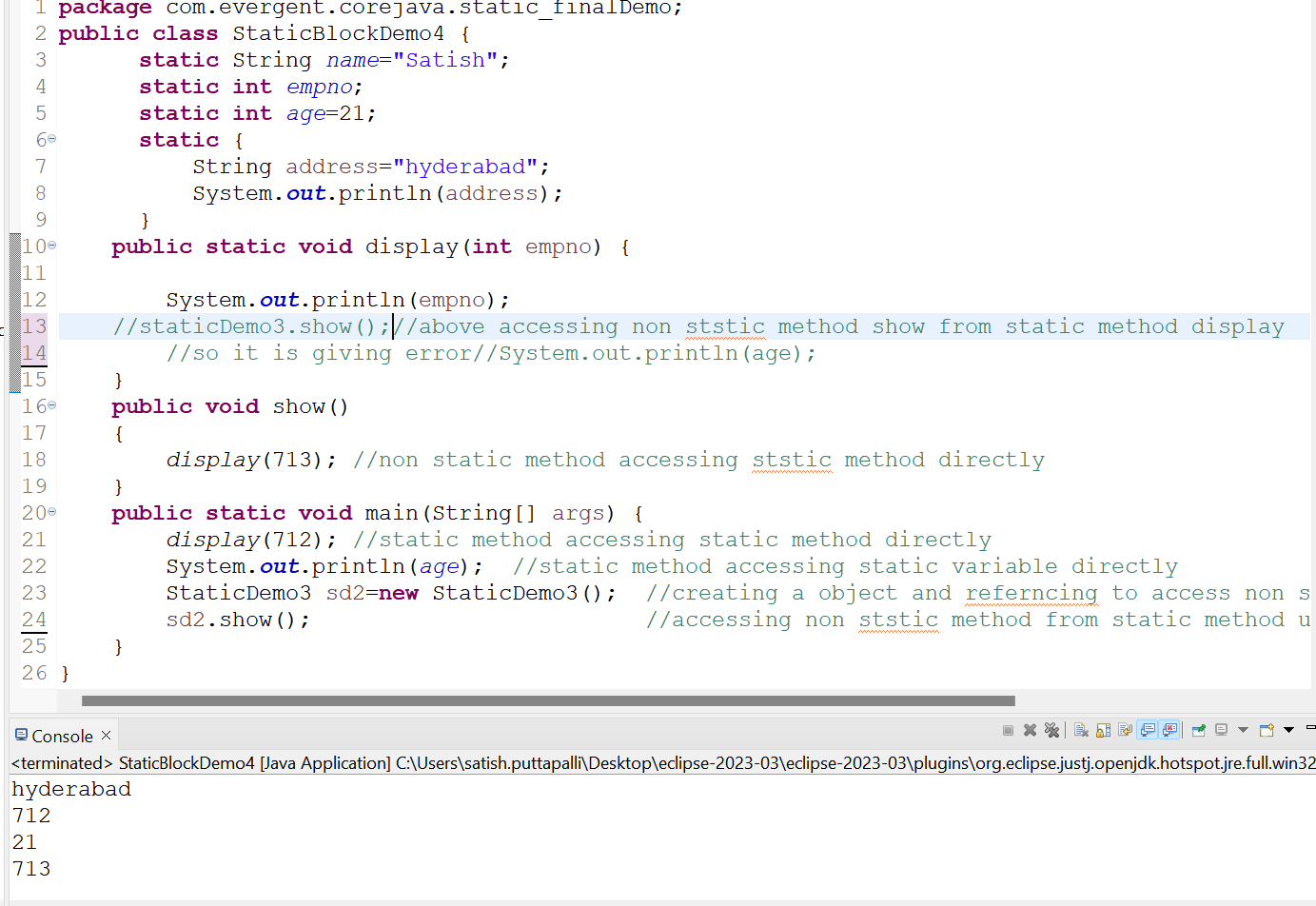
* 1. static methods can’t access non-static methods and variables



* 1. Non-static method can access the static methods and static variables

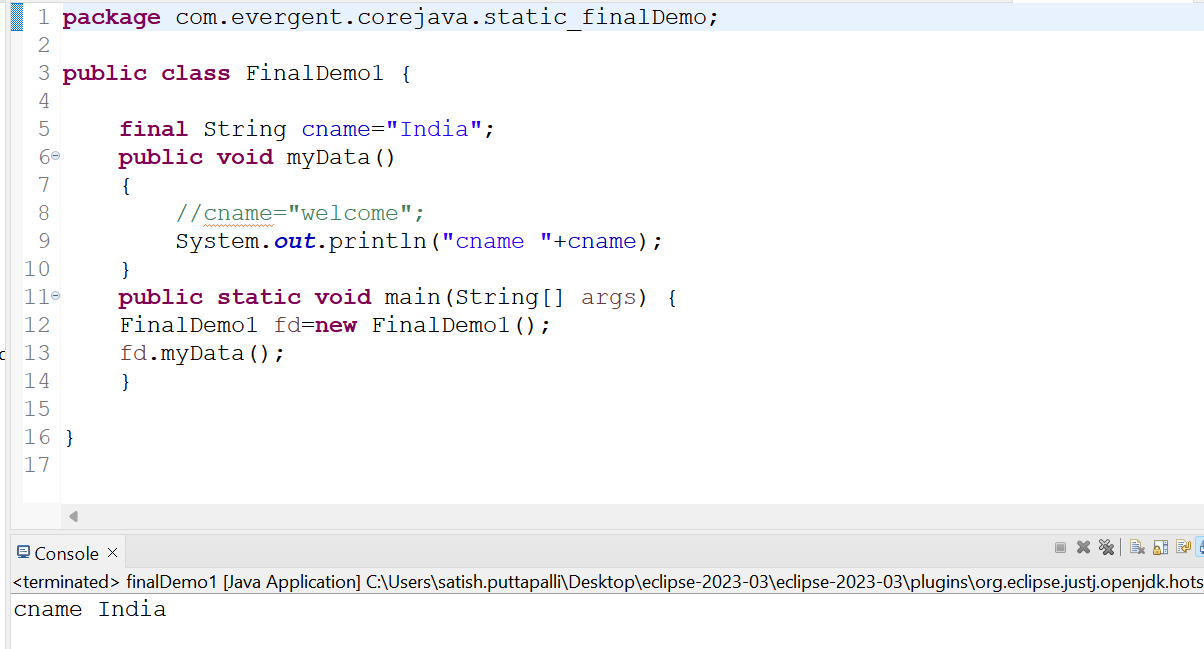


* 1. static block:-When class is loaded in to JVM static block get initialized first

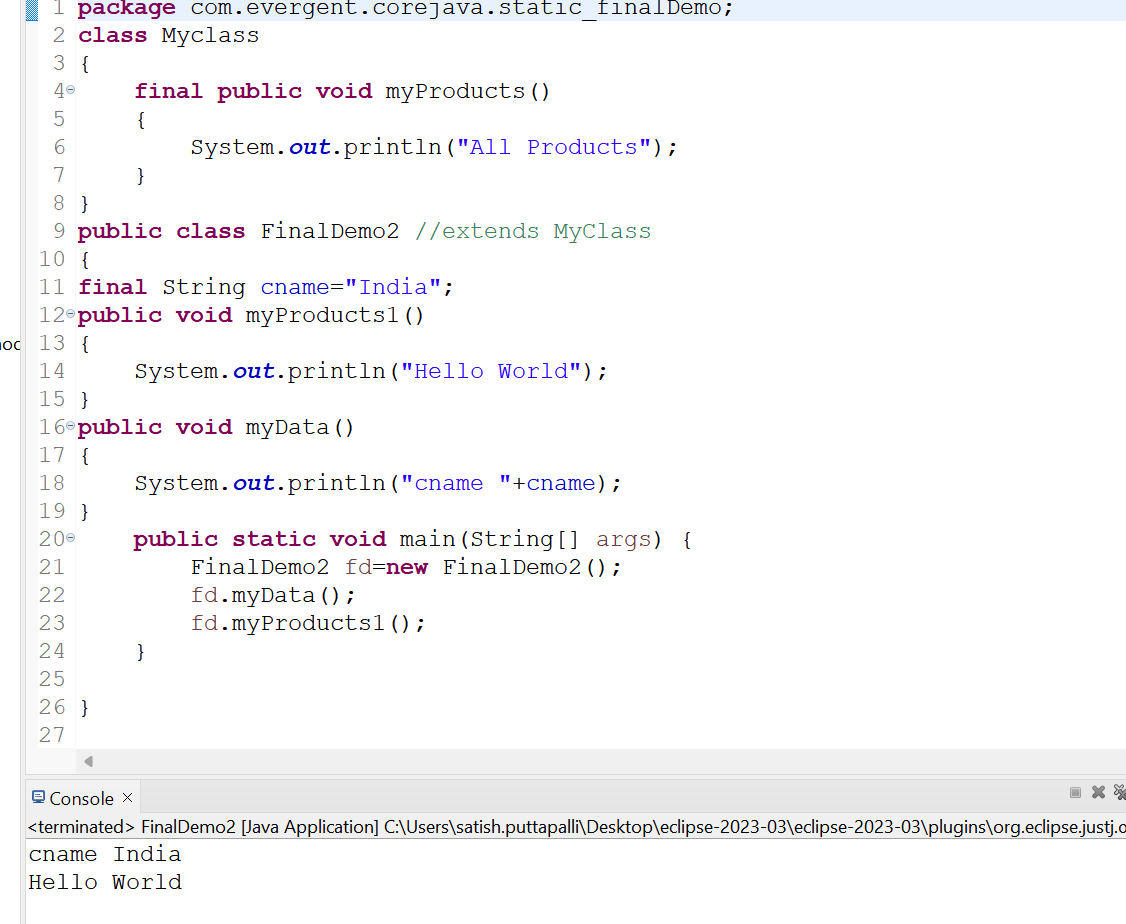


* 1. static variables get reflected when one make changes in method

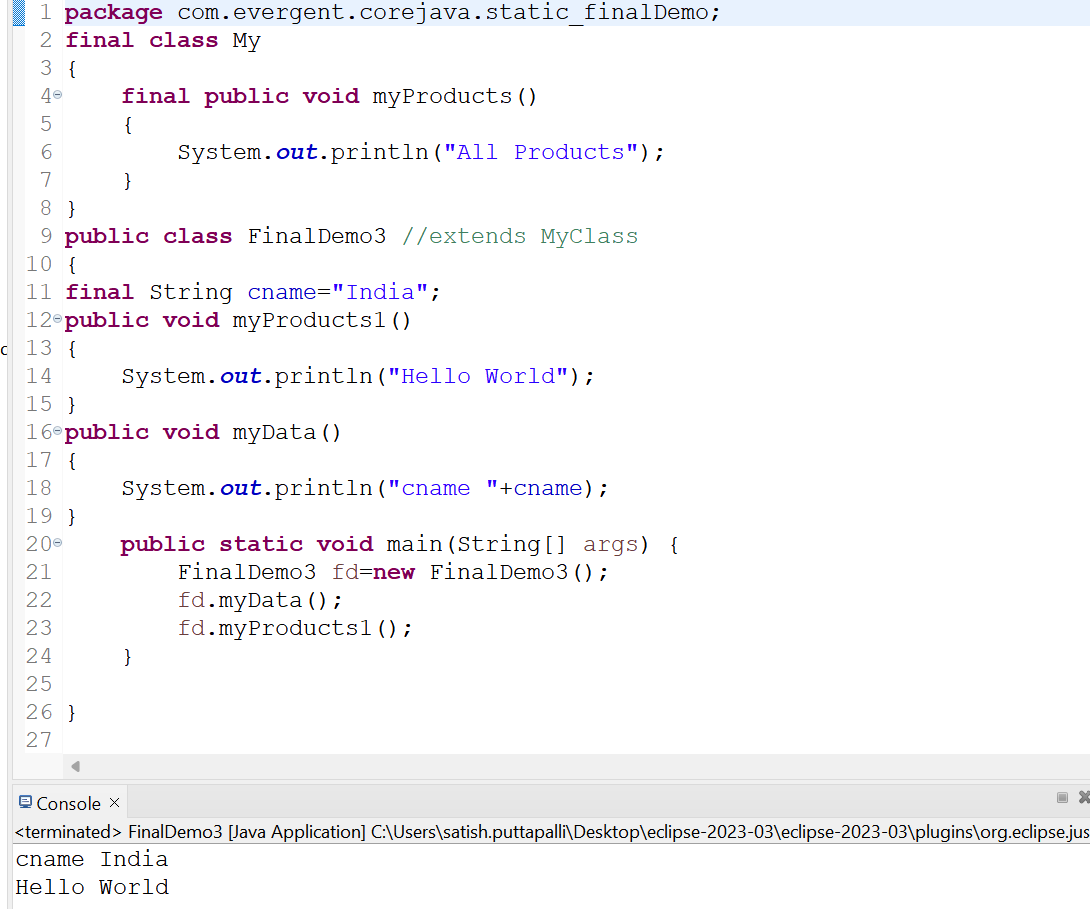
1. **final Keyword in Depth**
   1. final is a keyword
   2. We can declare final as variables,methods,classes
   3. final variables we can’t modify



* 1. final method can’t be overridden



* 1. final class can’t be inherited



1. **Strings** **12-8-24 Day-6**
   1. String class is final class
   2. String are immutable
   3. String methods are non-synchronize

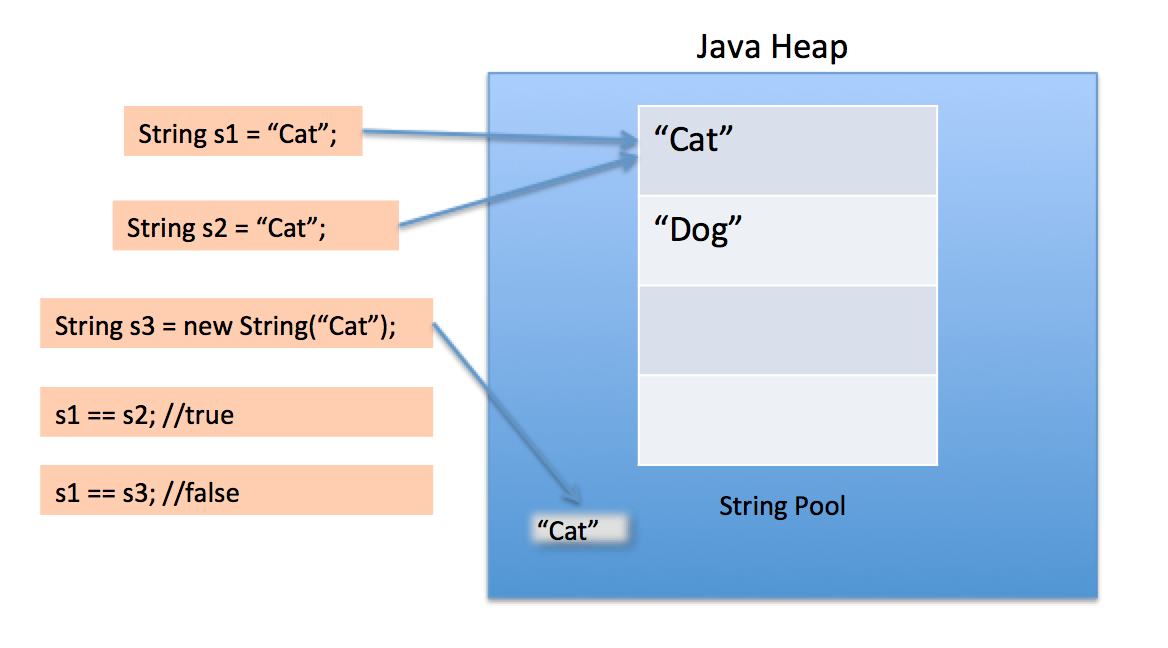
**String class Other Imp Points**

* 1. In java ,a string is a sequence of characters often used to represent text
  2. Once we declare any String objects it is a constant and if we are try to modify the existing string and it will create another memory location,Existing object is eligible to garbage collection
     1. Example:-

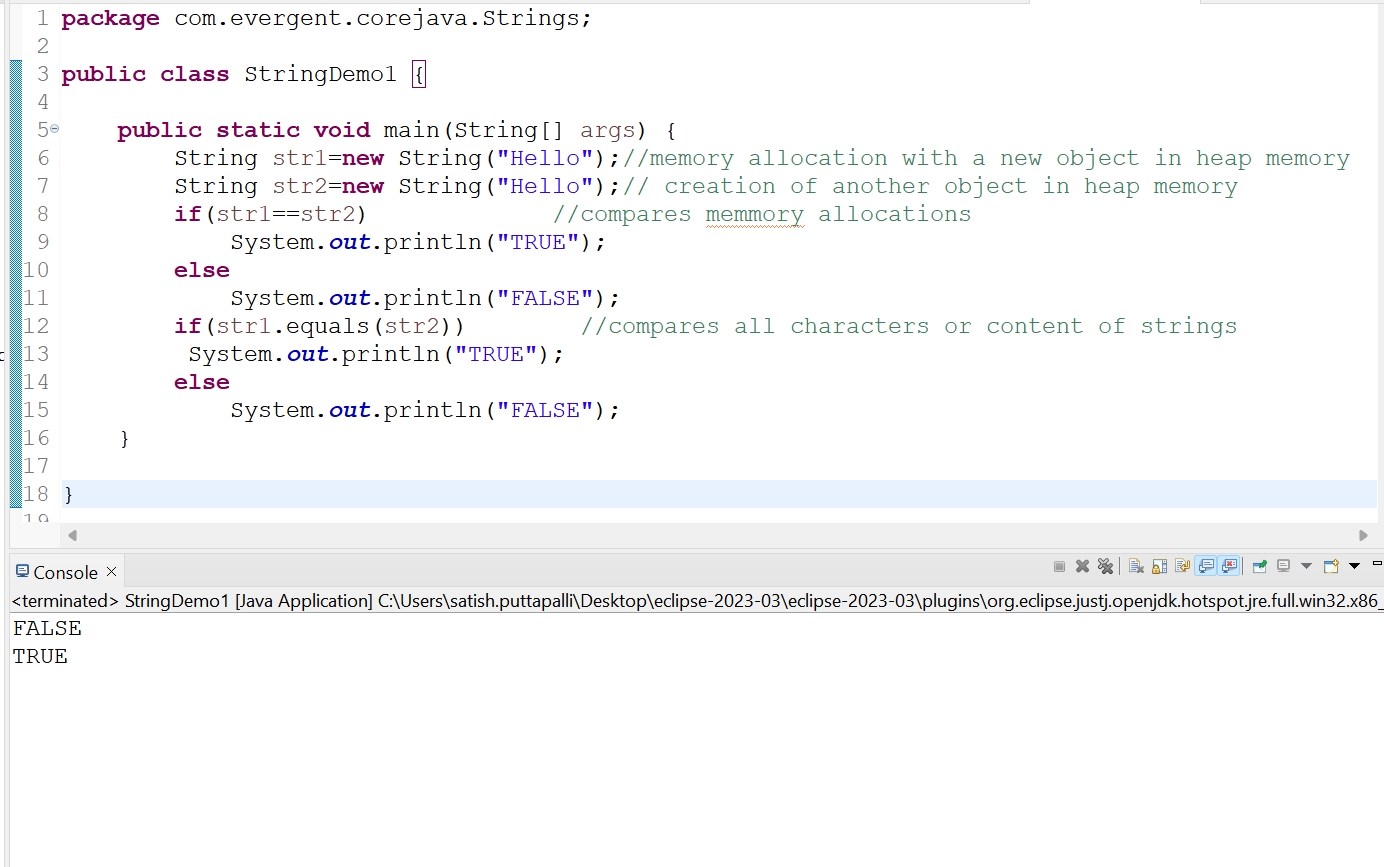
String name=new String(“Hello”);

Name=”Welcome”;

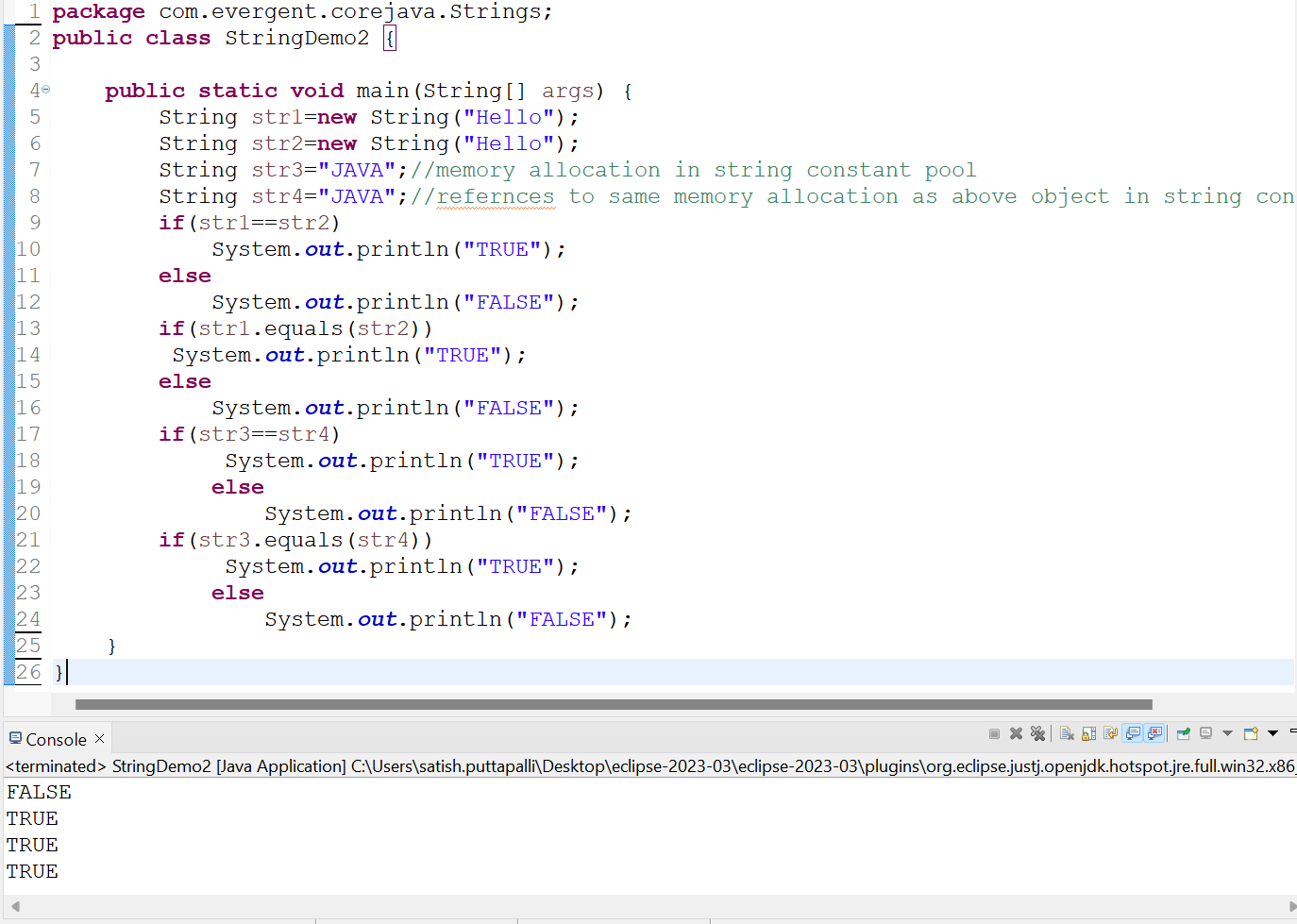
* 1. String objects in java and are instances of the string class ,which is part of the java.lang.package
  2. Key Features of strings in java :
     1. Immutable:-Once a string object is created it can’t be modified
     2. Any modifications to a string creates a new string object
  3. Java Optimizes memory usage by storing in s special area of memory known as the string pool
  4. If two Strings have the same value and are created without using new keyword,they will reference the same object in the string pool
  5. Storage of Strings in java done in two ways
     1. Heap Storage Memory
     2. String pool



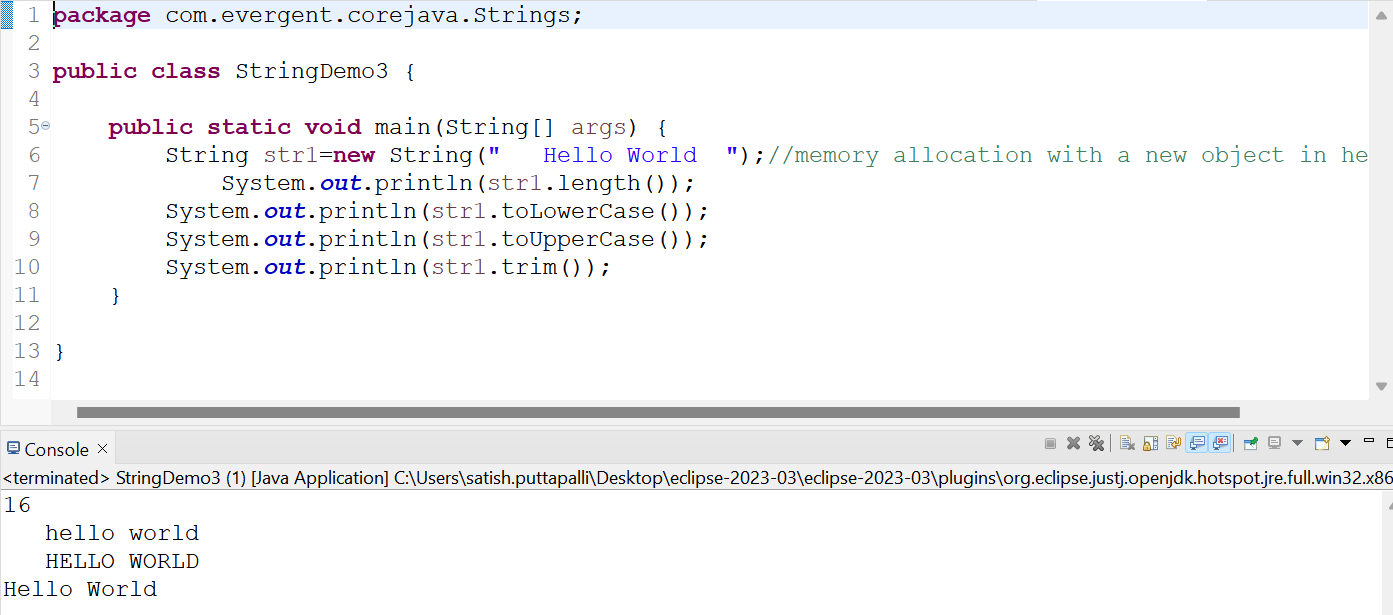
* 1. We can Declare the Strings in two ways I,e
     1. Using new Keyword like String str=new String(“Hello”); **---Heap memory storage**
     2. And the other way is like normal way of declaration like String str=”Hello”;**- String pool**
  2. String Program using new Keyword

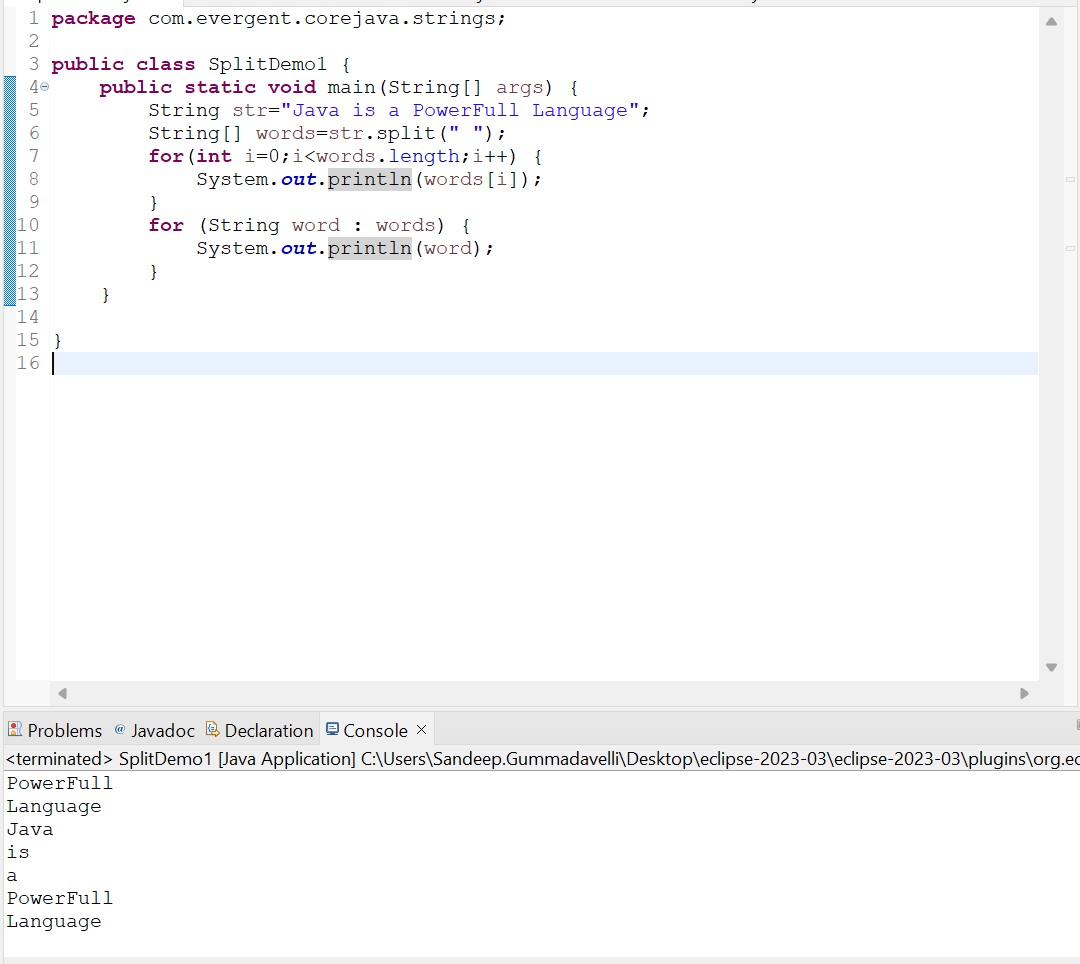


* 1. String Program without using the new keyword

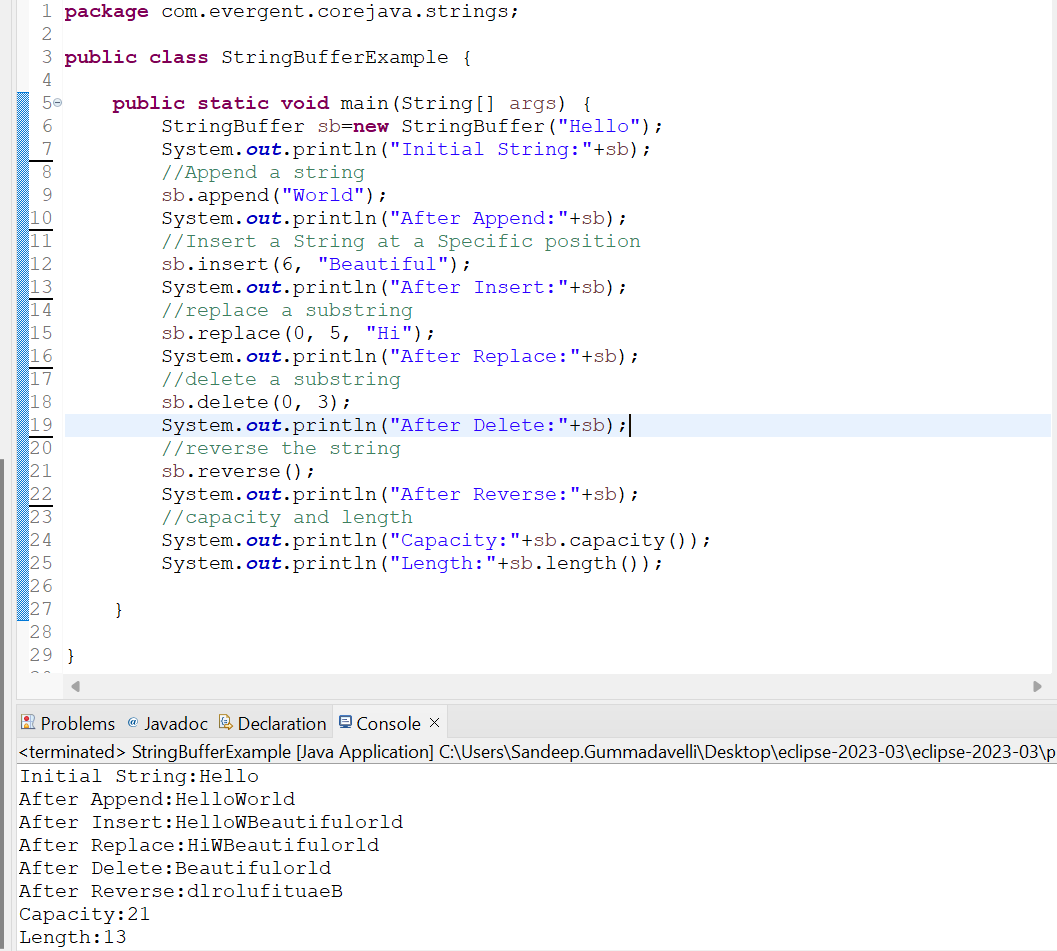


* 1. Different types of String methods include:
     1. Length();
     2. toLowerCase();
     3. toUpperCase();
     4. trim();
     5. startsWith();
     6. endsWith();
     7. equals();
     8. equalsIgnoreCase();
     9. toCharArray();
     10. indexOf();
     11. lastIndexOf();
     12. subString();
     13. split();
     14. replace();
     15. isEmpty();
     16. valueOf();
     17. isBlank();





1. **StringBuffer:-**In Java,StringBuffer is a final class and mutable and it has some methods include
   1. Append:-adds a string to the end of the current string
   2. Insert:-adds a string at a specific position in the current string
   3. Replace:-replaces a substiring with new string
   4. Delete:-removes s substring from the current string
   5. Reverse:-reverses the order of characters in the current string

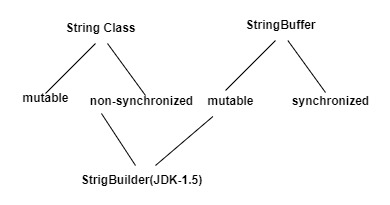


1. **StringBuilder:-**In Java,StringBuffer is a final class and mutable and it has some methods include
   1. Append:-adds a string to the end of the current string
   2. Insert:-adds a string at a specific position in the current string
   3. Replace:-replaces a substiring with new string
   4. Delete:-removes s substring from the current string
   5. Reverse:-reverses the order of characters in the current string



1. Difference Between **String ,StringBuffer,and StringBuilder**

|  |  |  |
| --- | --- | --- |
| **String** | **StringBuffer** | **StringBuilder** |
| 1.String is final class | 1.StringBuffer is final class | 1.StringBuilder is also final class |
| 2.String is Immutable | 2.StringBuffer is mutable | 2.StringBuilder is also mutable |
| 1. All methods are non-synchronize   (Not Thread safe) | 3.All methods are synchronize(Thread safe) | 3.All methods are non-synchronize  (Not Thread safe) |
| 1. String methods include Length(),toUpperCase(),   toLowerCase()..etc | 4.StringBuffer has methods like append(),insert(),replace(),  Delete(),reverse(),capacity(),length() | 4.StringBuilder has methods like append(),insert(),replace(),  Delete(),reverse(),capacity(),length() |
|  |  |  |



1. **String class Performance**



**13-8-24 Day-6**

1. We can Create our own Immutable Class
   1. We can declare class as final
   2. The class is declared as final so that it can’t be subclassed

**Private Final Fields:-**

**-** We can declare class as final

- The field and age are private & final

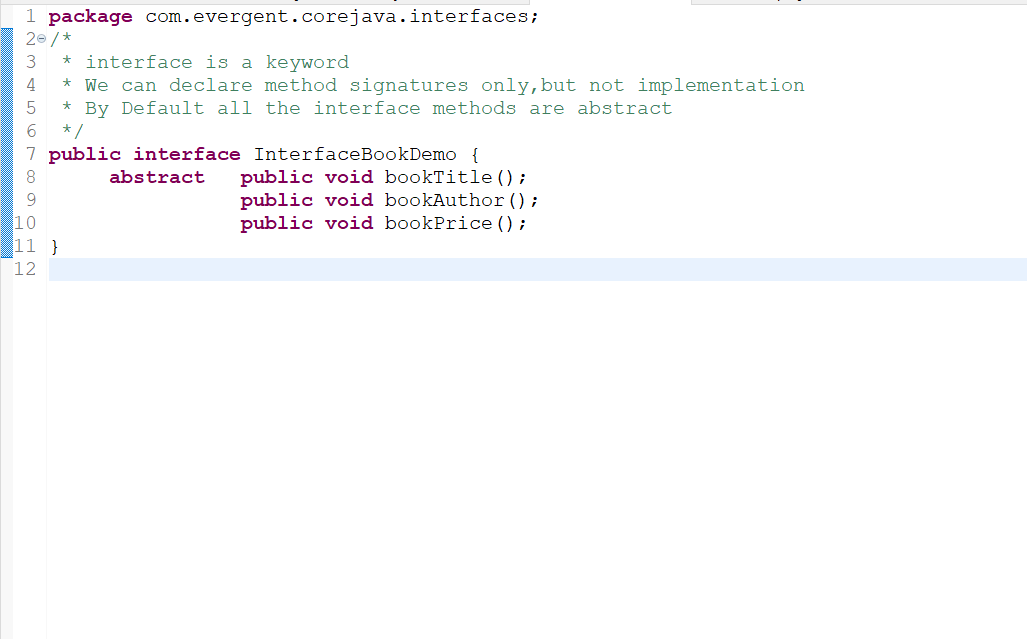
**Constructor:-**

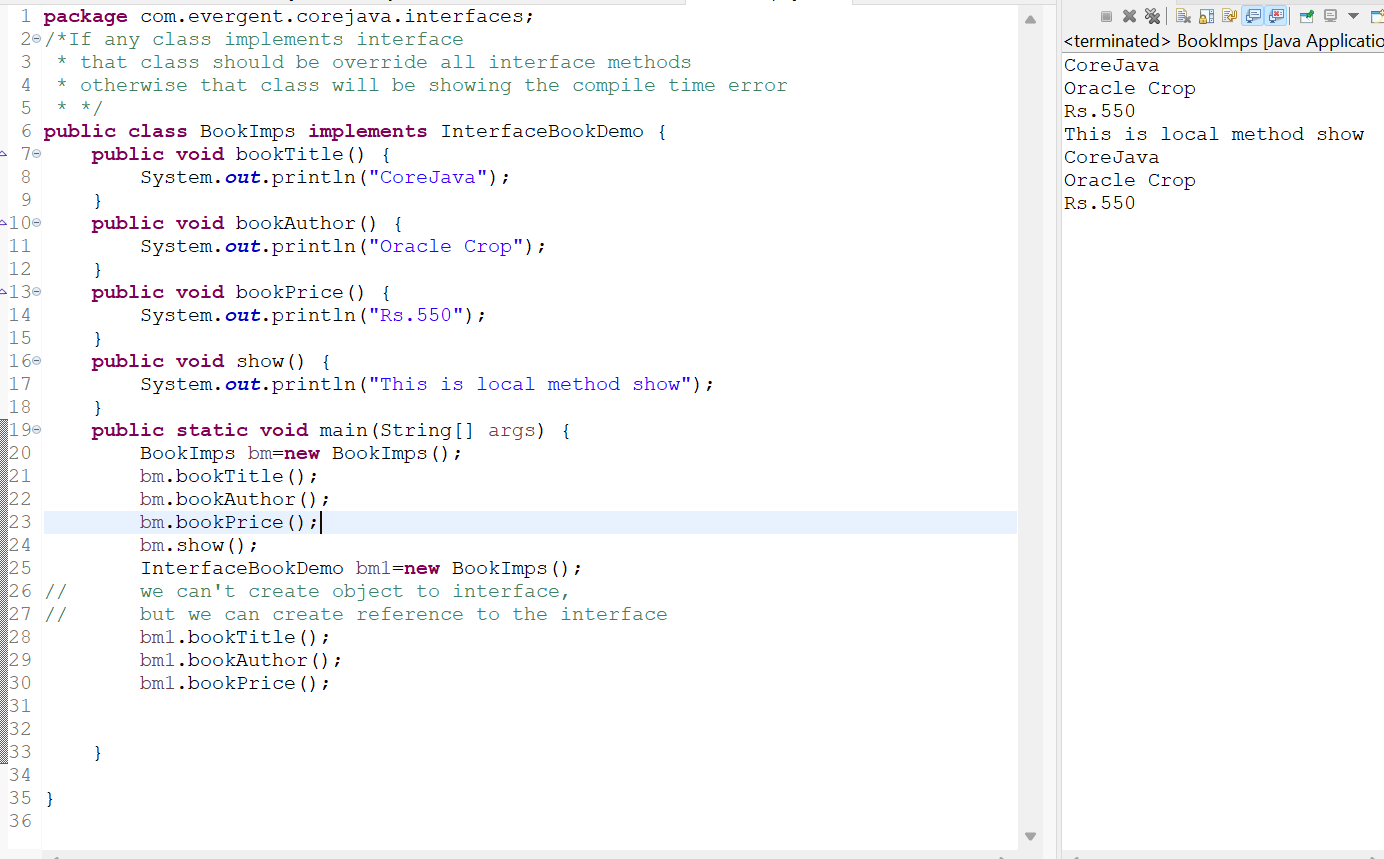
- The constructor initializes the final fields when a person object is created





1. **Interface’s -In Depth**
   1. Interfaces is keyword
   2. We can declare methods signature only not implementation
   3. By Default all interface methods are abstract
   4. If any class implements interface that class should be override all interface methods otherwise that class will be shown with compile time error
   5. We can’t create object to interface,but we can create reference to the interface

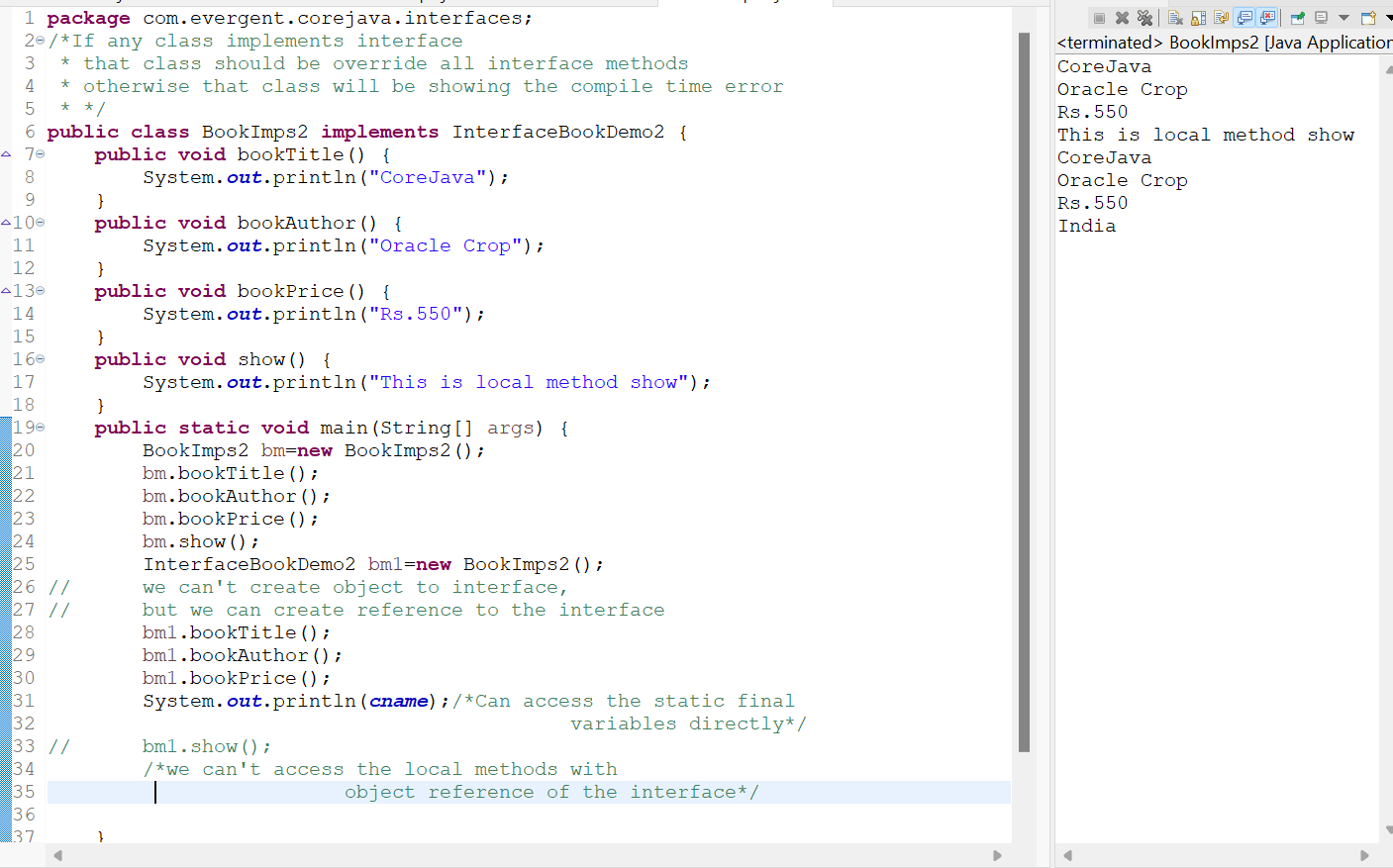




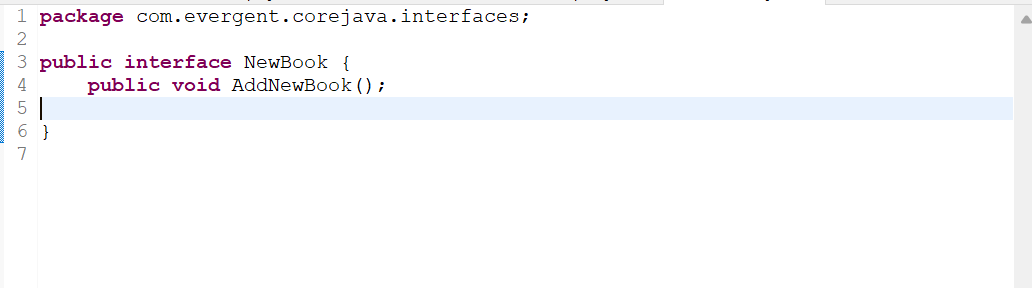
**we can't access the local methods with object reference of the interface**

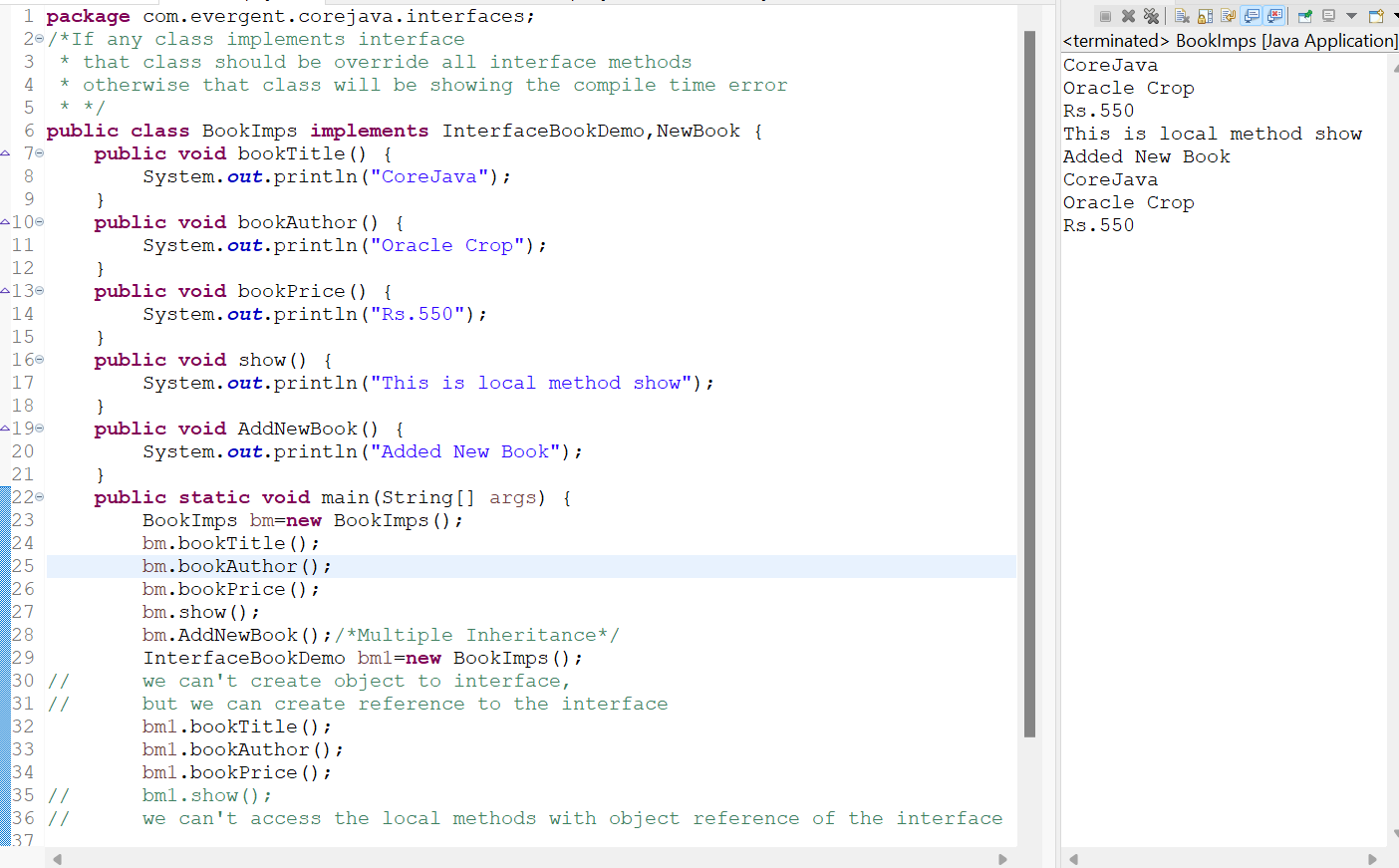


* 1. We can declare variables and by default the variables are default final

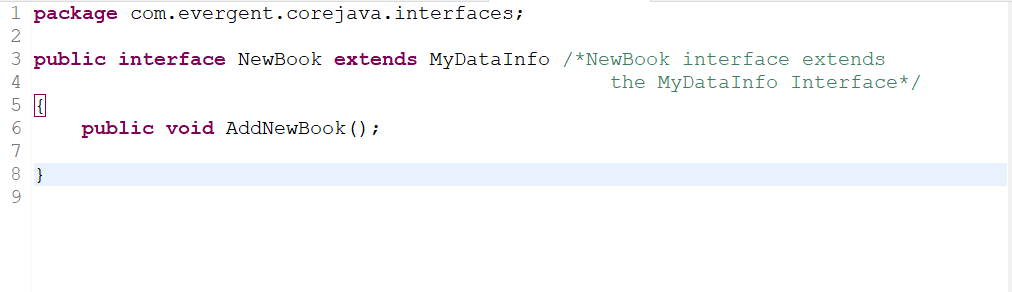


* 1. Java will support multiple inheritance through interface





* 1. One class can implements interfaces
  2. One interface can extend other interfaces



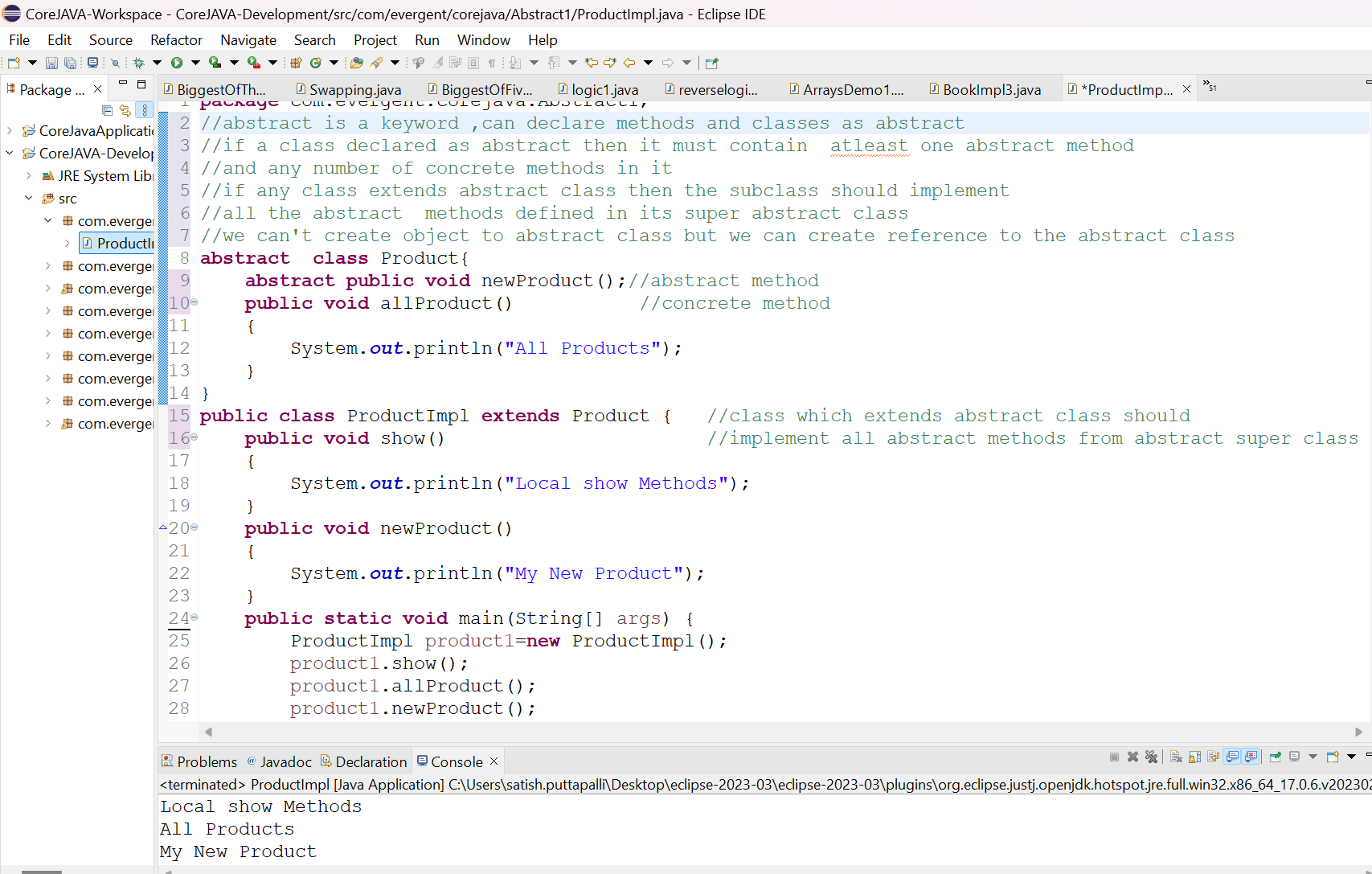
1. **Abstract Classes and Methods**
2. Abstract is a keyword in java
3. Abstract class must have at least one Abstract method and any number of concrete methods
4. If any class having one abstract method then the class containing it must be declared as abstract

else it will give compile time error

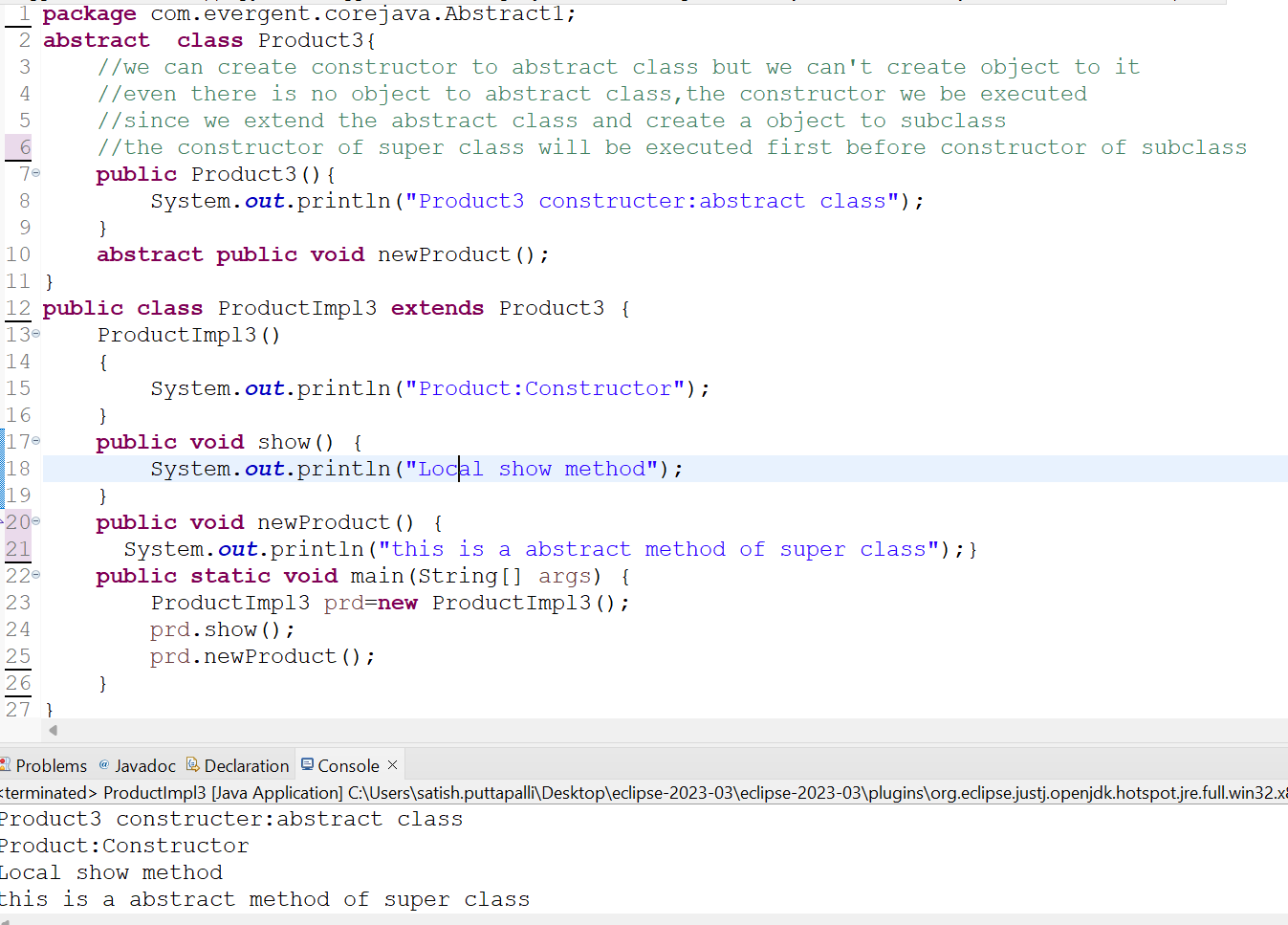
1. If any class extends abstract class then the subclass should override all abstract methods else it

will give compile time error

1. We can’t create object to abstract class but we can create object reference to abstract class.



1. We can create a constructor to the abstract class,even there is no object creation
2. We can access the abstract class constructor through subclass object creation
3. Abstract class constructor will be executed first before the sub class constructor



1. **Exceptional Handling**

1)Exceptional Handling is a mechanism

2) Exceptions are inbuilt mechanism of JAVA

5)Once any exceptions are occuring in java code then remaining lines of codes is unreachable

6) java.lang.Throwable is a superclass for Exception and Errors.

7)There are two types of Exceptions in java

A)Checked Exception

B)Unchecked Exception

8)All checked Exceptions are compile time exceptions

9)all Unchecked Exceptions are runtime exceptions

10)there are 10 keywords in exceptions Handling

a)try

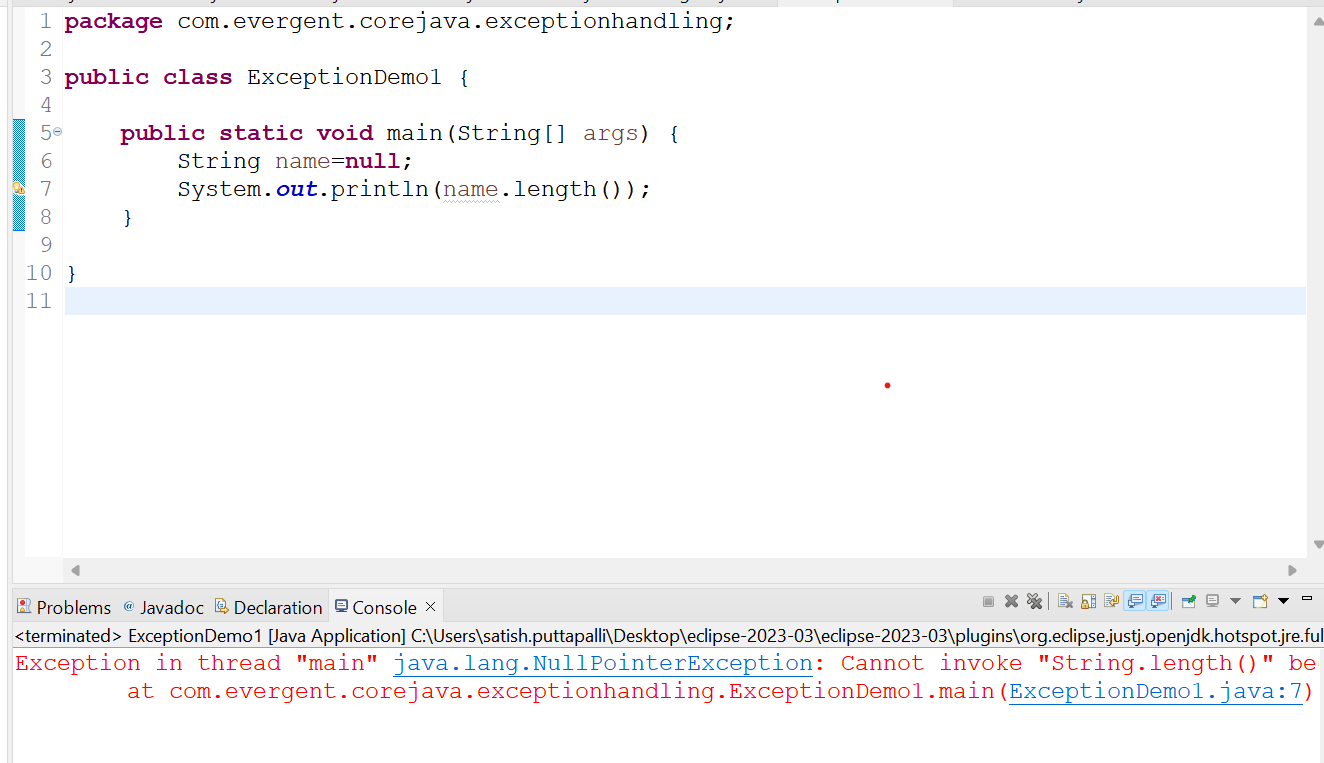
b)catch

c)finally

d)throws

e)throw

20)All exceptions classes are in to java.lang package



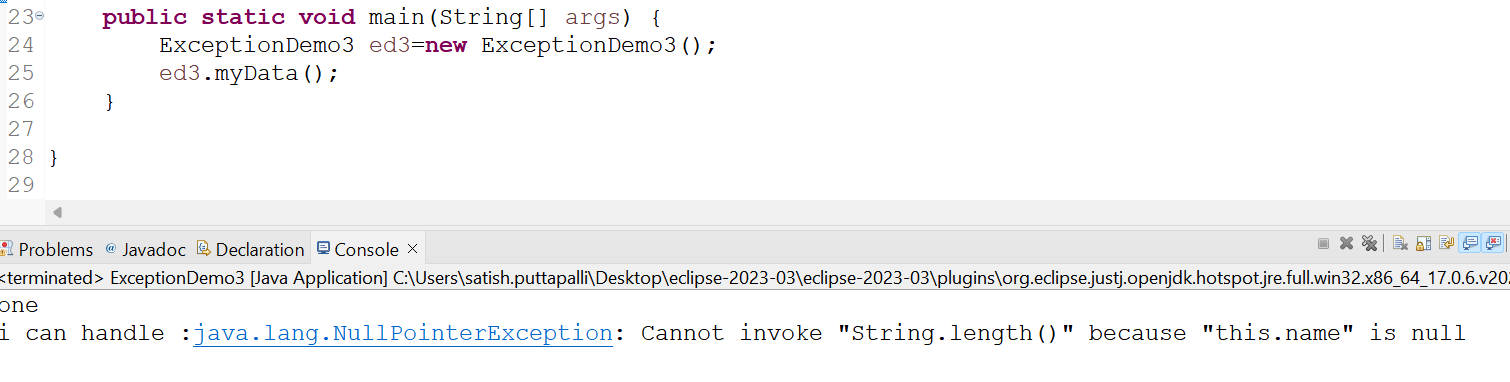
3) All Exception are executed while abnormal conditions only

4) Normal flow it won’t execute any exceptions



21)If there are multiple exceptions in class ,developer should be handle 1st

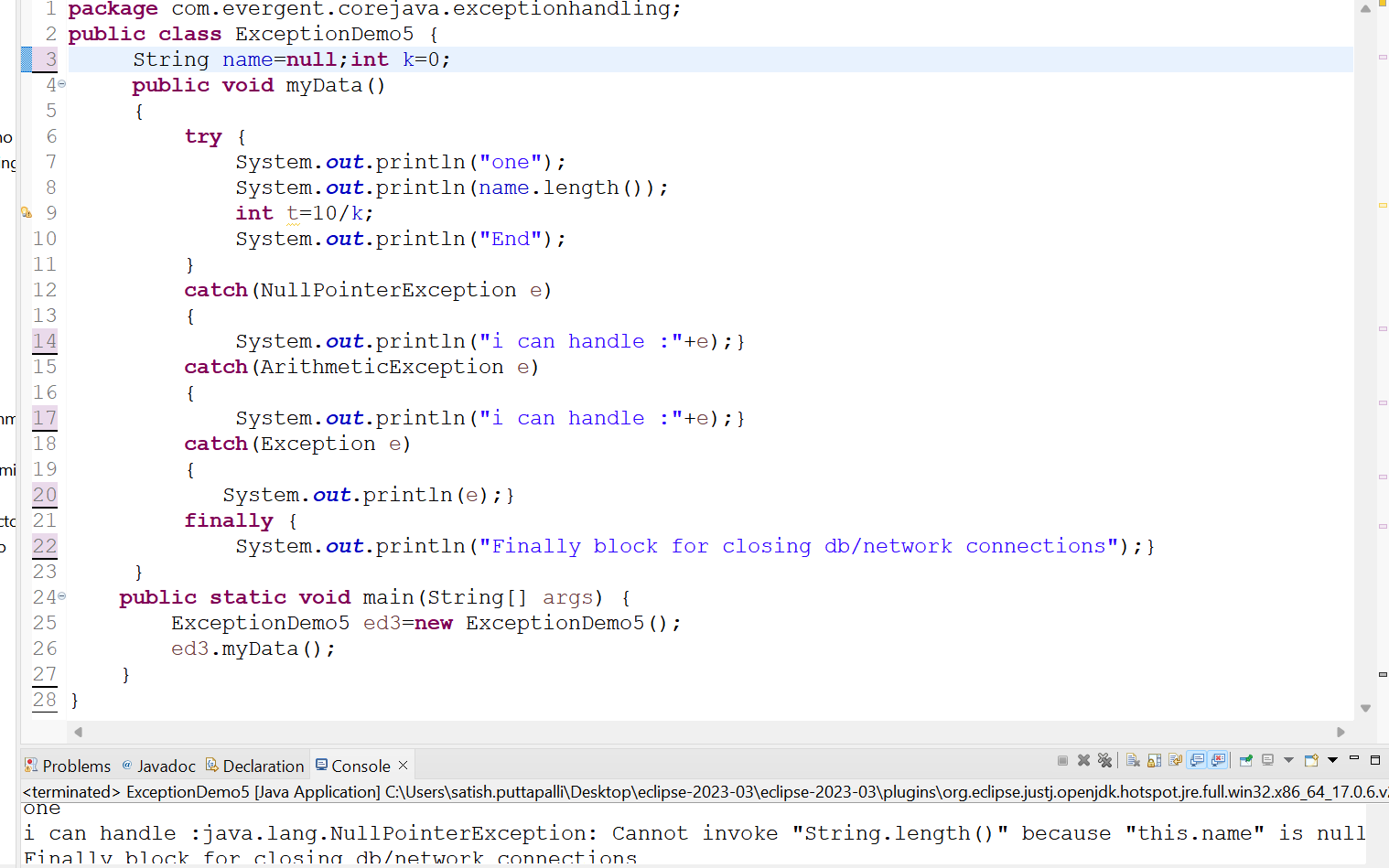
Exception then after 2nd exception will be handled 



1. We should follow exceptional hierarchies



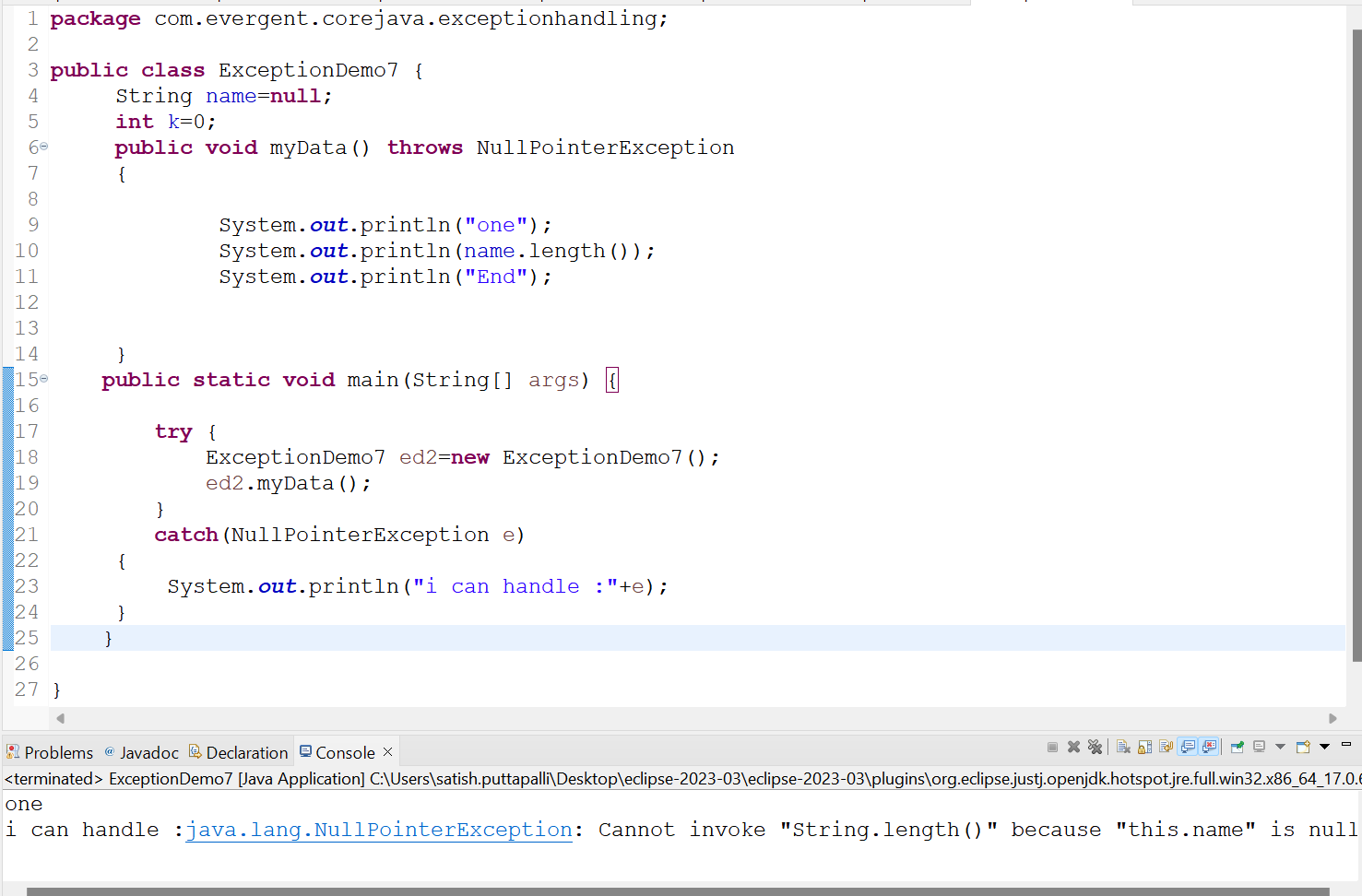
1. Finally ,is a block ,irrespective of exceptions it will be executed



1. try block is to write bussiness logic
2. Catch for handling exceptions
3. try followed by either catch block or finally block



1. throws-an exception will be executed method by method

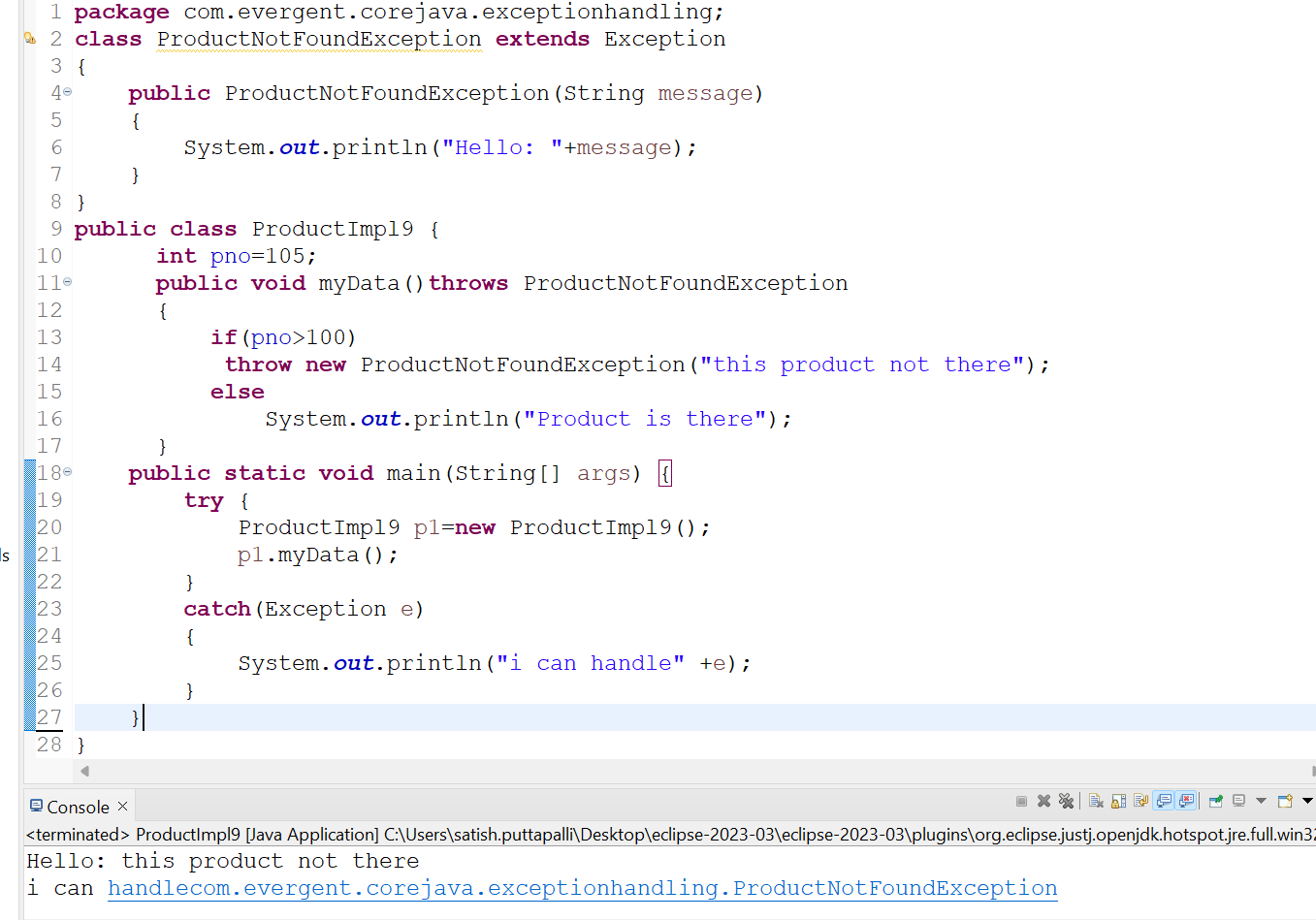




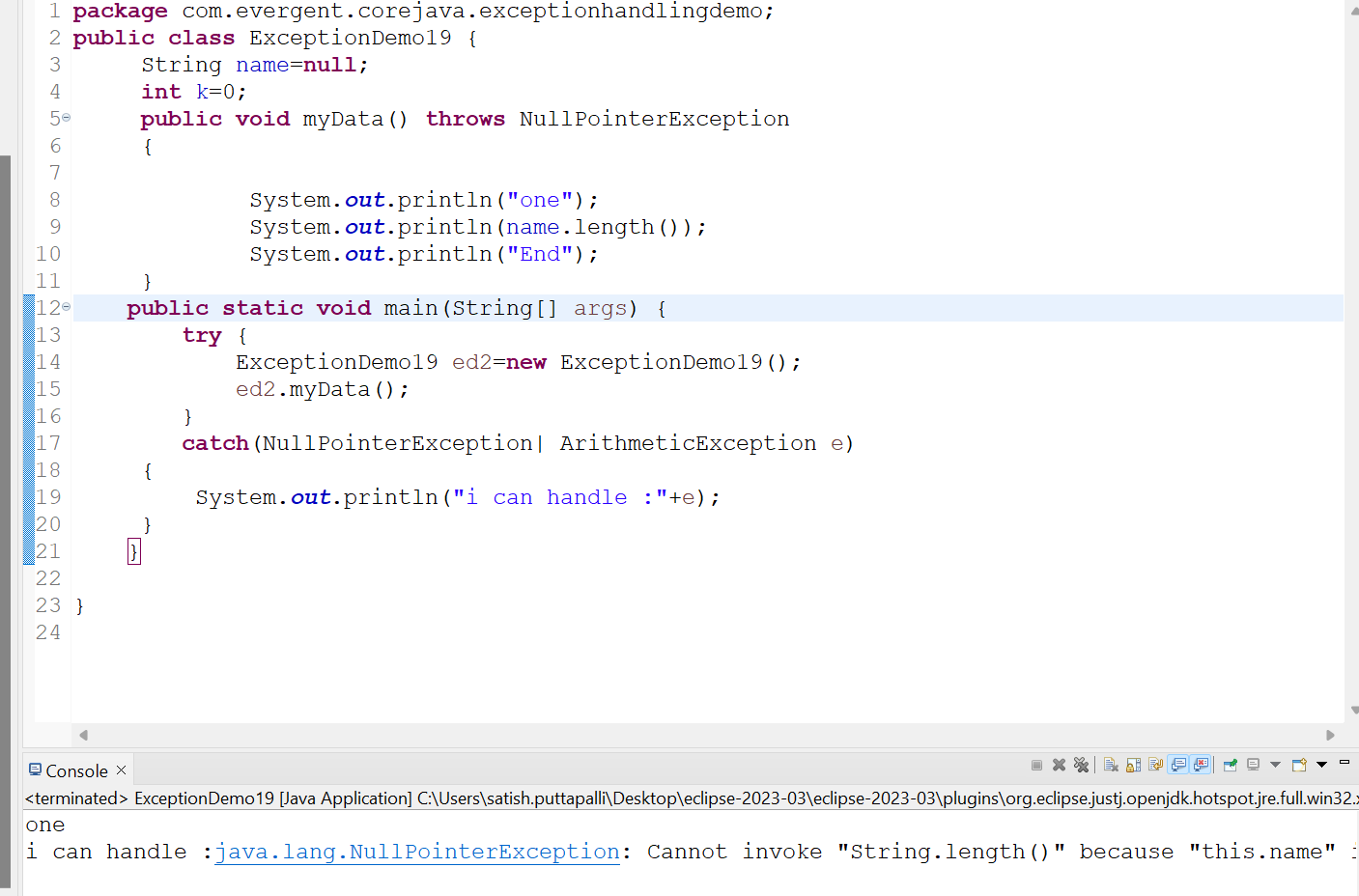
1. Throw is for runtime exceptions and will call predefined exception classes

18)we can create our own exceptions(userdefined)

19)our own exceptions extends Exception or runtime exception



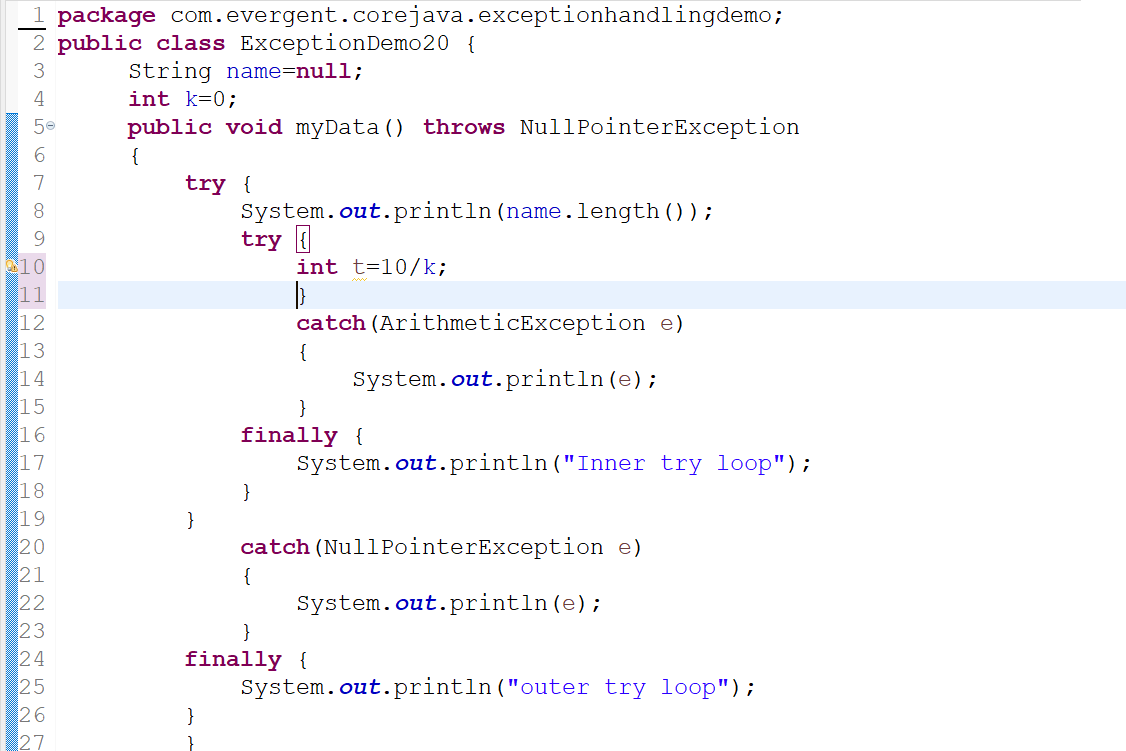
1. Handling Multiple exceptions using catch block can be done by separating exceptions using pipe(|)symbol

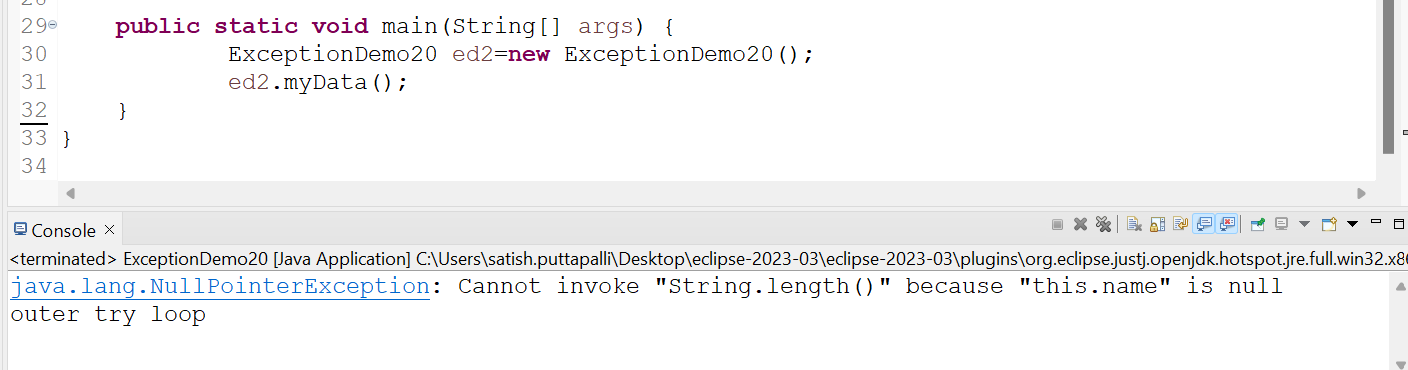


1. Handling multiple exceptions using throws keyword by separating exceptions using comma(,)



1. we can write try inside try block





**[Back To Index](#table_of_content)**