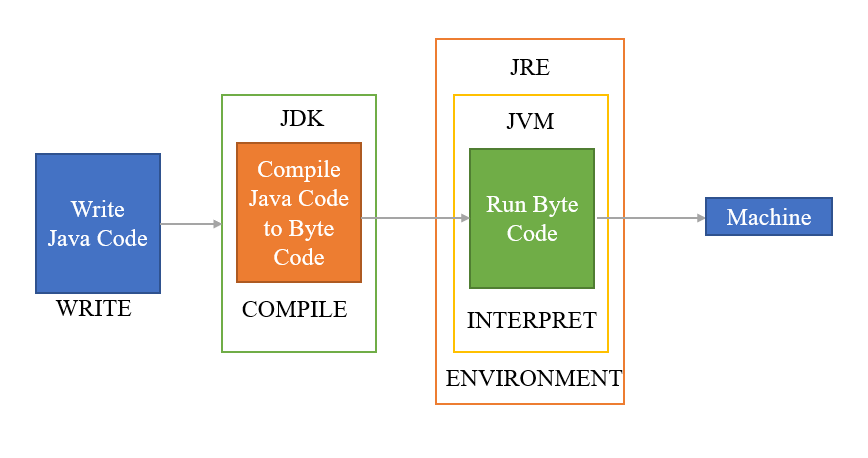
**Core Java Index Version 1.0**

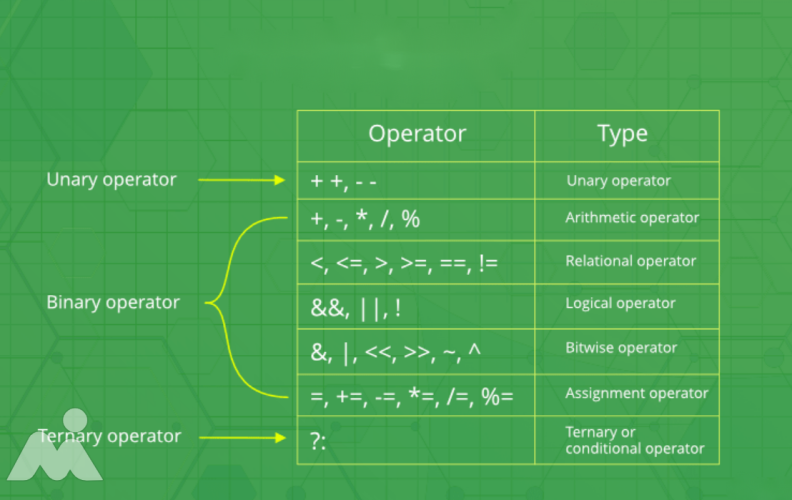
|  |  |
| --- | --- |
| **Table of Contents** | **Page No** |
| **Different Type’s of Language’s** |  |
| 1. **Introduction to Java** |
| 1. **Java Feature’s** |
| 1. **JVM,JDK,JRE** |
| 1. **JDK Version’s** |
| 1. **operator’s** |
| 1. **Condition Statement’s** |  |
| 1. **Iteration Statement’s** |
| 1. **Jump Statement’s** |
| 1. **Data Type’s** |
| 1. **Nested Loop’s** |  |
| 1. **Array’s** |
| 1. **Package’s** |  |
| 1. **Switch Case Statement’s** |  |
| 1. **enum** |
| 1. **Scanner Class** |  |
| 1. **OOP’s Concepts** |  |
| 1. **Constructor’s** |  |
| 1. **static keyword** |  |
| 1. **final keyword** |  |
| 1. **String’s** |  |
| 1. **StringBuffer** |  |
| 1. **StringBuilder** |  |
| 1. **Difference between String,StringBuffer and StringBuilder** |
| 1. **String Class Performance** |  |
| 1. **Creating our Immutable class** |  |
| 1. **Interfaces-In Depth** |  |
| 1. **Abstract classes** |  |
| 1. **Exception handling** |  |

Different Language's Like C,C++, .net, Java there Intro and Differences between them

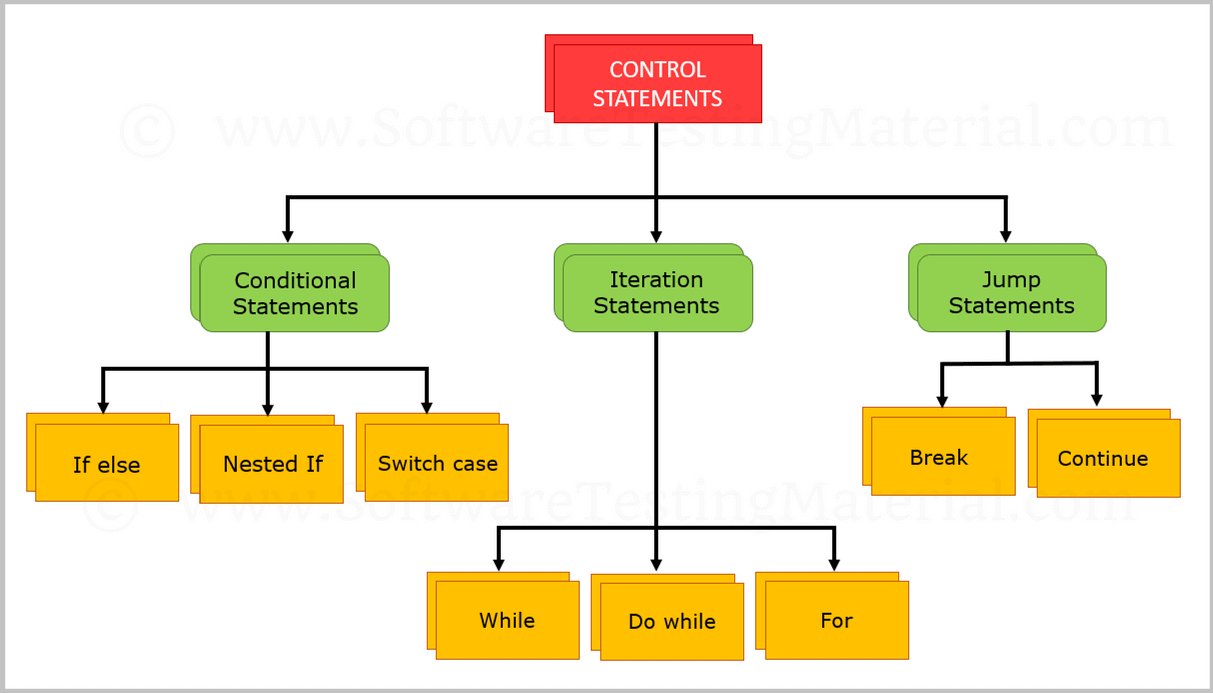
1. **Introduction to Java’**
2. **Java Feature's**
3. Platform Independent
4. Open Source
5. OOPS Concept
6. Garbage Collection(Memory Management)
7. Exception Handling
8. Multi-Threading
9. Secure(Security)
10. **JDK,JRE,JVM**
11. Difference Between JRE and JVM



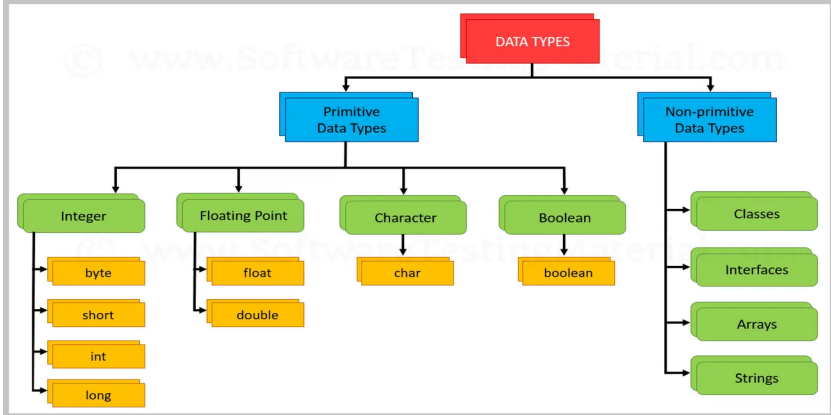
1. **JDK Version’s**
2. 1.0 Release date 23rd January 1996
3. 1.1 Release date 18th February 1997
4. 1.5 or 5(StringBuilder,for-each loop,enum,Annotation)
5. 1.8 or 8
6. **Operator's**
7. Arithmetic Operator’s(**+,-,\*,%,/**)
8. Relational Operator’s**(<,>,<=,>=,==,!=)**
9. Logical Operator’s**( &&,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



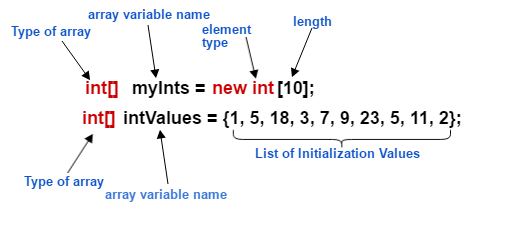
1. **Condition Statements**
2. If
3. If else
4. else if
5. Nested if
6. Switch case
7. **Iteration Statements**
8. While
9. Do while
10. For
11. **Jump Statements**
12. Continue
13. Break



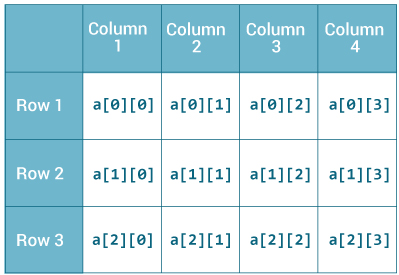
1. **Data Type’s**
2. Primitive
3. byte,Short,Int ,long,float,double
4. Non-Primitive
5. Arrays
6. Strings
7. Interface’s
8. enums..etc



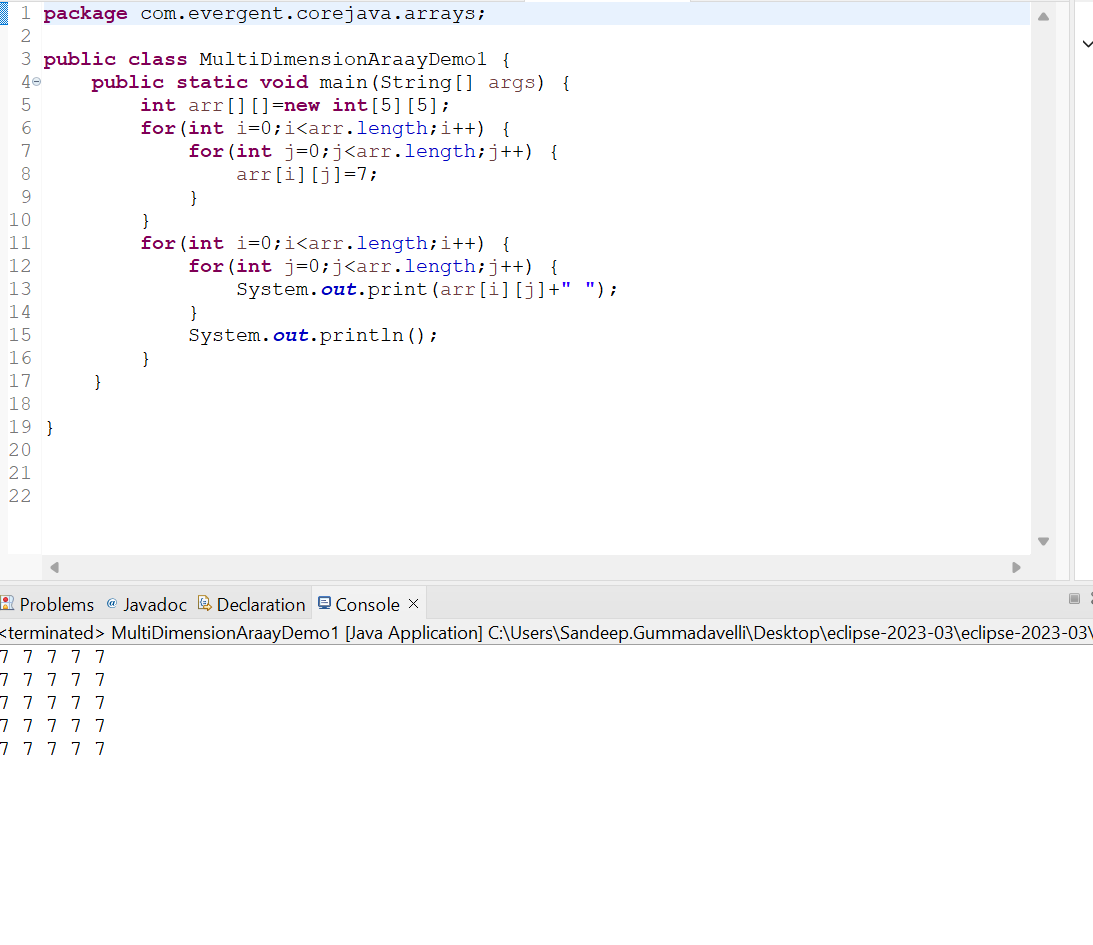
1. **Nested Loop’s**
2. For loop’s
3. **Array’s**
4. Single Dimensional Array’s



1. 2D Array’s



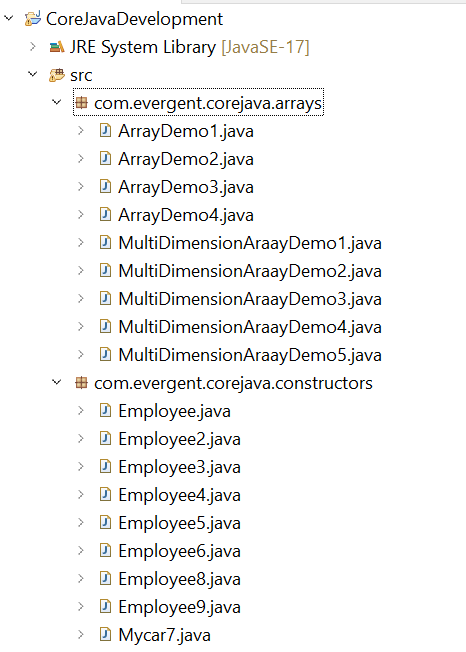
**Example:-**



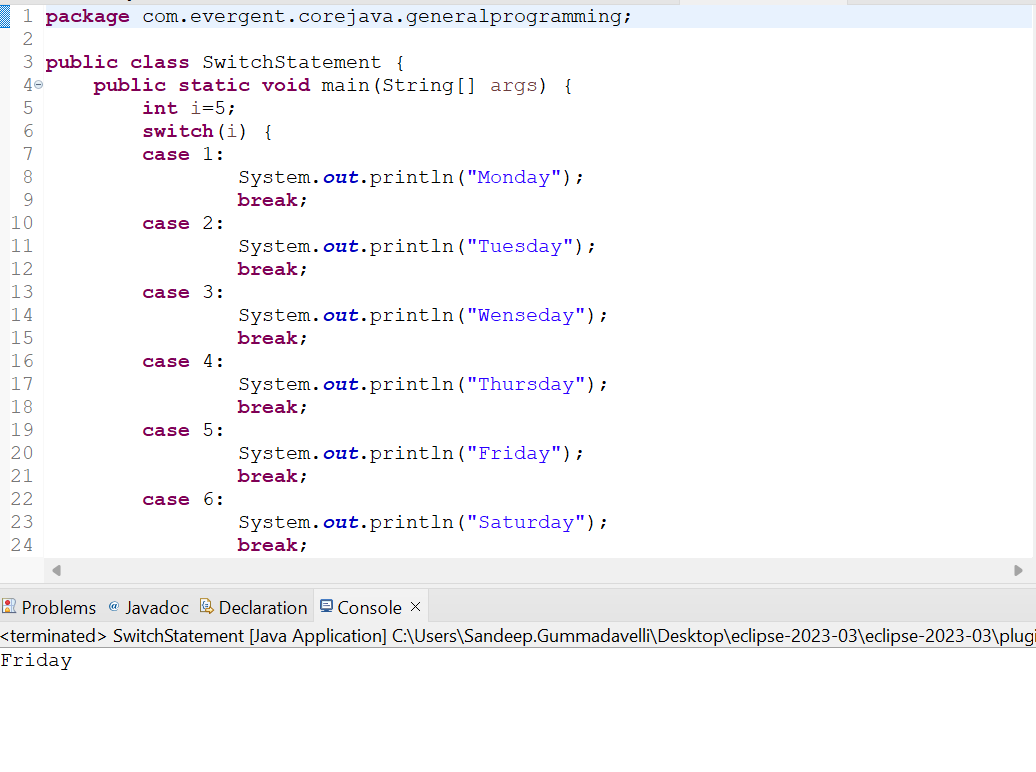
1. **Package’s :-**Packages contains classes and interfaces in java
2. Pre-defined
3. Java.lang;

* System.stream
* String Class
* StringBuffer

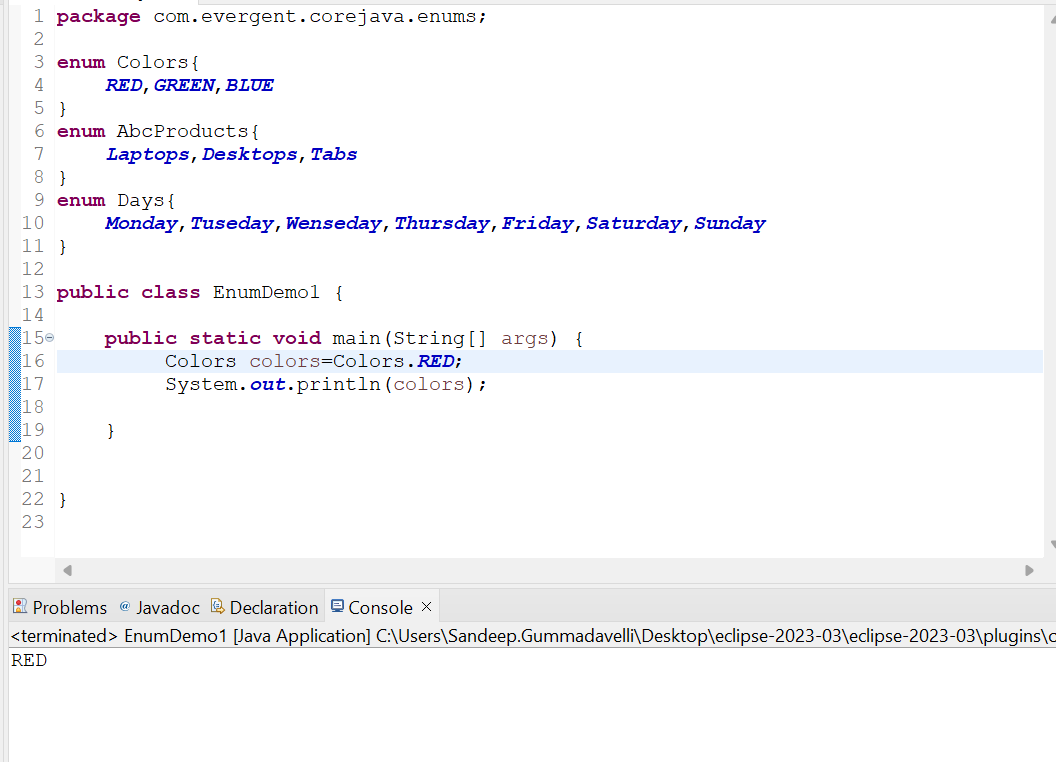
1. StringBuilder
2. Java.util;
3. Java.sql;
4. User-defined
5. The User-defined packages are the those packages which are created by the user explicitly in order to maintain Structure of the Application by placing the similar class into one package
6. Example:-com.evergent.corejava.arrays;



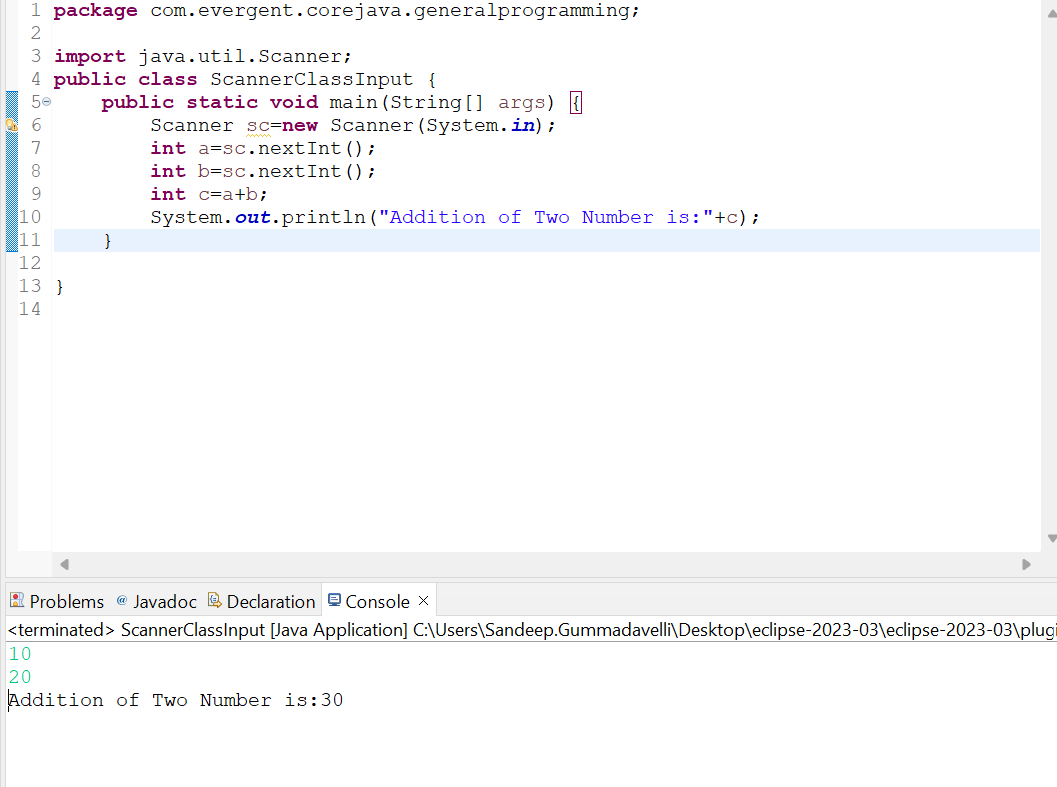
1. **Switch Case Statement:-**Switch statement in java is a muti-way branch statement.



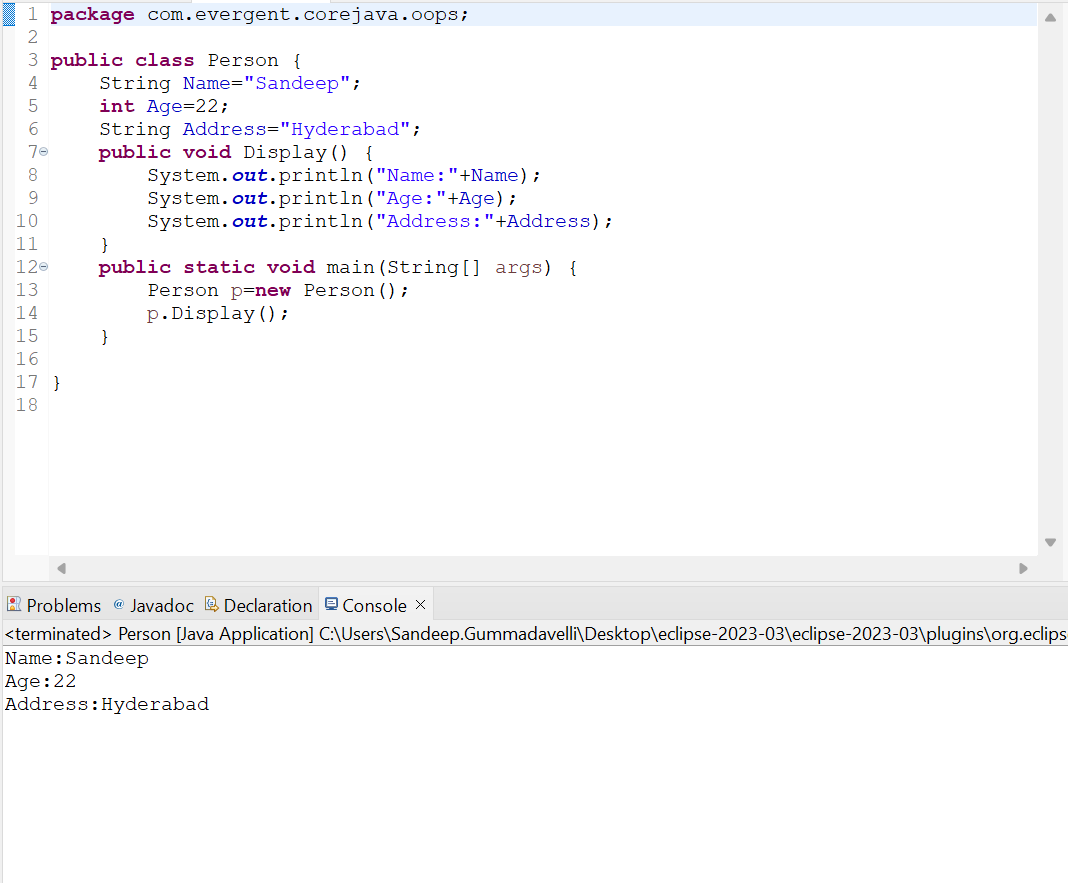
1. **Enum:-**enum is a special class that represents the group of constants



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



1. **Oops Concept’s**
2. **Encapsulation :**Encapsulation is mechanism in binding attributes and methods together in class and object creation is called Encapsulation
3. We can’t access the methods and attributes without creation of object as of now.
4. But we can access the methods and attributes in java through object creation & reference
5. Object Creation
6. **Loosely coupled Example:-**

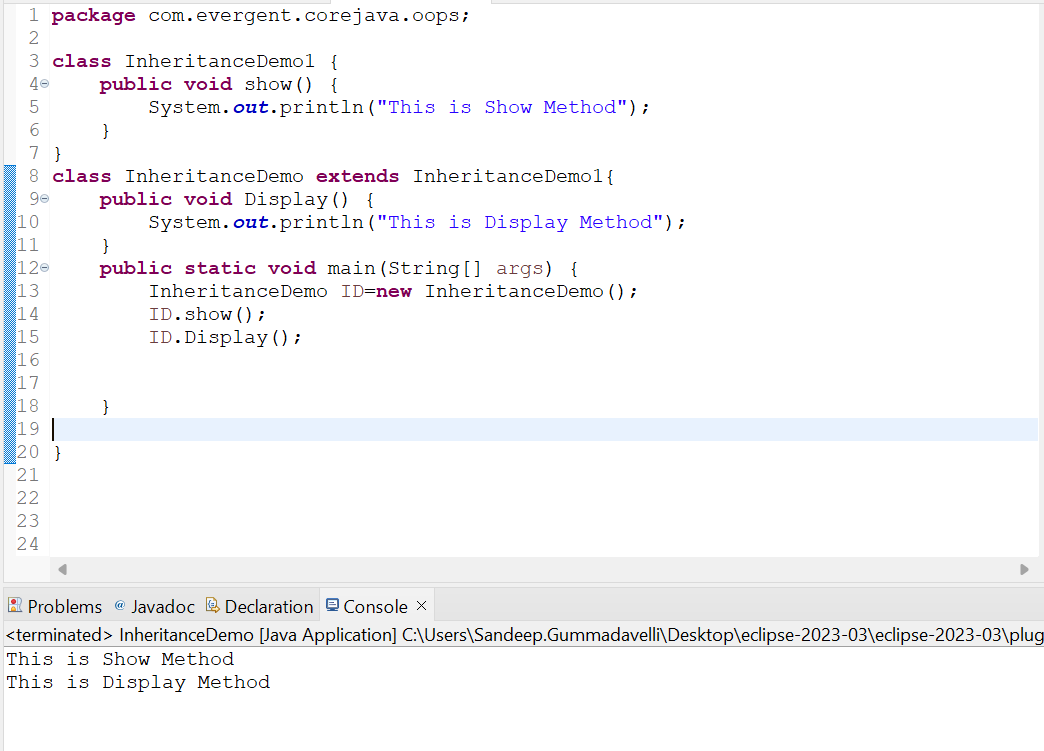


e) Method Types or Method Flows

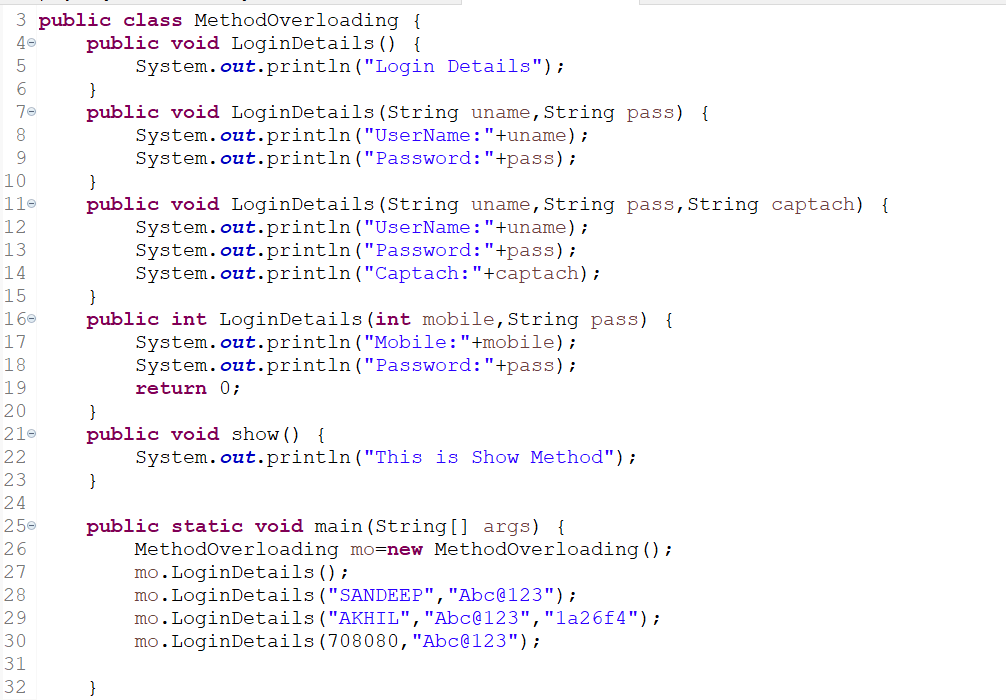
* + - 1. No Parameters with no return type
      2. Parameters with no return type
      3. Parameters with return type
      4. No Parameters with return type



1. **Inheritance :** Resuability of existing functionalities from super class to sub-class is called as inheritance
2. Java won’t support multiple inheritance through classes(Ambiguous)
3. Java will support multiple inheritance through interfaces

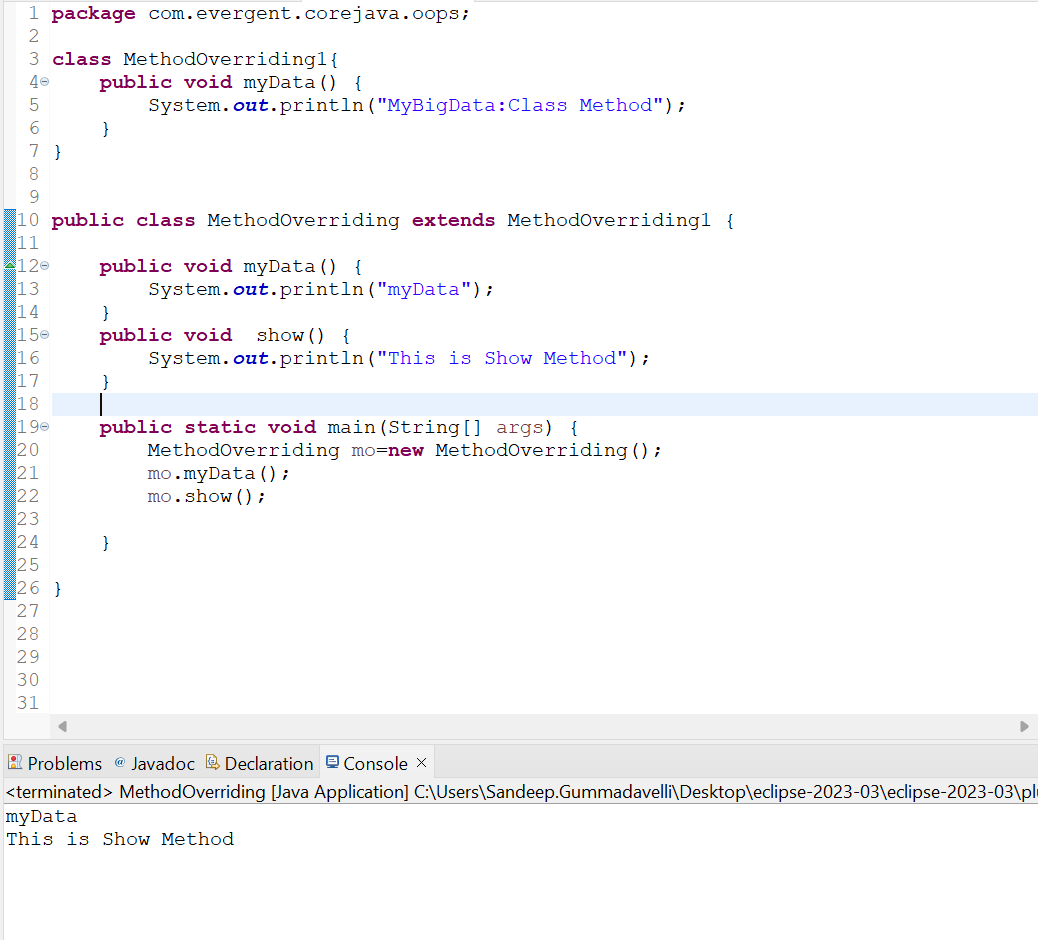


1. **Polymorphism:**many techniques or single action is going to performed in multiple ways
2. **Method Overloading:**Overloading means methods names are same parameters should be different and return type may or may not be same and it will happen in the same class or different class

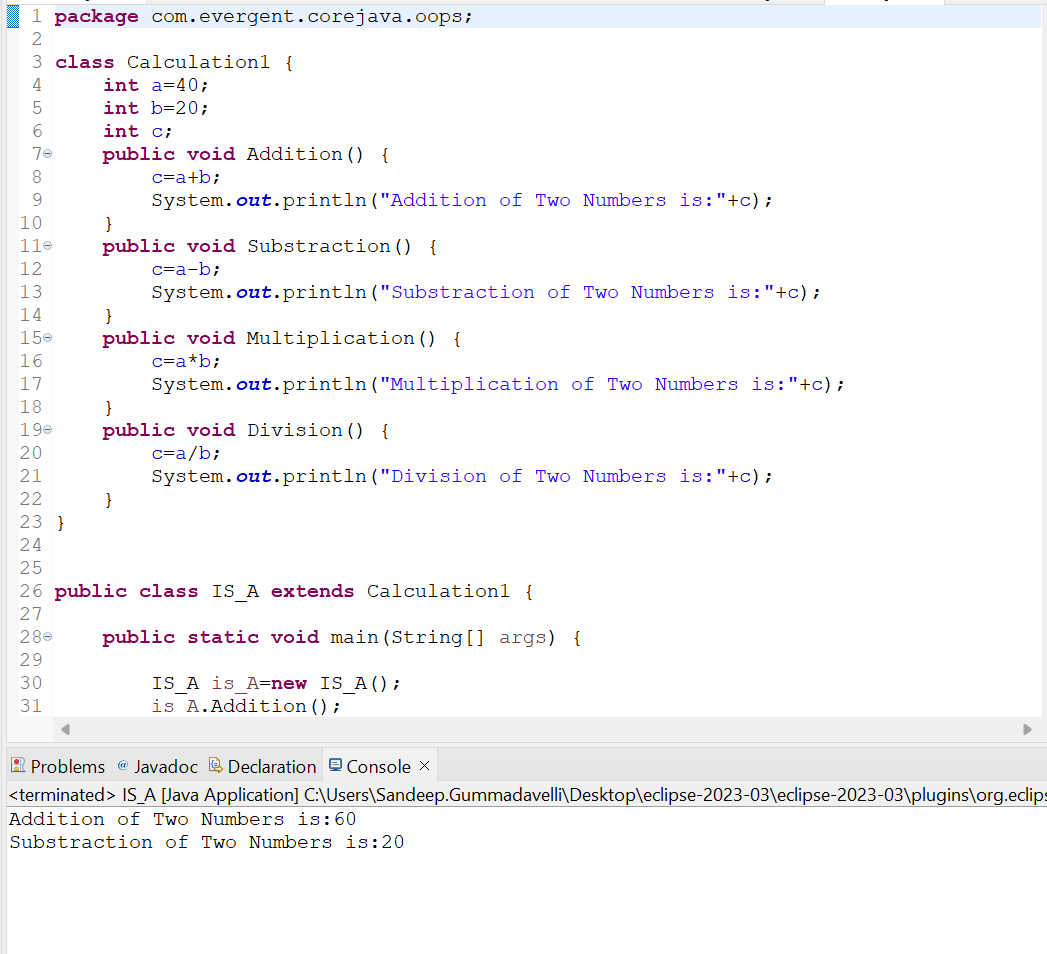


1. **Method Overriding:**Methods are same and parameters also same and it will happen in tow different classes through inheritance or Interface

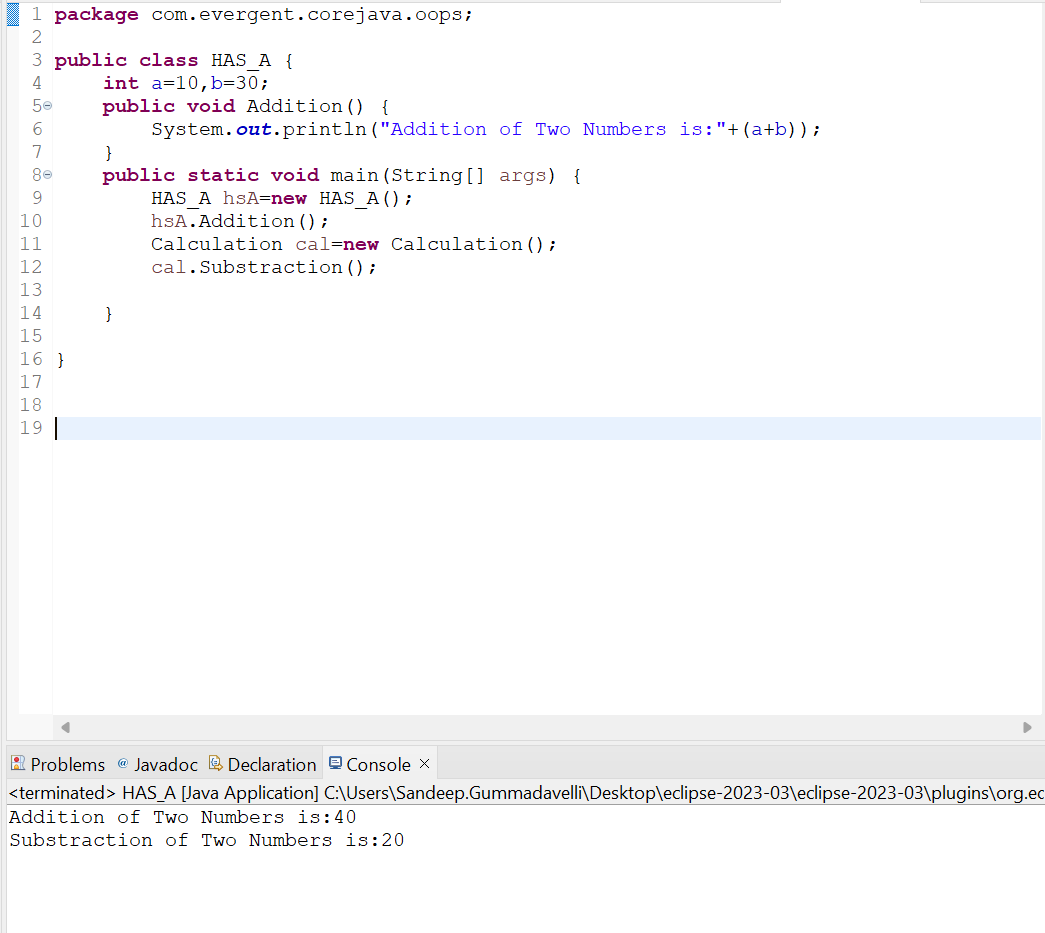
**Best Example for overriding is interfaces**



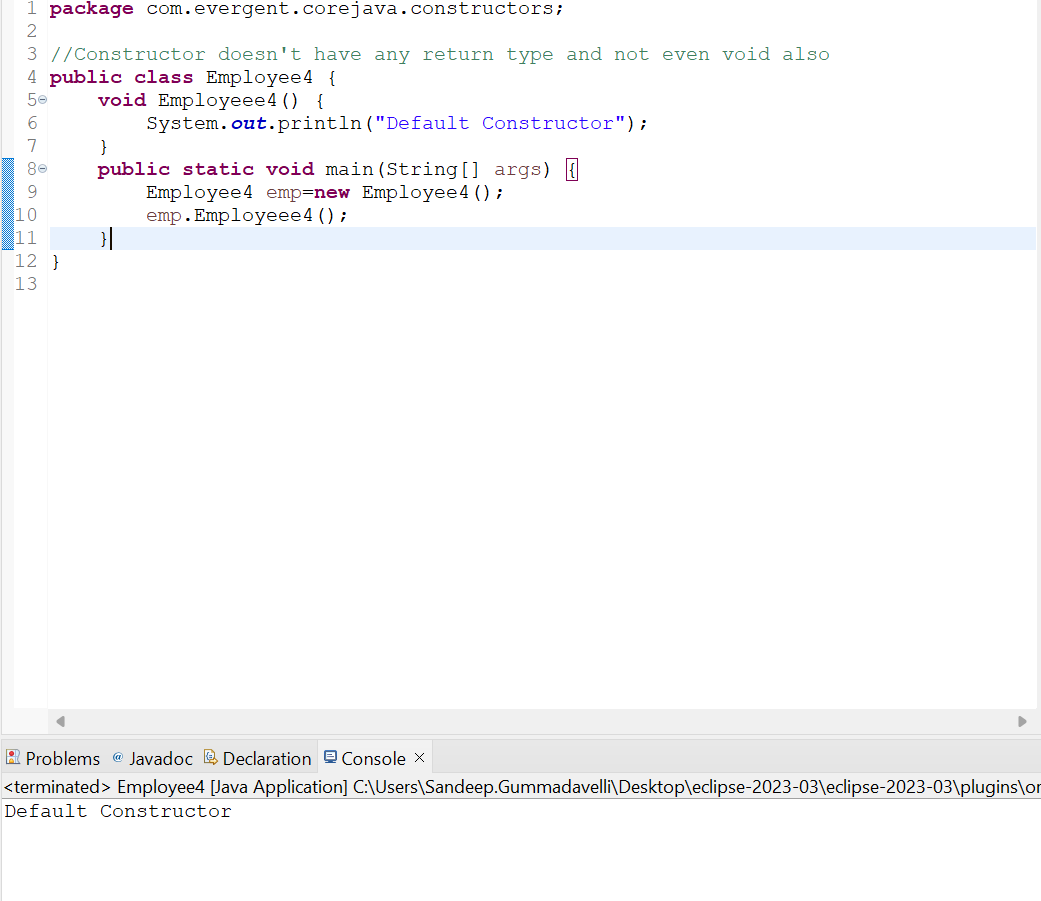
1. **Abstraction:**Showing only the necessary details and hiding the unwanted details from the user
2. **IS-A:**Used in Inheritance



1. **HAS-A:**Used to access only particular methods in another class by creating the object of that class and accessing the method and it can be accessed inside the packages or other packages(through object creation)



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
8. Constructors are mainly for initialization
9. Constructors doesn’t have any return type and void
10. If we Declare as a void it will consider as normal method not a constructor
11. Every class need’s at least on default constructor either the user will create or default constructor
12. Always Constructors are Overloaded
13. We can call one constructor in another constructor



**Program:** The Class name and the Constructor name should be same

We can access the constructor while creation of object



**Program: -** Constructors are mainly for initialization

- Constructors doesn’t have any return type and void

- If we Declare as a void it will consider as normal method not a constructor

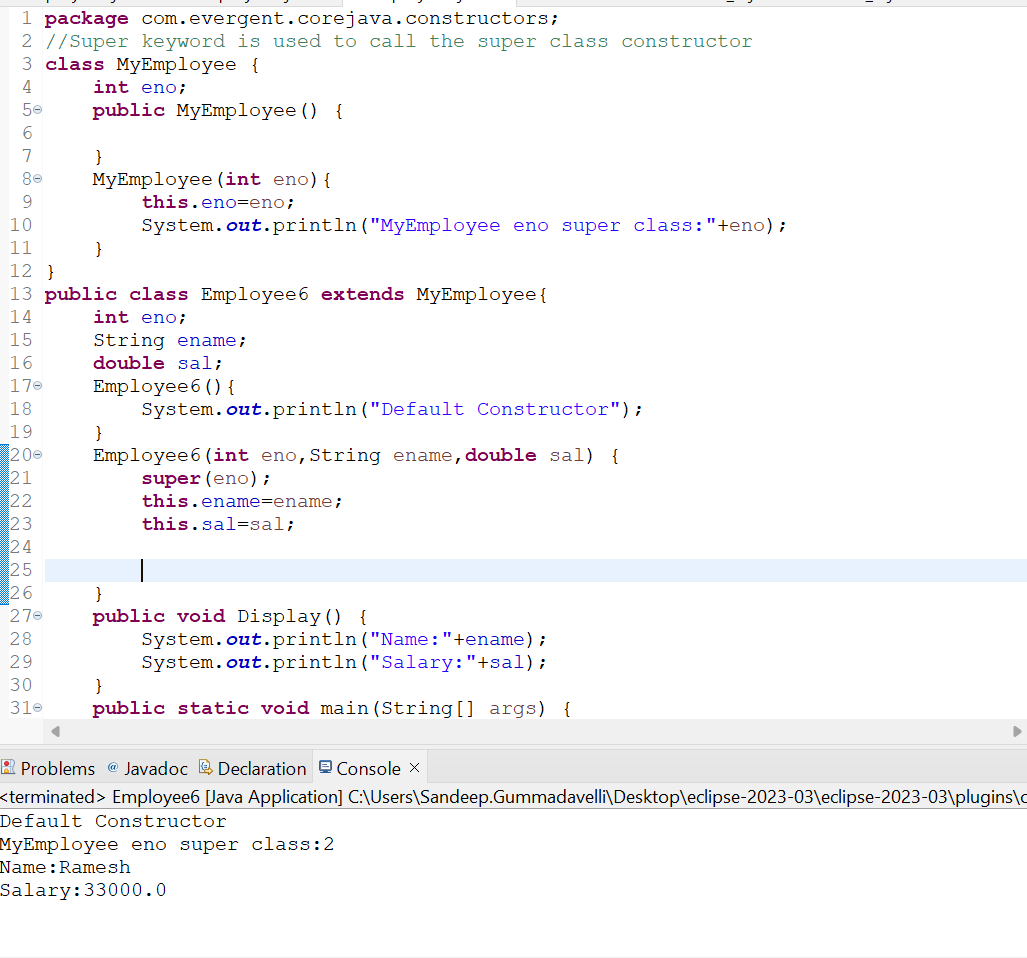


**Program: -** Every class need’s at least on default constructor either the user will create or default constructor

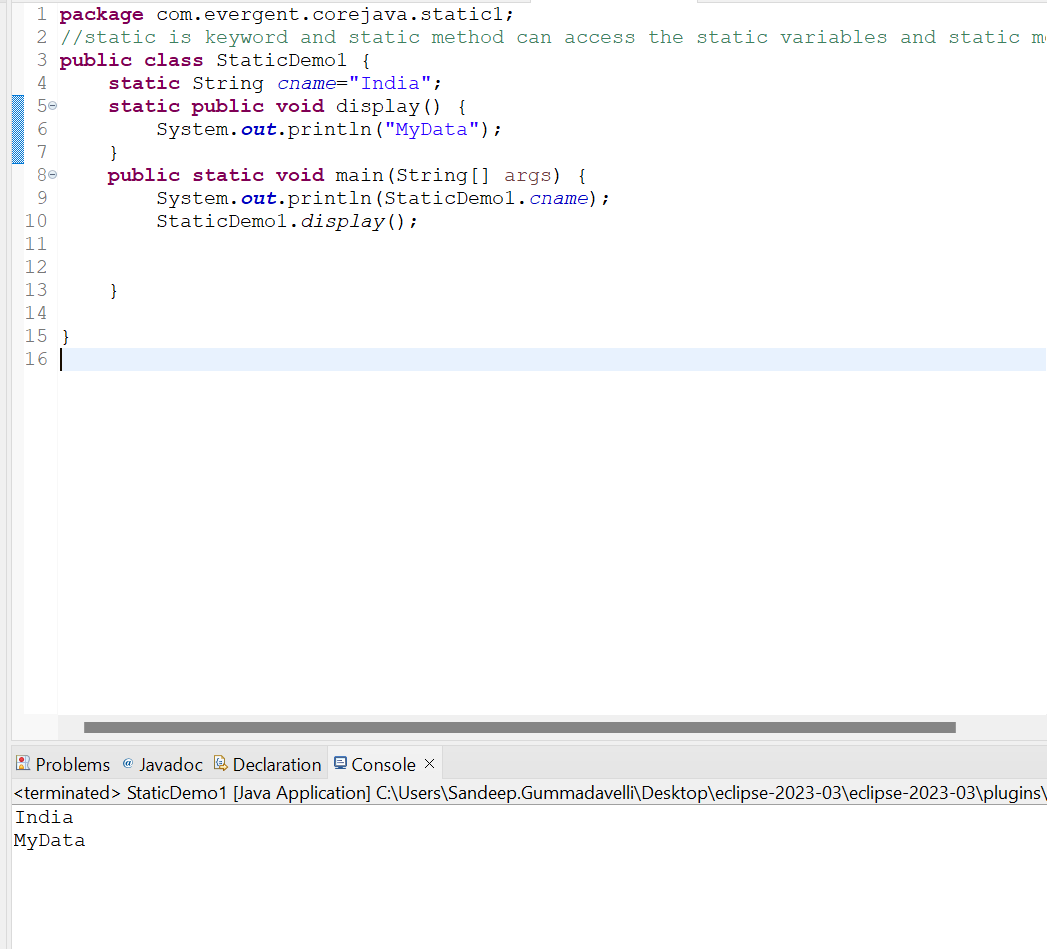
- Always Constructors are Overloaded

- We can call one constructor in another constructor

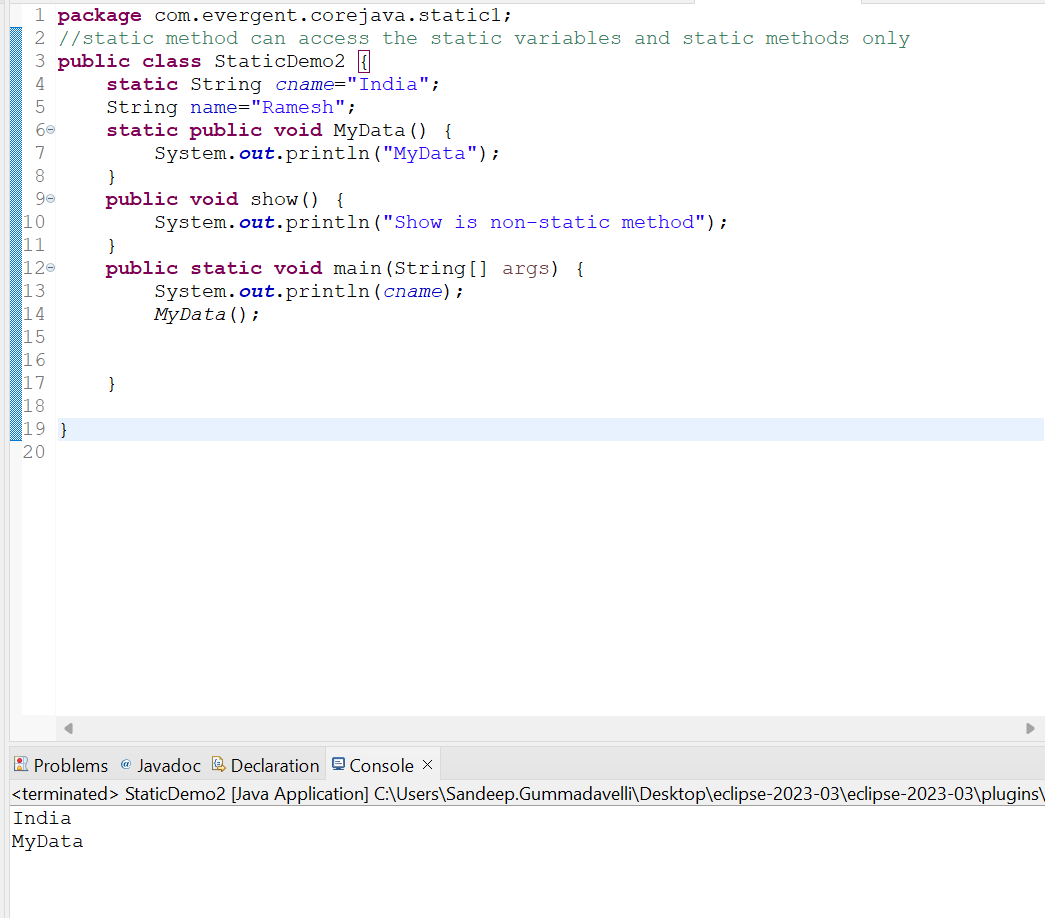
* **This Keyword:-**This is a keyword used to refer to the current instance variables and this keyword is used to invoke the one constructor to the other constructor in the same class
* **Super Keyword:-**Super is a keyword which is used to call the super class constructor from the base class constructors



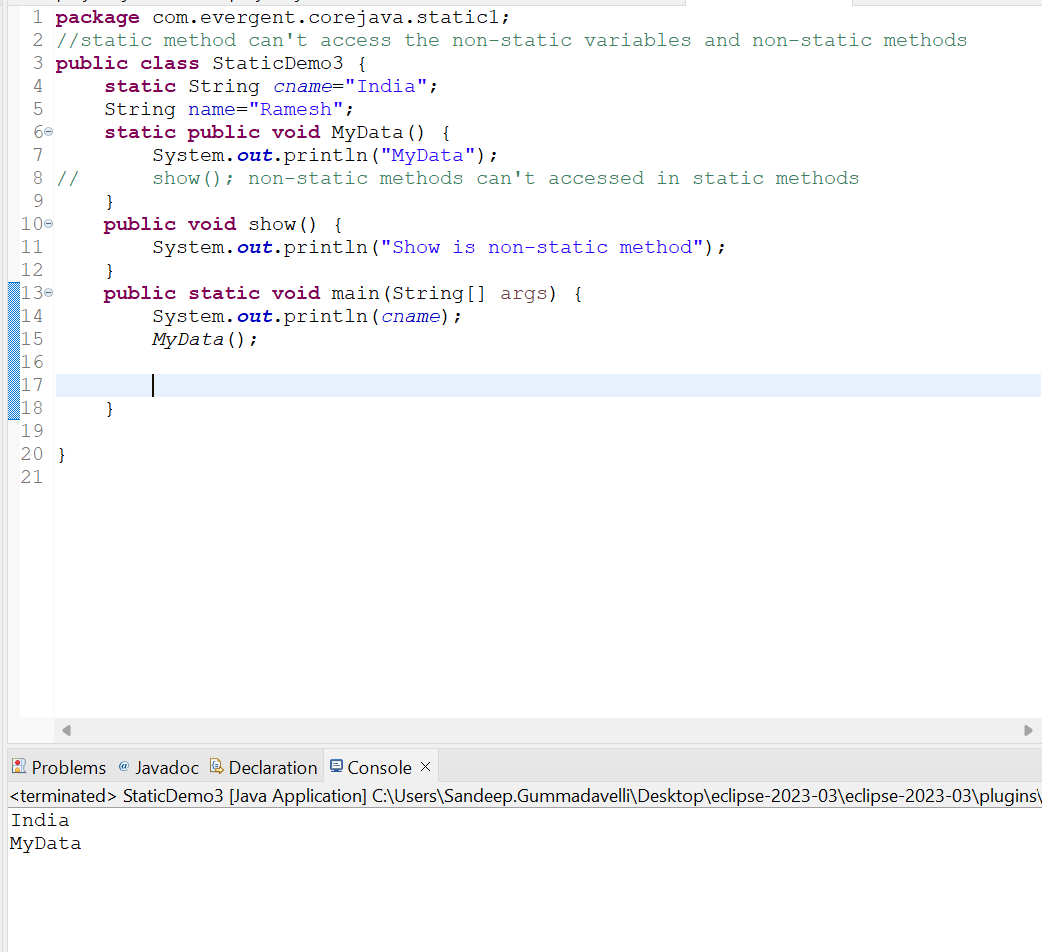
1. **static Keyword In Depth**
2. static is keyword
3. We can declare the static as variables and methods
4. We can access the static methods and variables directly through className..method or className.variable
5. static methods can access the static variables and static methods
6. static methods can’t access non-static methods and variables
7. Non-static method can access the static methods and static variables
8. Static blocks get initialized whenever the class loaded into the JVM



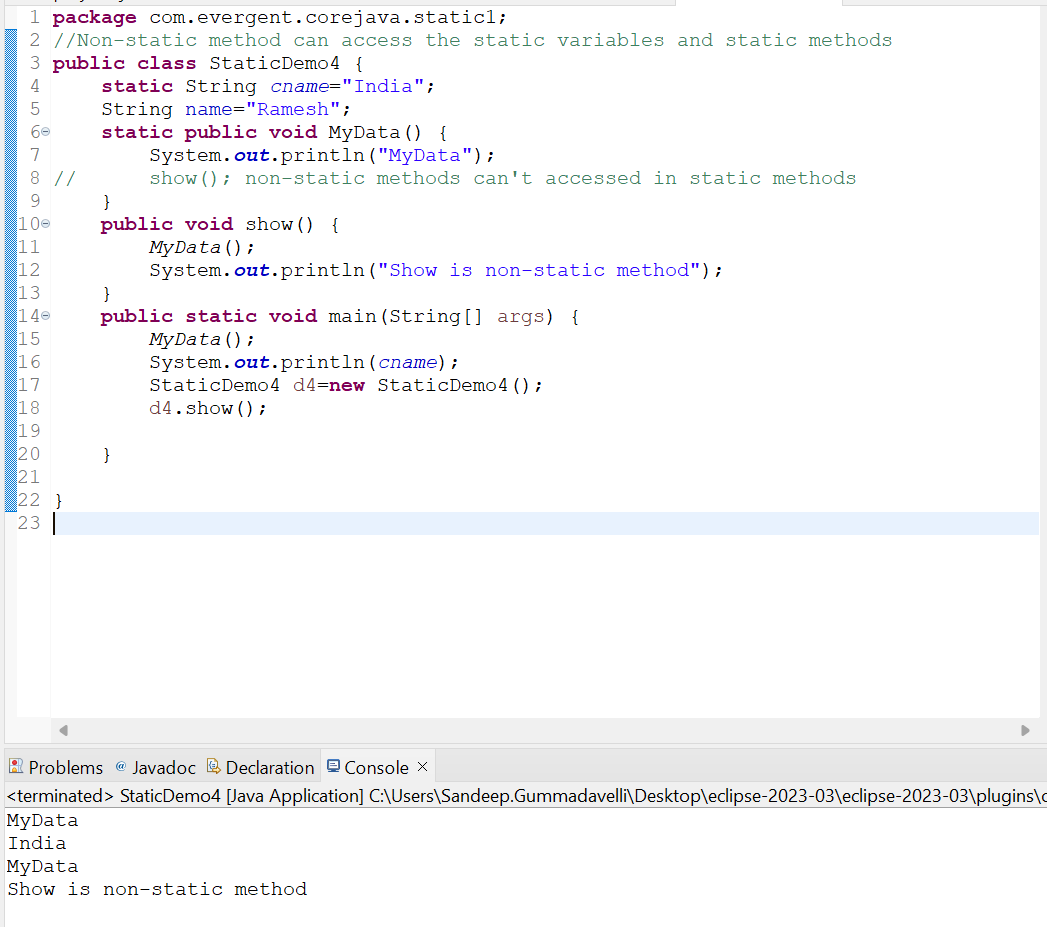
**Program:** static methods can access the static variables and static methods



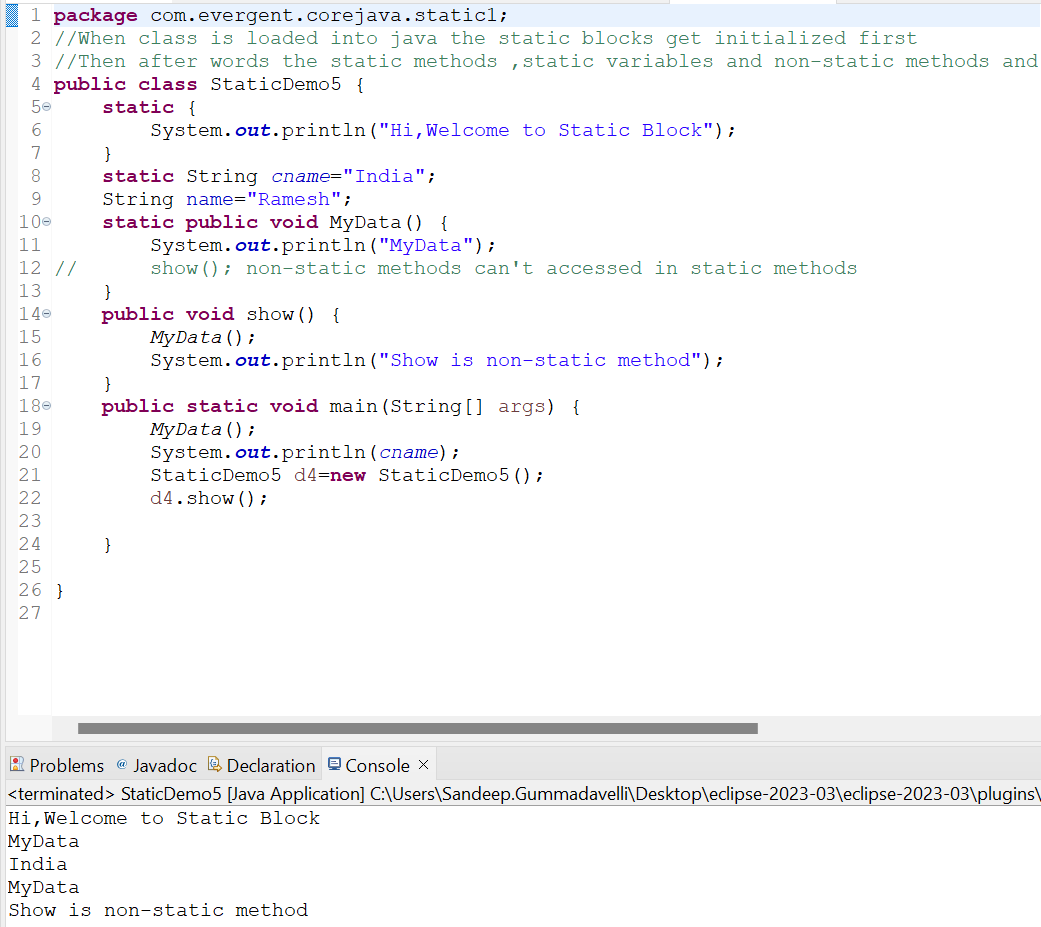
**Program:** static methods can’t access non-static methods and variables



**Program:** Non-static method can access the static methods and static variables

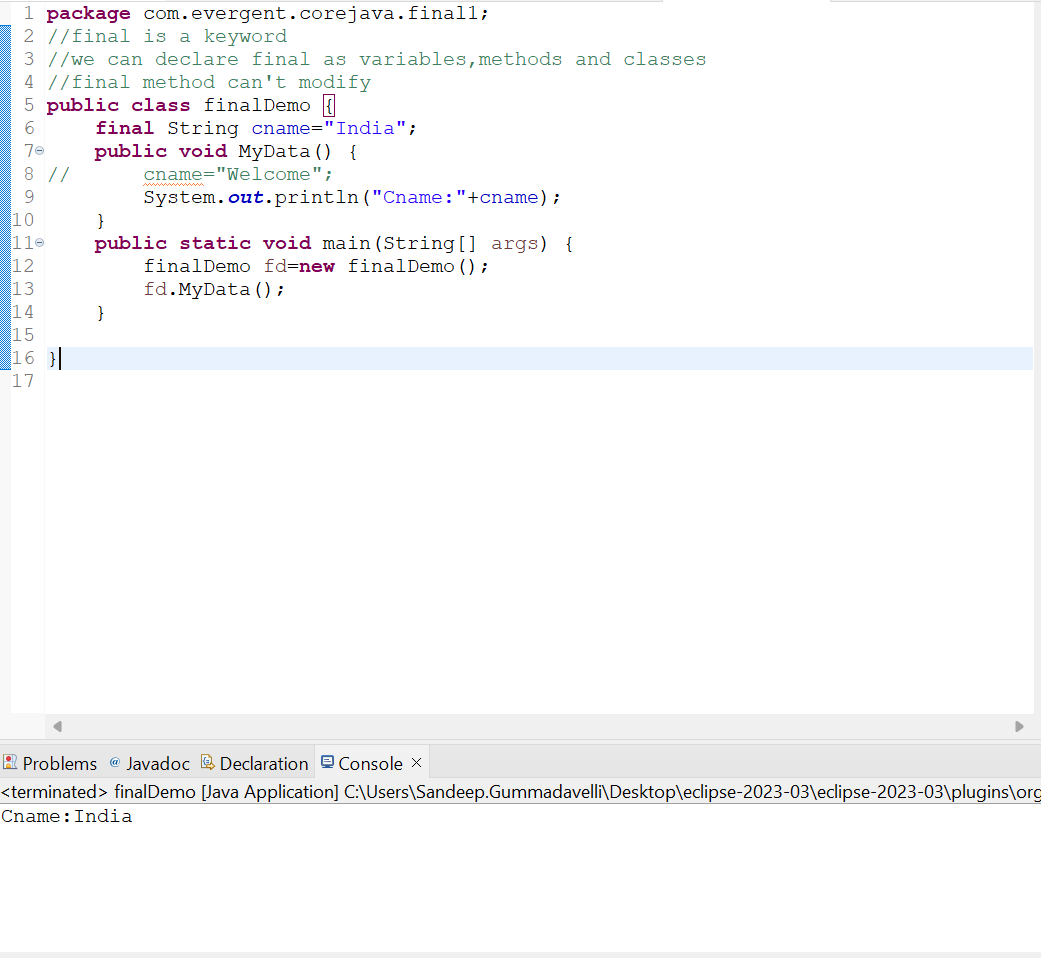


**Progarm:** static block:-When class is loaded in to JVM static block get initialized first



**Program:** static blocks get initialized first whenever the class is loaded into in the JVM

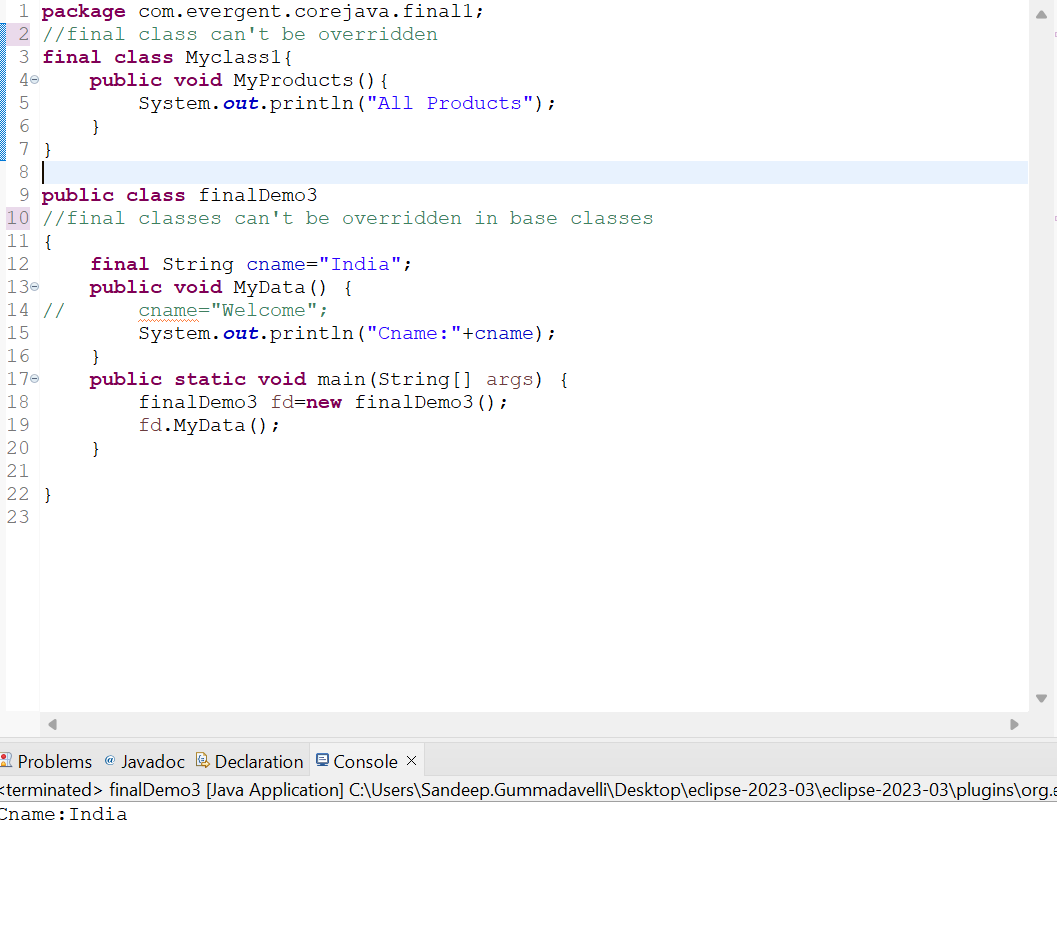
1. **final Keyword in Depth**
2. final is a keyword
3. We can declare final as variables,methods,classes
4. final variables we can’t modify
5. final method can’t be overridden
6. final class can’t be inherited



**Program:** final variables we can’t modify



**Program:** final method can’t be overridden



**Program:** final class can’t be inherited

1. **Strings**
2. String class is final class
3. String are immutable
4. String methods are non-synchronize

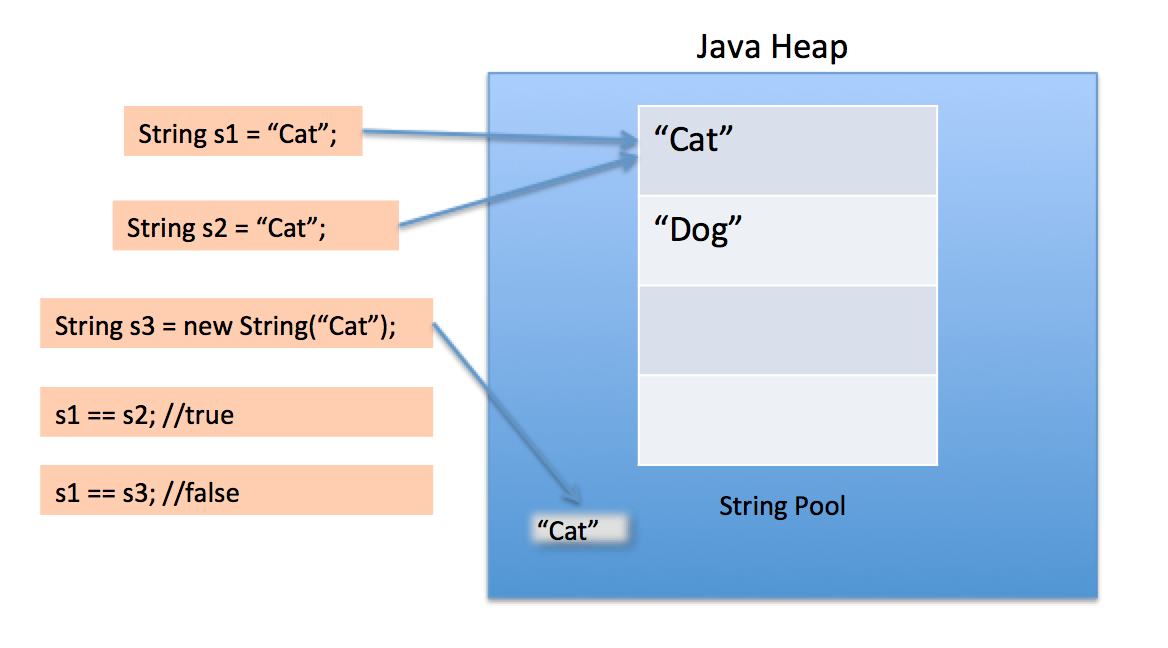
**String class Other Imp Points**

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
3. Example:-

String name=new String(“Hello”);

Name=”Welcome”;

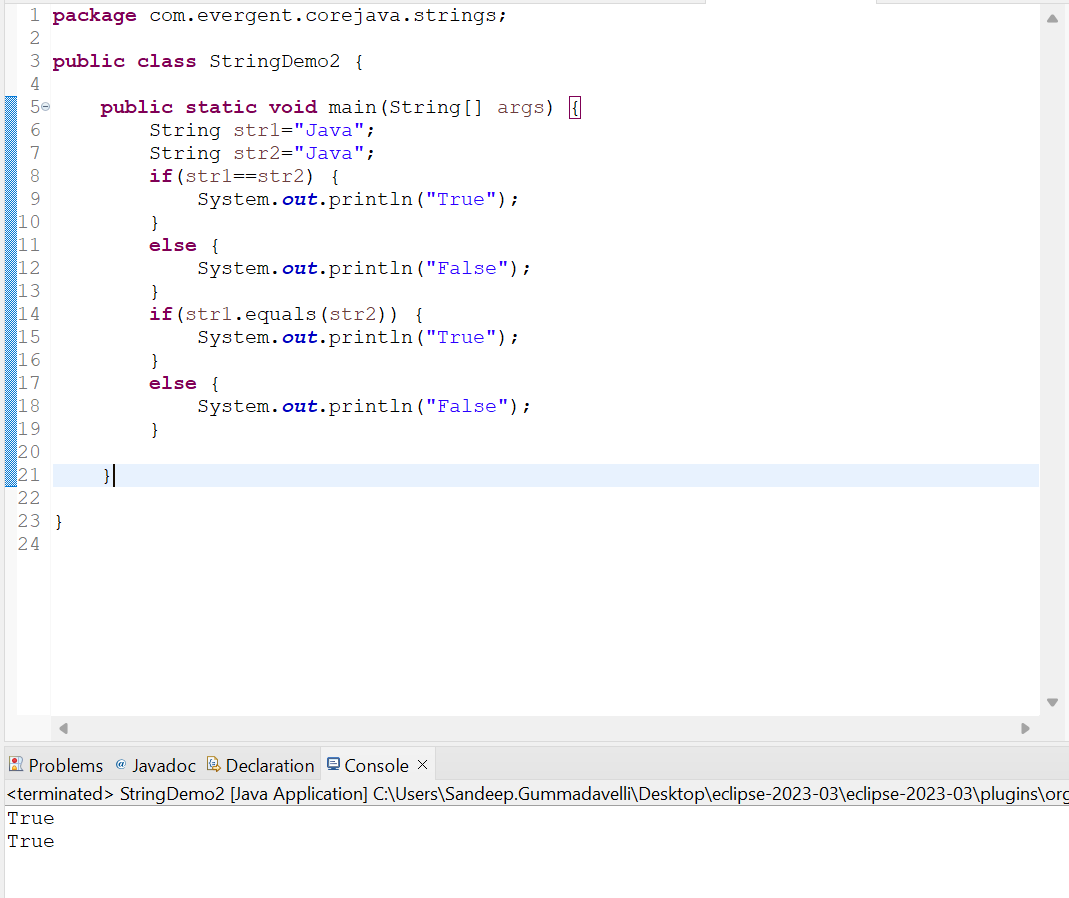
1. String objects in java and are instances of the string class ,which is part of the java.lang.package
2. Key Features of strings in java :
3. Immutable:-Once a string object is created it can’t be modified
4. Any modifications to a string creates a new string object
5. Java Optimizes memory usage by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



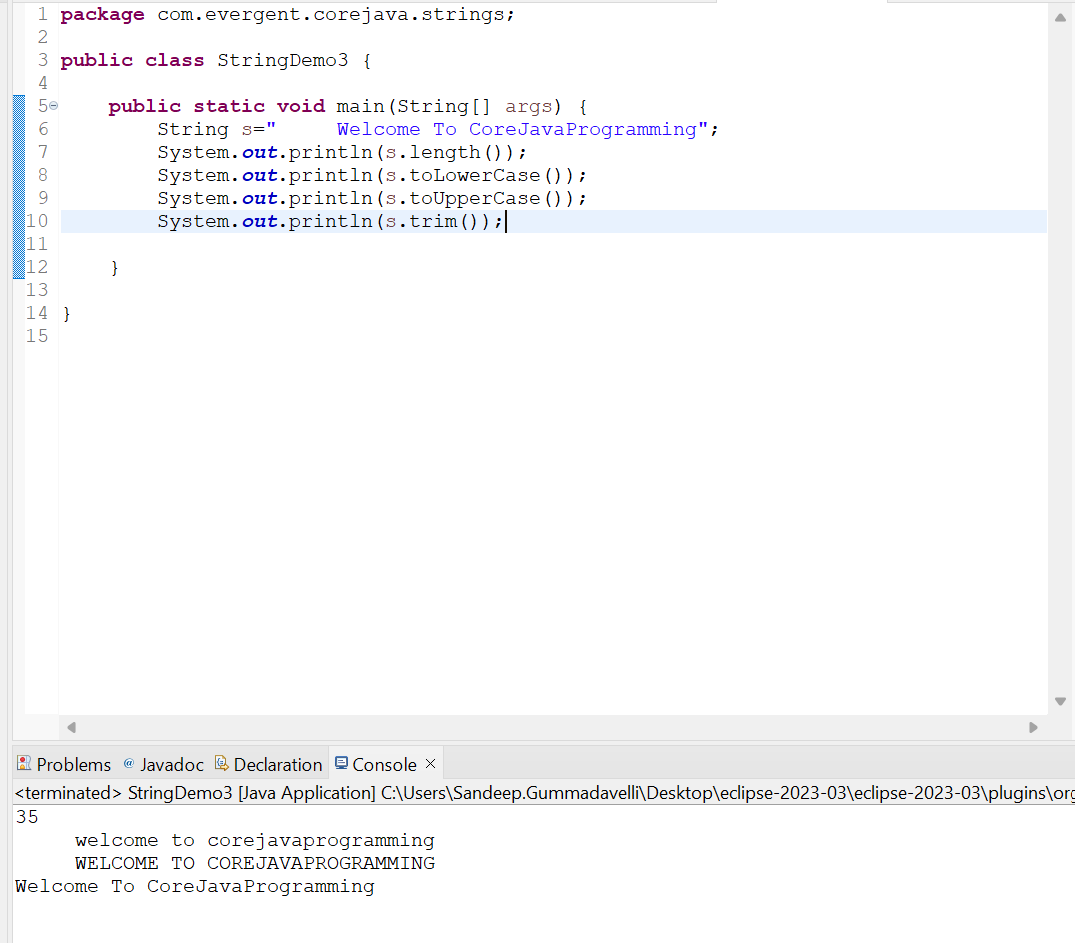
1. We can Declare the Strings in two ways I,e
2. Using new Keyword like String str=new String(“Hello”); **---Heap memory storage**
3. And the other way is like normal way of declaration like String str=”Hello”;**- String pool**
4. Different types of String methods include:
5. Length();
6. toLowerCase();
7. toUpperCase();
8. trim();
9. startsWith();
10. endsWith();
11. equals();
12. equalsIgnoreCase();
13. toCharArray();
14. indexOf();
15. lastIndexOf();
16. subString();
17. split();
18. replace();
19. isEmpty();
20. valueOf();
21. isBlank();



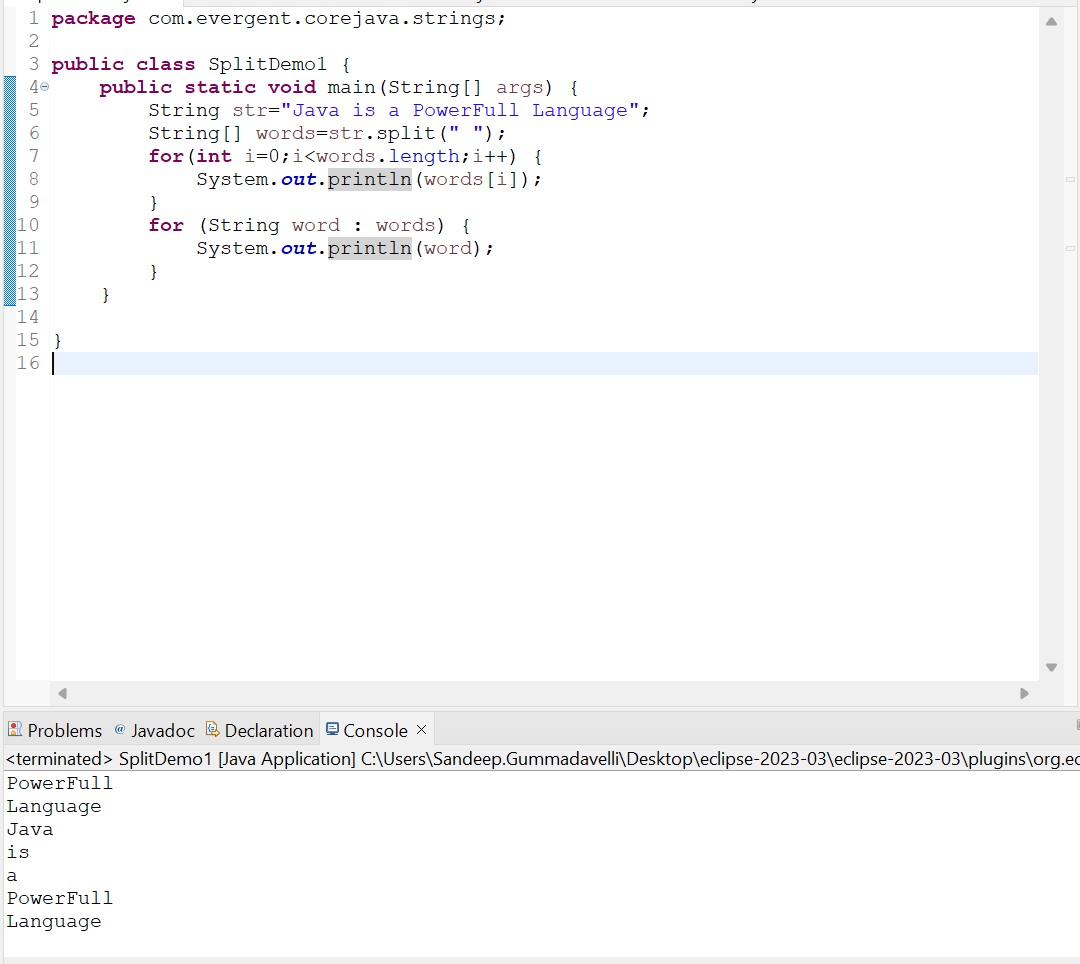
**Program: String Program using new Keyword**



**Program: String Program using normal declaration and assigning values**

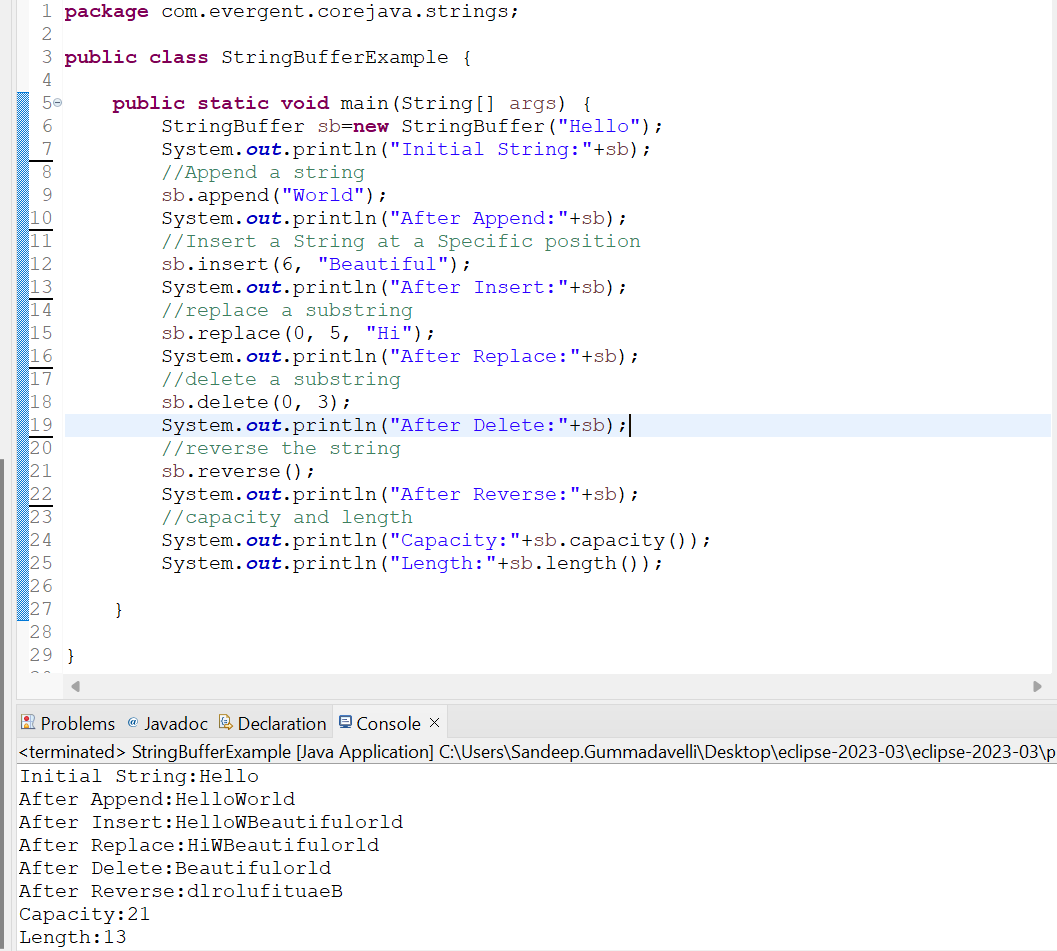


**Figure: String class Methods Program**



**Program: String Split () method Program**

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

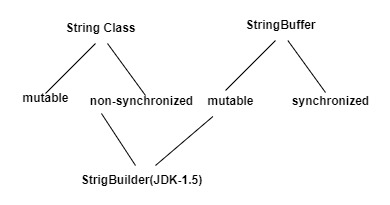


1. **StringBuilder:-**In Java,StringBuffer is a final class and mutable and it has some methods include
2. Append:-adds a string to the end of the current string
3. Insert:-adds a string at a specific position in the current string
4. Replace:-replaces a substiring with new string
5. Delete:-removes s substring from the current string
6. 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() |
|  | 5.It is not recommend to use ,it is in legacy api | 5.It is introudced In JDK 1.5 |



1. **String class Performance**



**Program:** String Performance

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

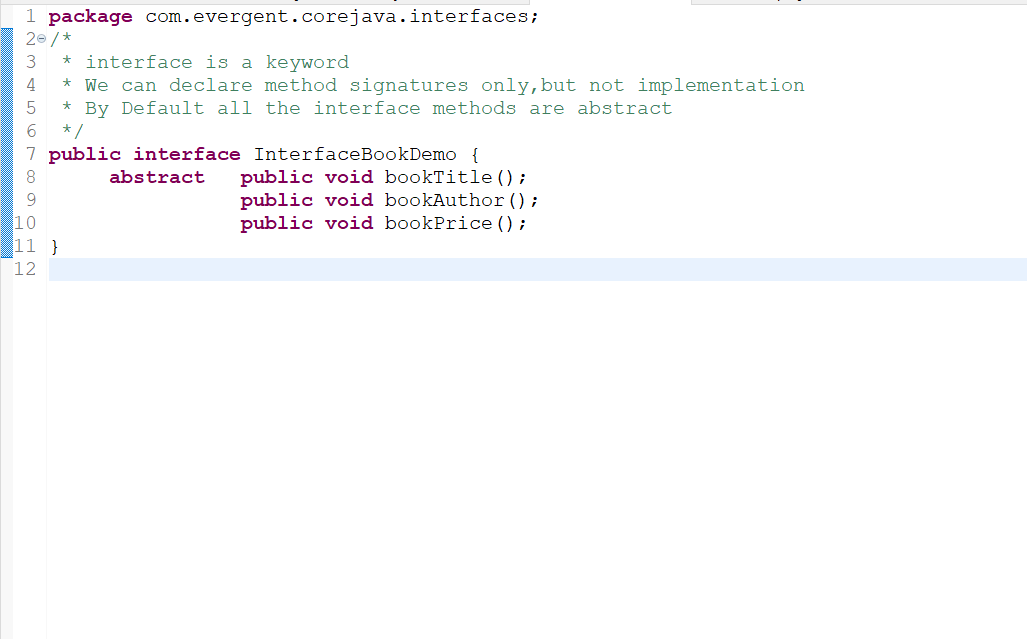
**Constructor:-**

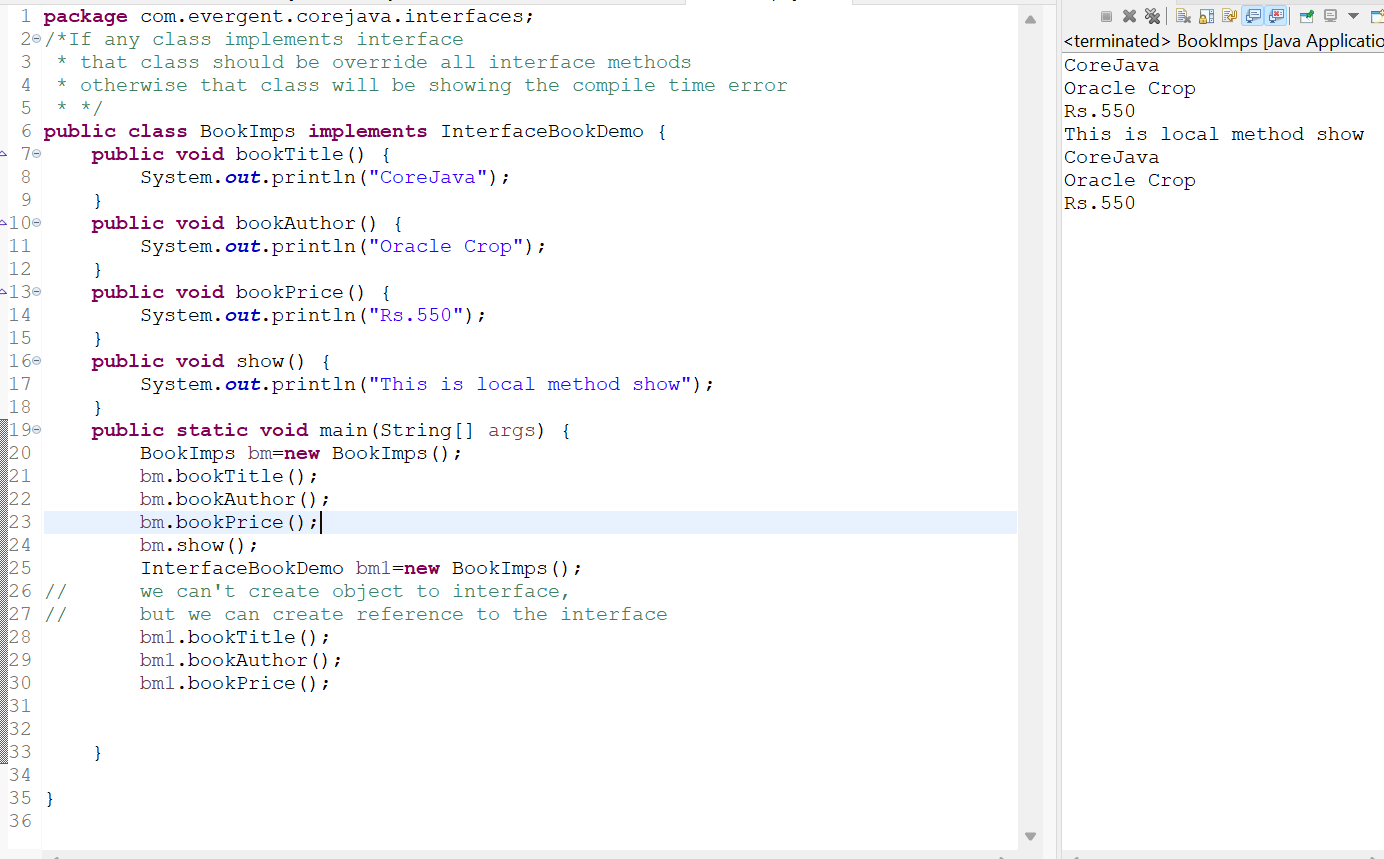
- The constructor initializes the final fields when a person object is created





1. **Interface’s -In Depth**
2. Interfaces is keyword
3. We can declare methods signature only not implementation
4. By Default all interface methods are abstract
5. If any class implements interface that class should be override all interface methods otherwise that class will be shown with compile time error
6. We can’t create object to interface,but we can create reference to the interface
7. We can declare variables and by default the variables are default final
8. Java will support multiple inheritance through interface
9. One class can implements interfaces
10. One interface can extend other interfaces
11. Interface’s doesn’t have any methods or we can create interface without methods or zero methods
12. Marker interfaces:-
13. Clonable
14. Serializable interface
15. Remote interface

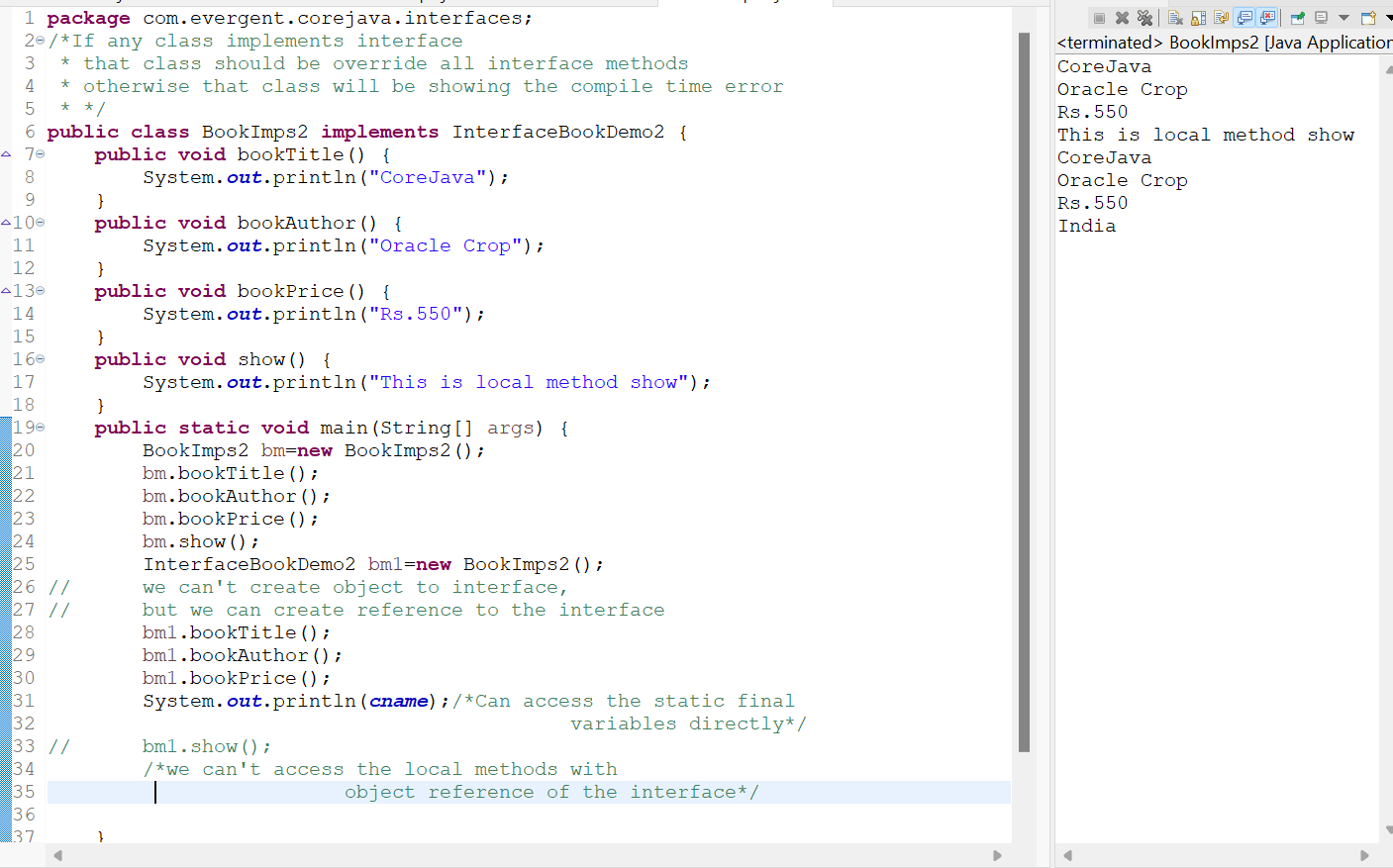




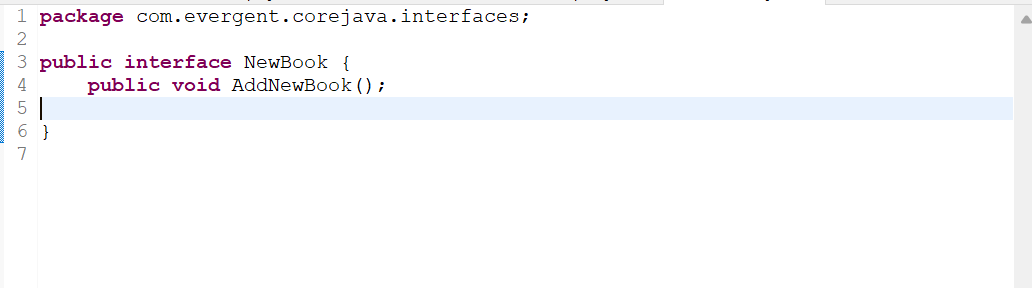
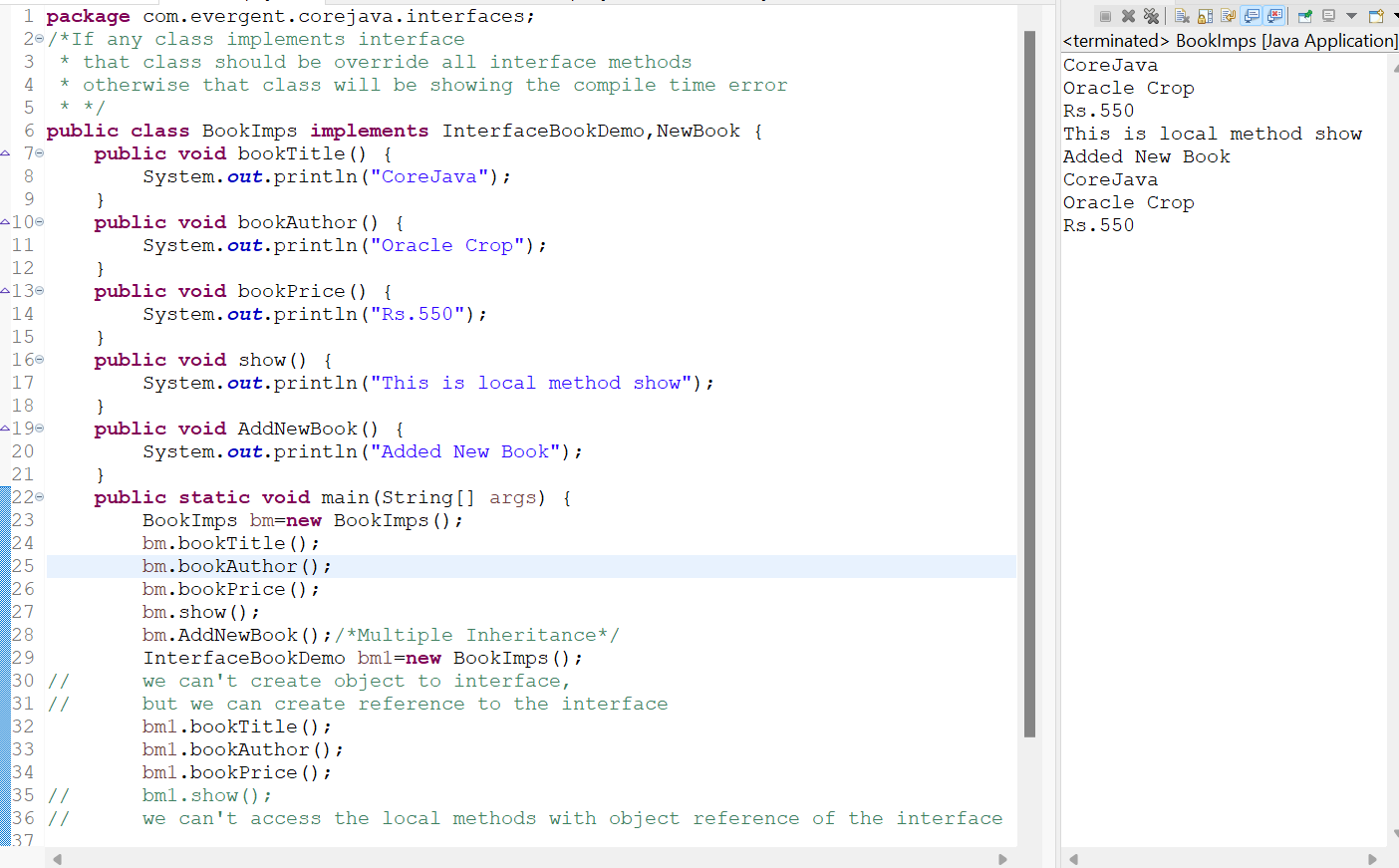
**we can't access the local methods with object reference of the interface**



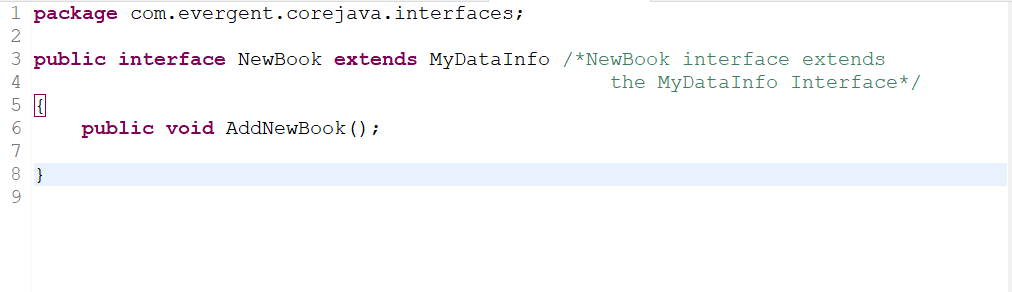
**Program:** We can declare variables and by default the variables are default final



**Program:** Java will support multiple inheritance through interface



**Program:** One class can implements interfaces

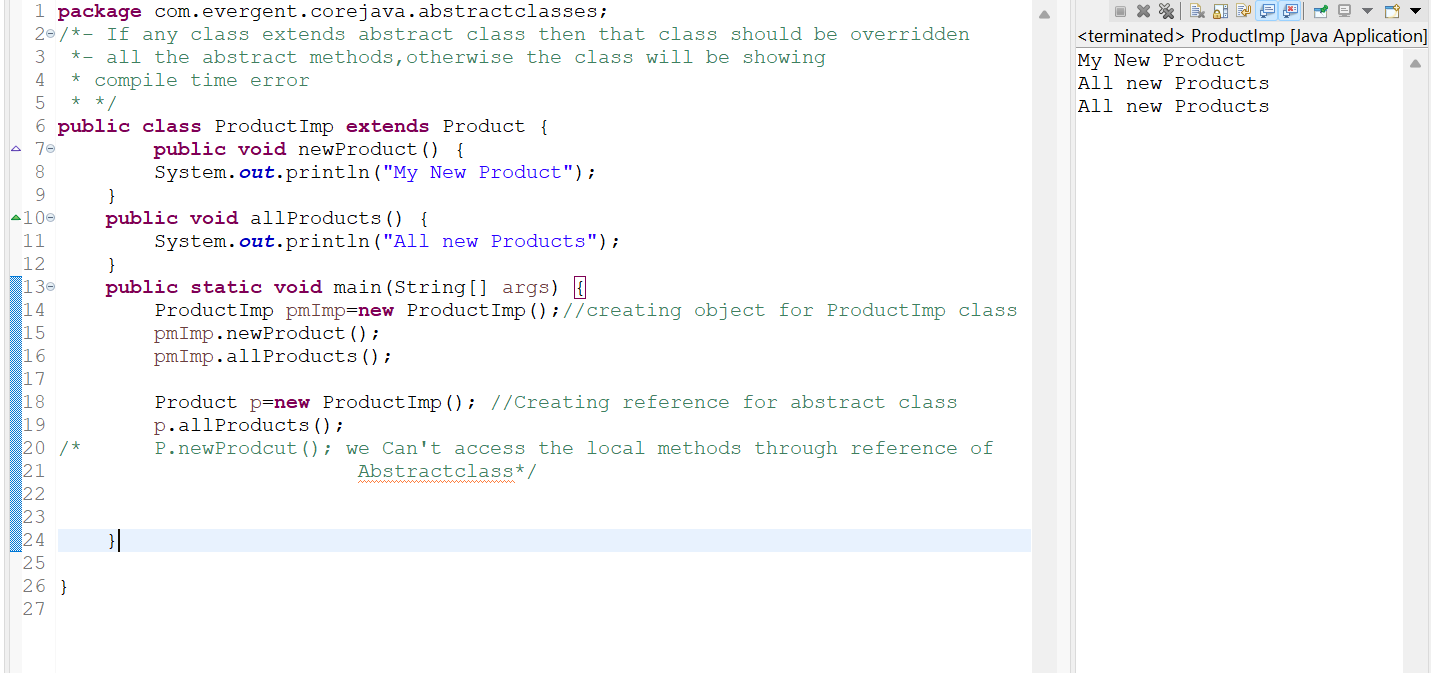


**Program:** One interface can extend other interfaces

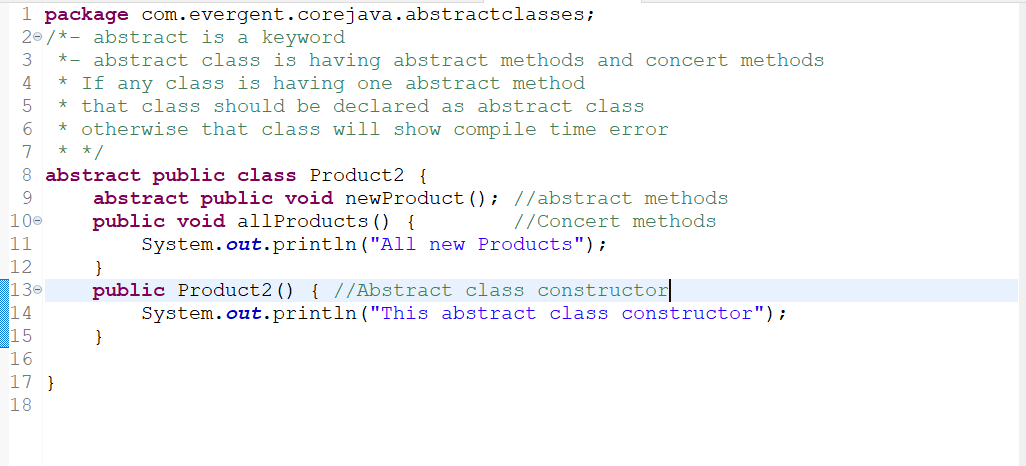
1. **Abstract Classes -In Depth**
2. abstract is keyword
3. Abstract class is having abstract methods and concert methods
4. If any class is having one abstract method that class should be declared as a abstract class or that class will be showing compile time error
5. If any class extends abstract class then that class should be overridden all abstract methods,otherwise the class will be showing compile time error
6. We can’t create object for abstract class but we can create reference to the abstract class
7. We can access the abstract class constructor through subclass object creation

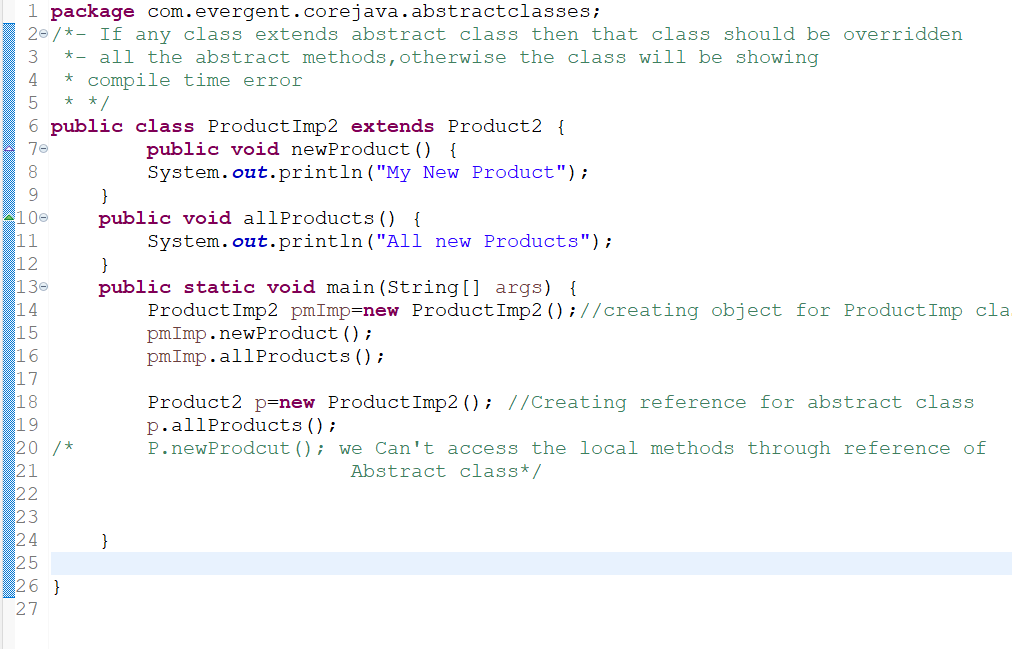


**Program:** abstract class **Product**



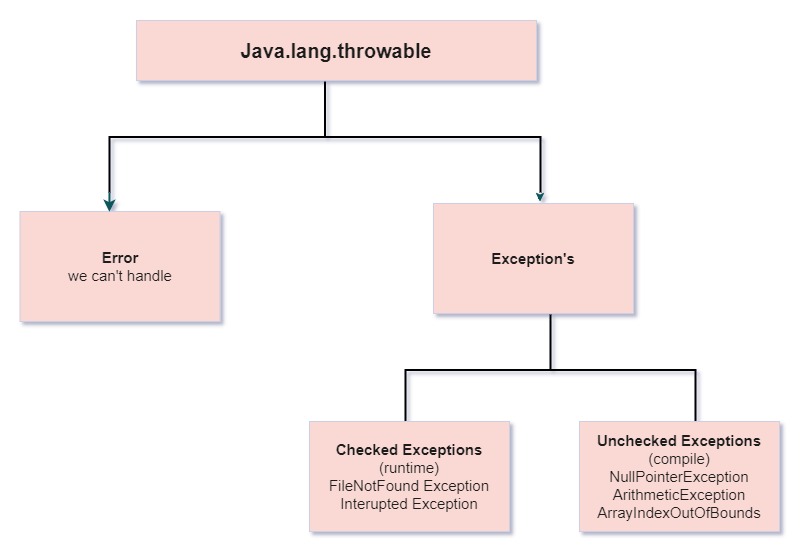
**Program:** ProductImp

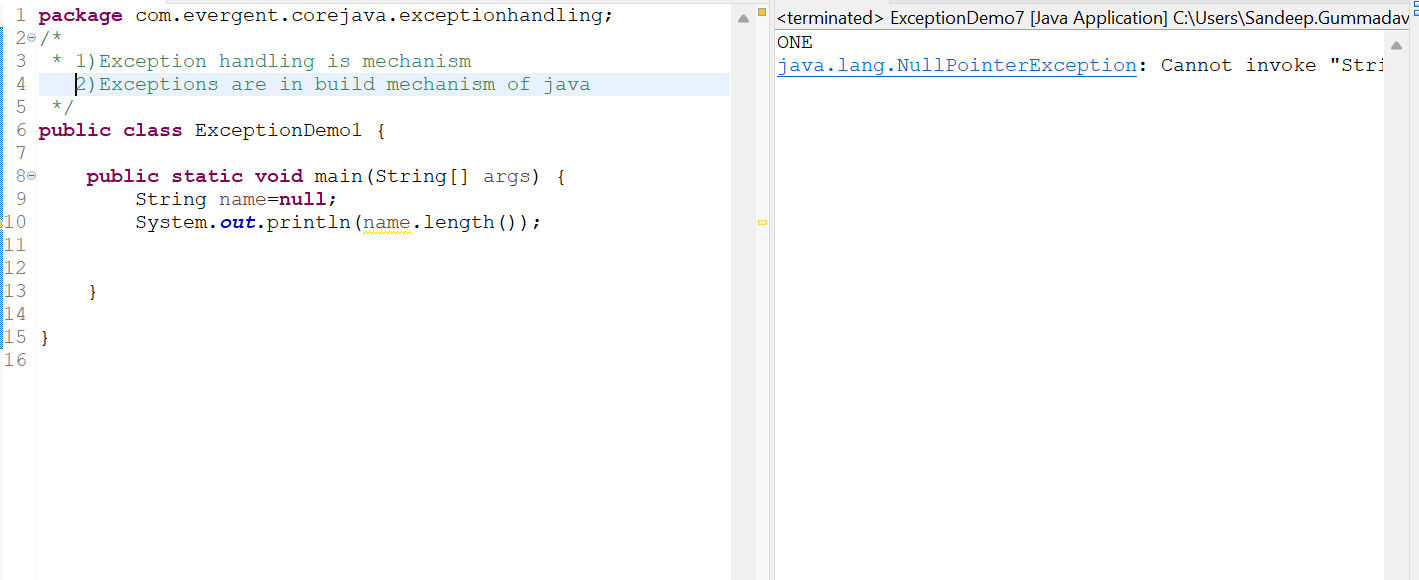




**Program:** Abstract Class Constructor

1. **Exception Handling**
2. Exception handling is mechanism
3. Exceptions are in build mechanism of java
4. All exceptions are executed while abnormal conditions only
5. In normal flow it won’t execute any exceptions
6. Once any exceptions are occurring in java code then remain lines of code is unreachable
7. Java.lang Throwable is super class for exception and errors
8. There are two types of exceptions in java
9. Checked Exception
10. Unchecked Exception
11. All checked Exceptions are compile time exceptions
12. All Unchecked Exceptions are runtime exceptions
13. There are 5 keywords in Exception handling
14. Try
15. Catch()
16. Finally()
17. Throws
18. Throw
19. Try for Business logic
20. Catch for exception handling
21. If Exception occurs or not finally block will be executed
22. Throws an exception will be executed method by method
23. Throw is for runtime and will call predefined exception
24. Try followed by either catch block or finally block
25. We should follow exception hierarchical
26. We can create our own(UserDefined) exceptions
27. Our own exception extends exception or runtime exception
28. All exception classes are in to java.lang.package
29. If there are two exceptions the developer need to fix the first exception then after only the second exception is handled
30. Error’s are not in developer control
31. Handling multiple Exceptions with throws
32. Handling Multiple Exceptions using Multi-Catch
33. In java we can use try inside a try is called nested try

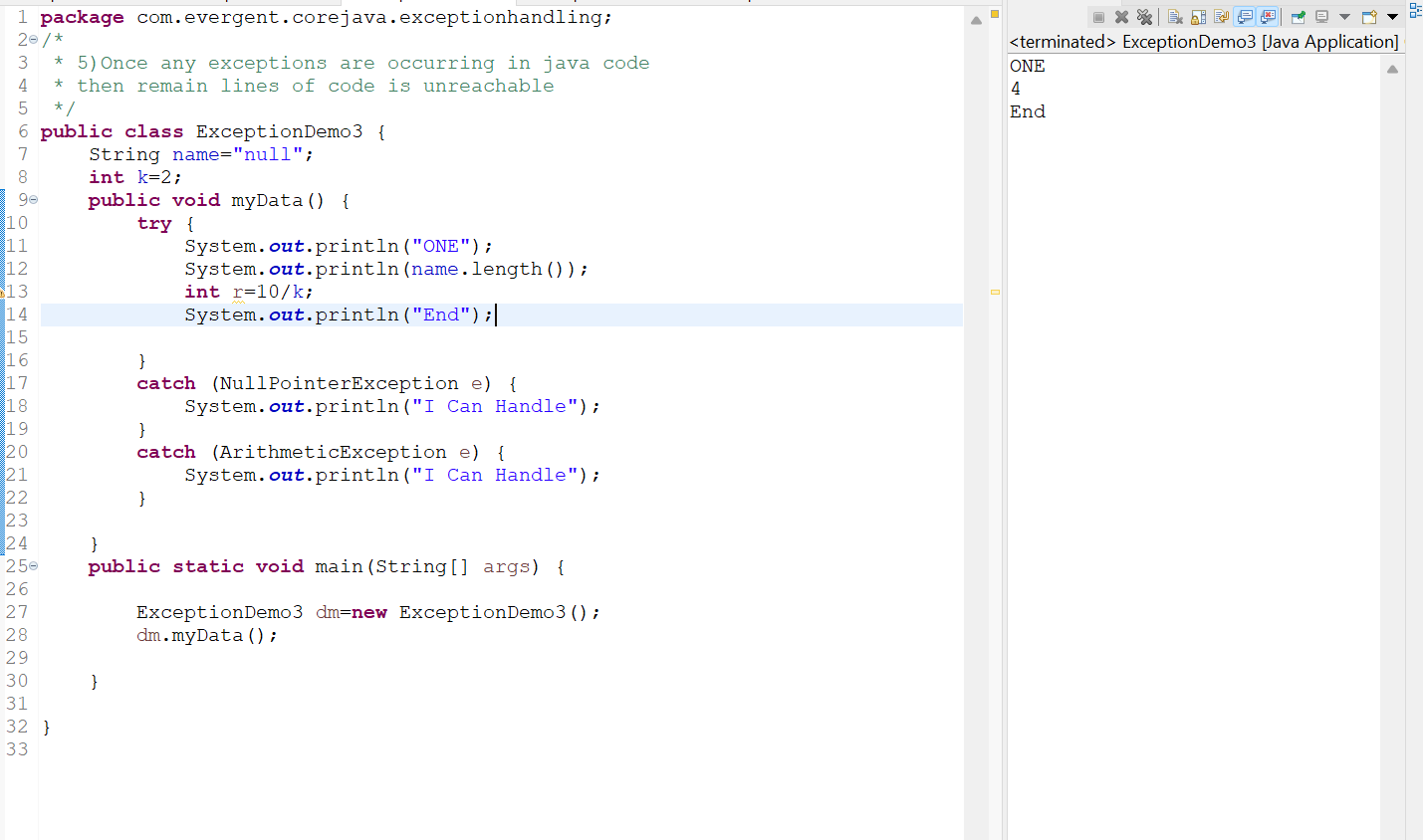




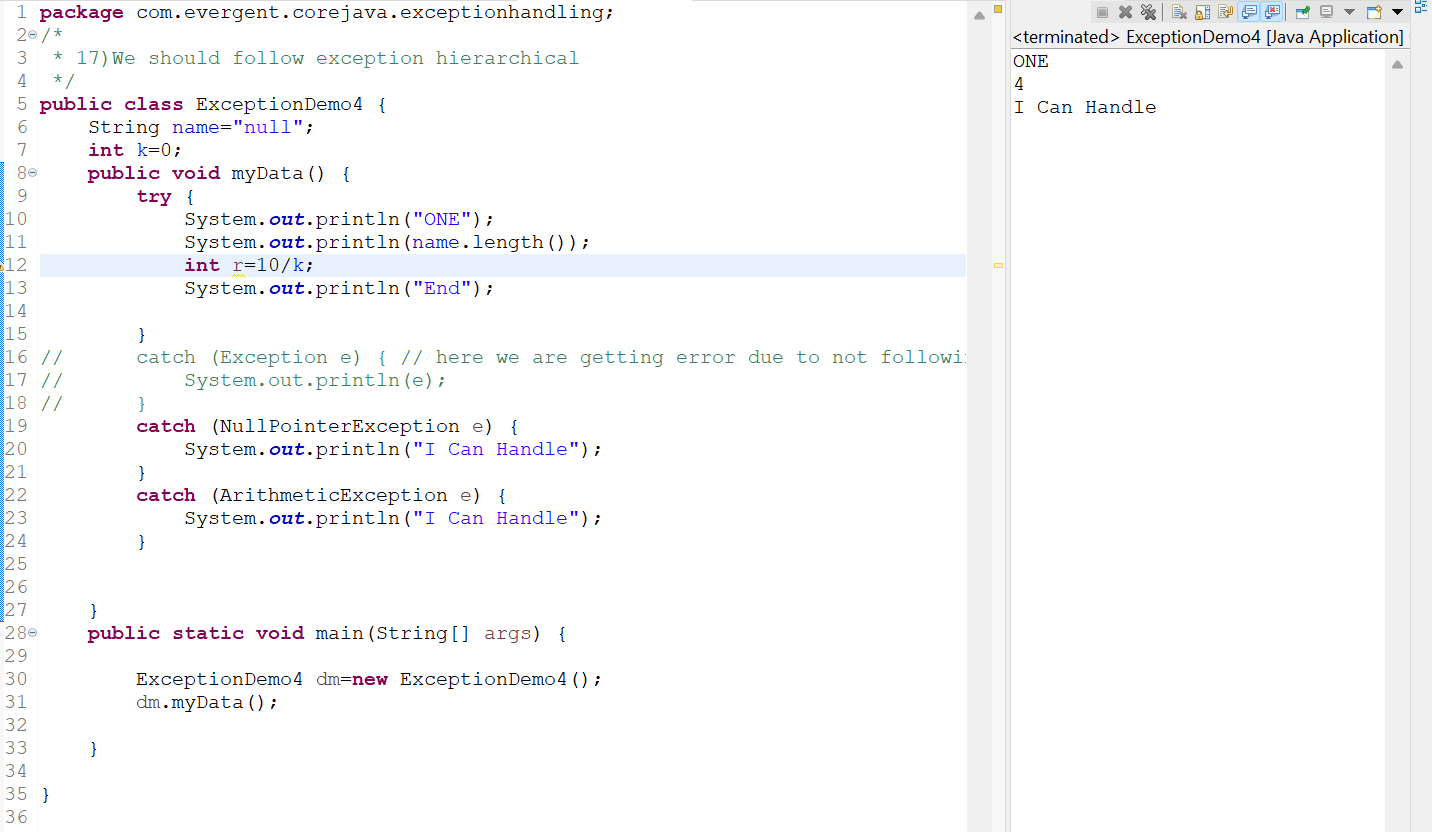
**Program:** Exception handling In Build Mechanism



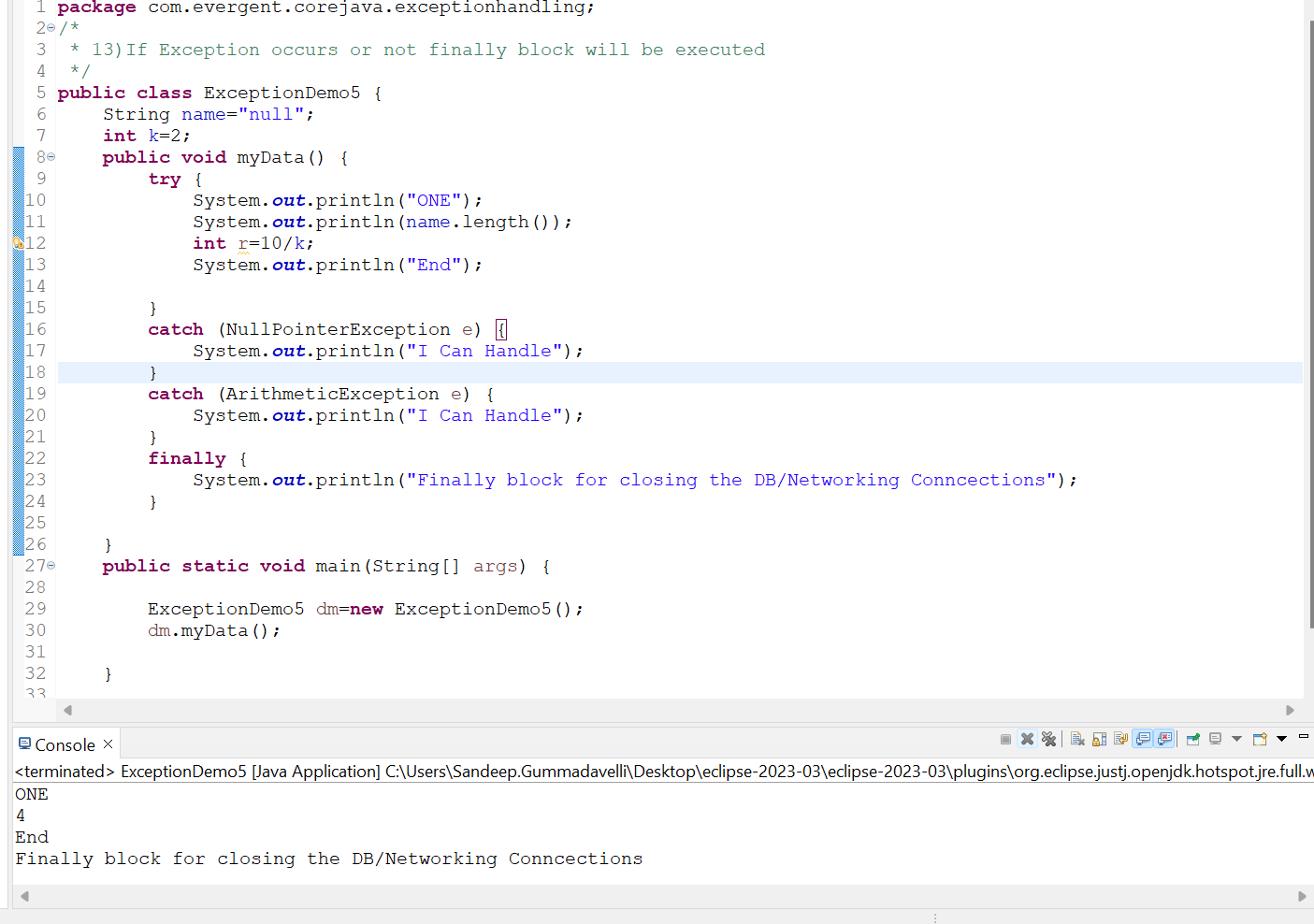
**Program:** All Exceptions are executed while abnormal conditions



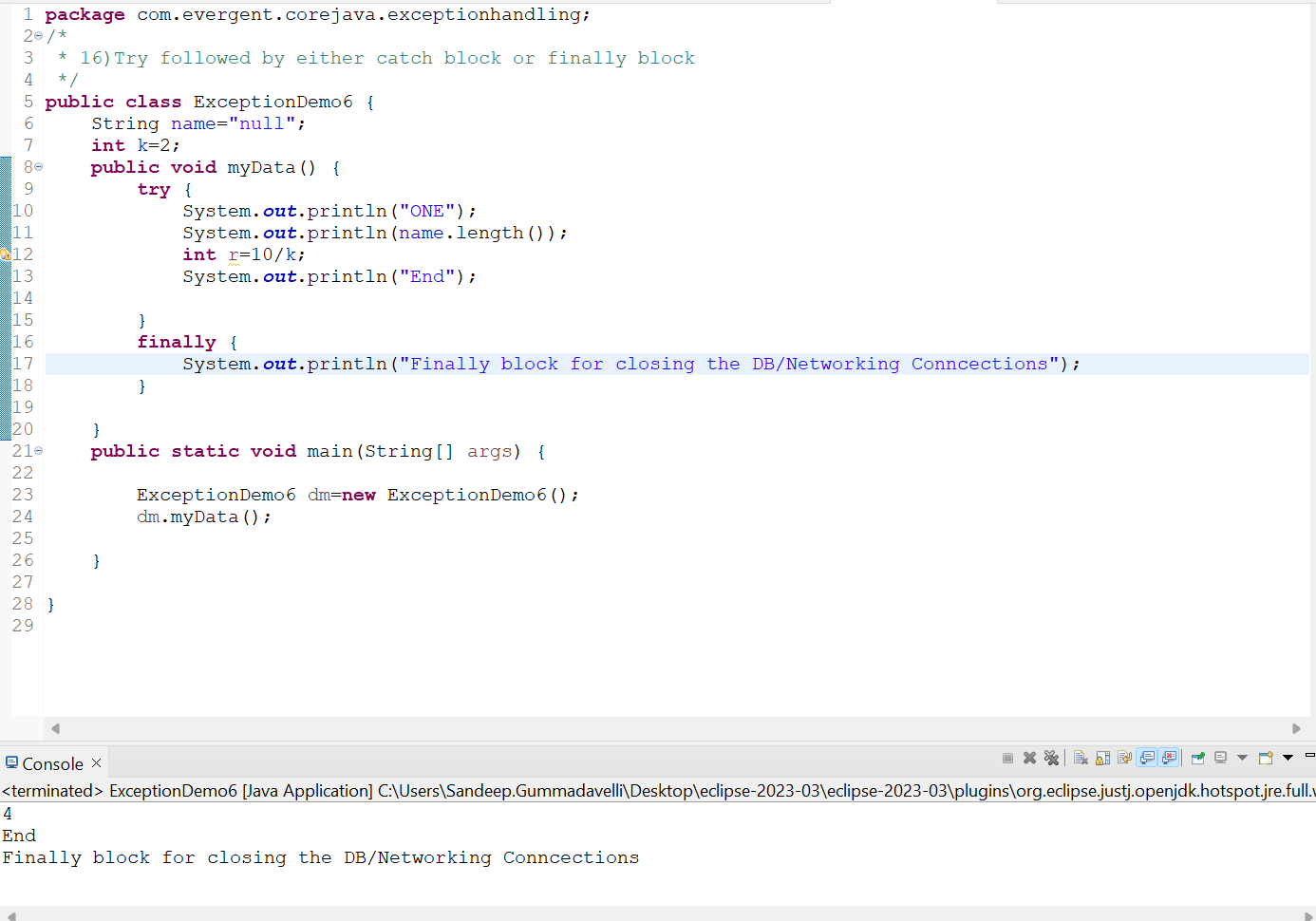
**Program:** ExceptionDemo3



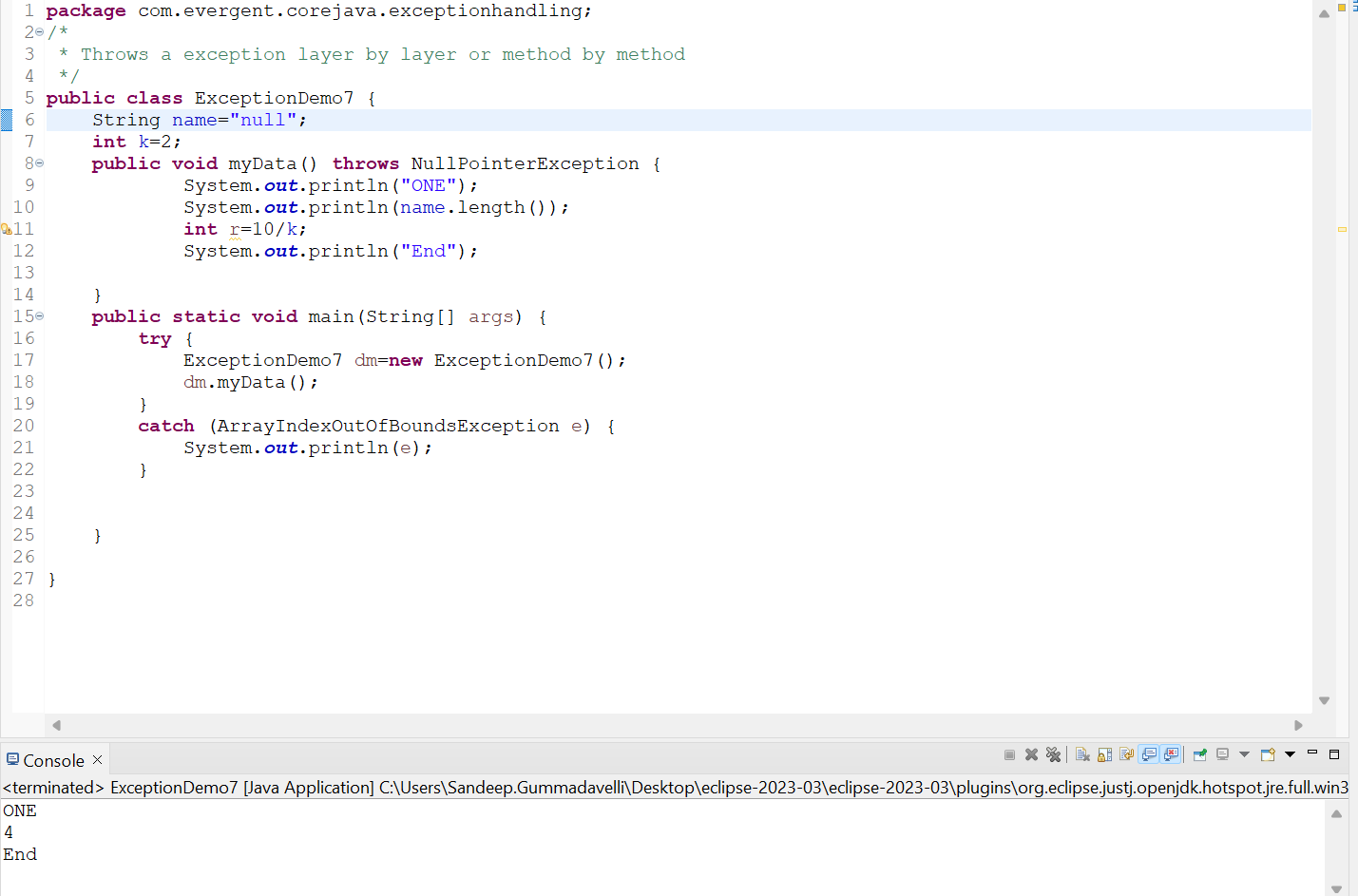
**Program:** We should follow exception hierarchical



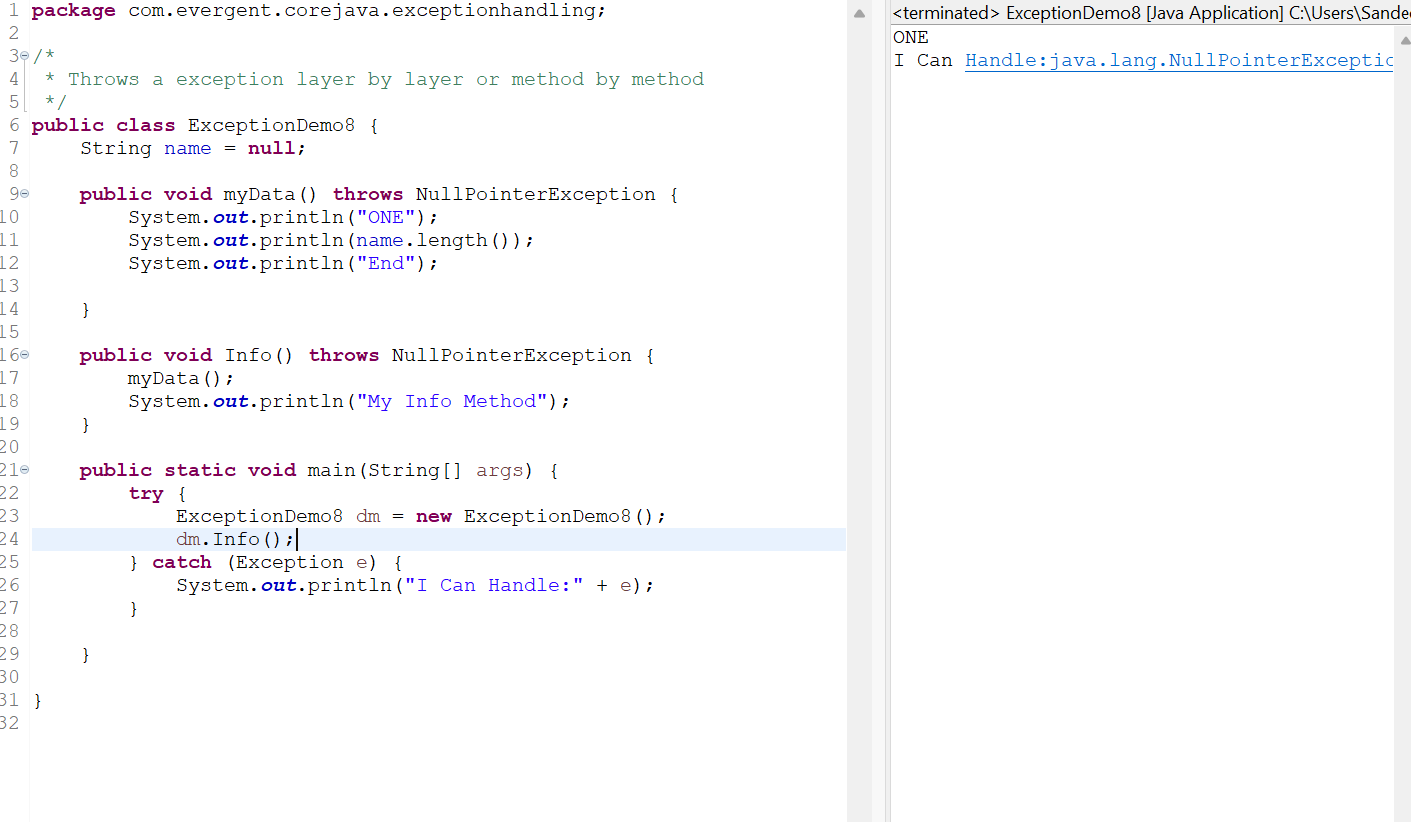
**Program:** If exception occurs or not finally block will be executed



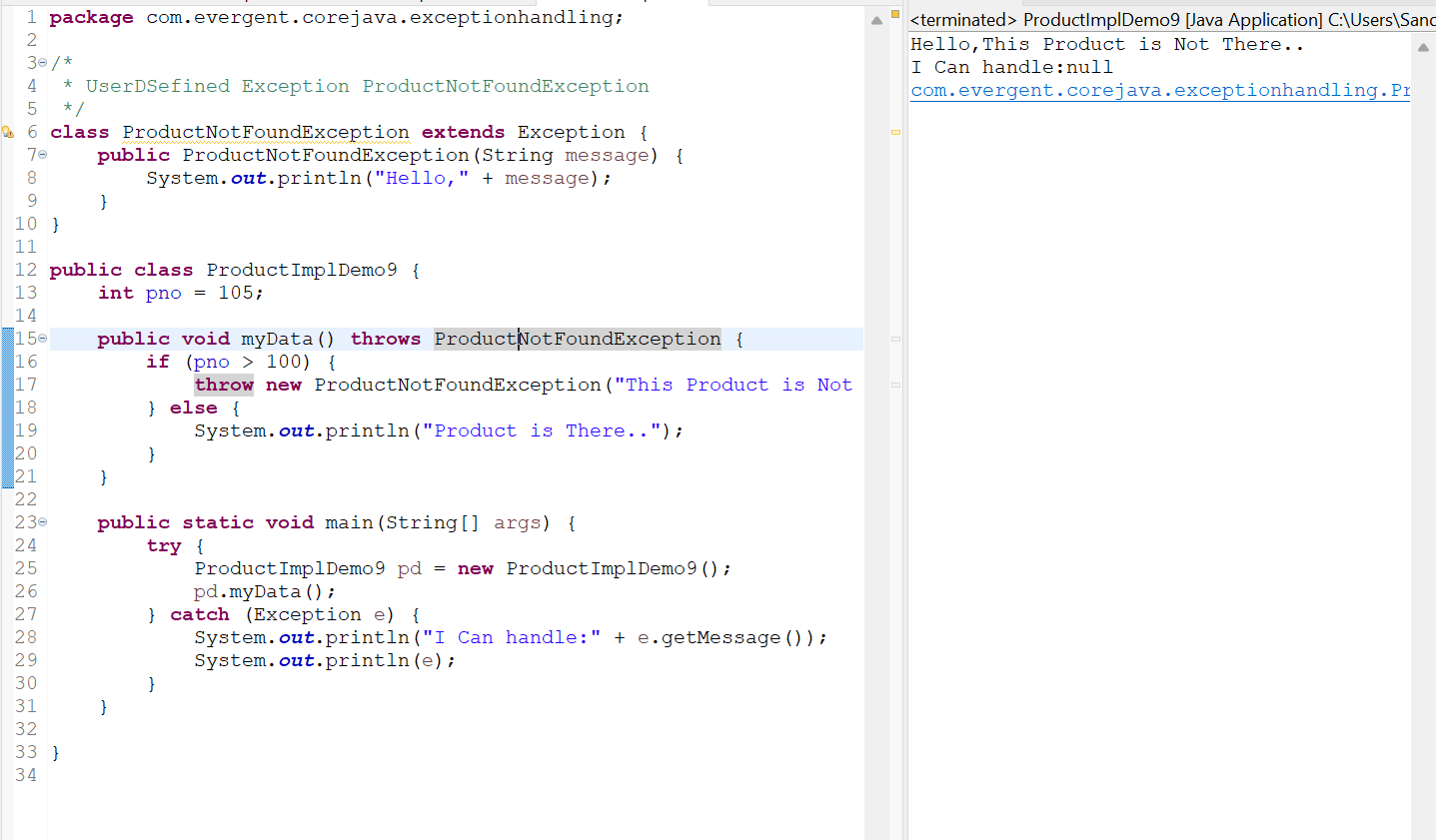
**Program:** Try followed by either catch block or finally block



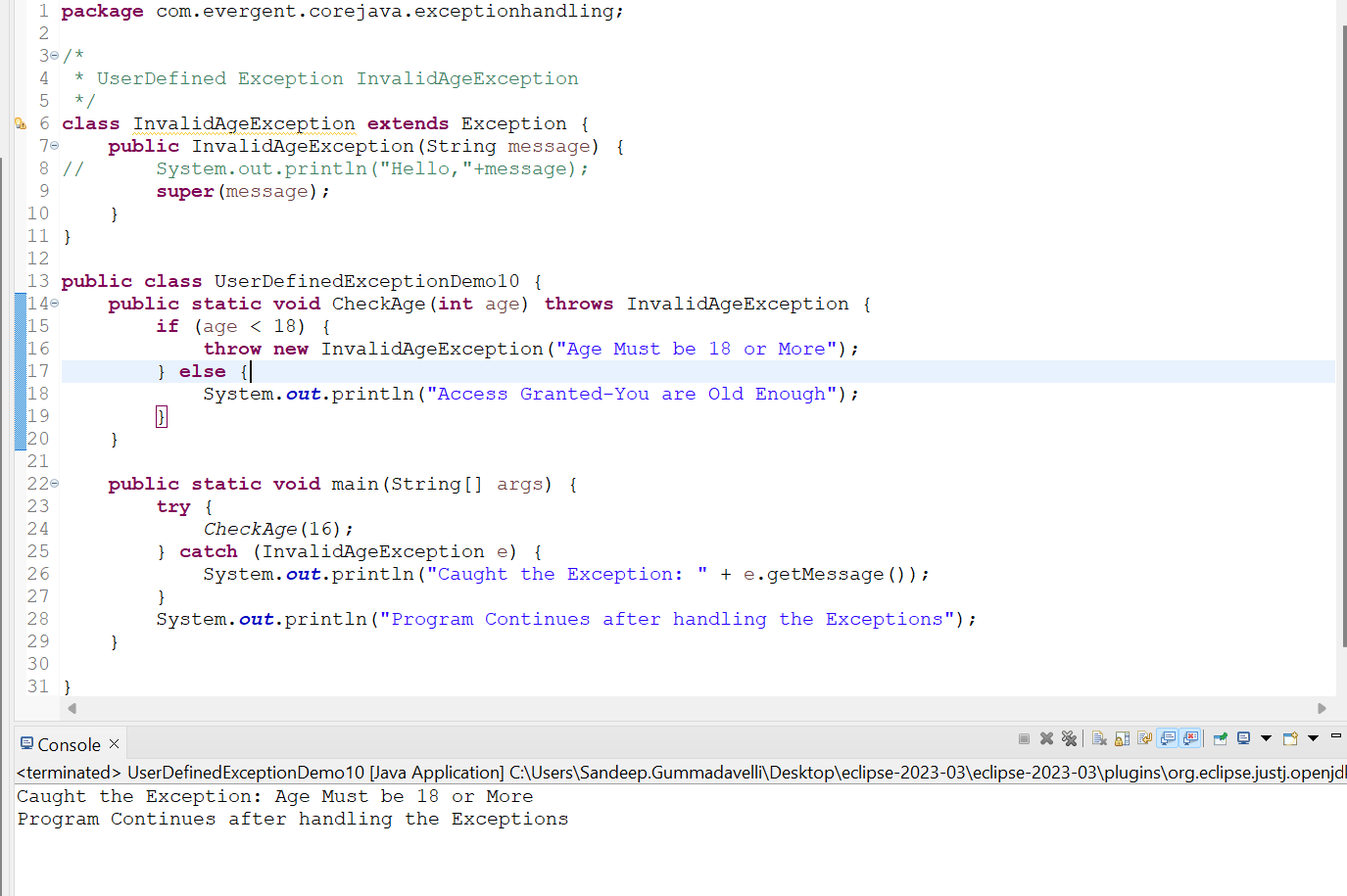
**Program:** Throws a exception layer by layer or method by method



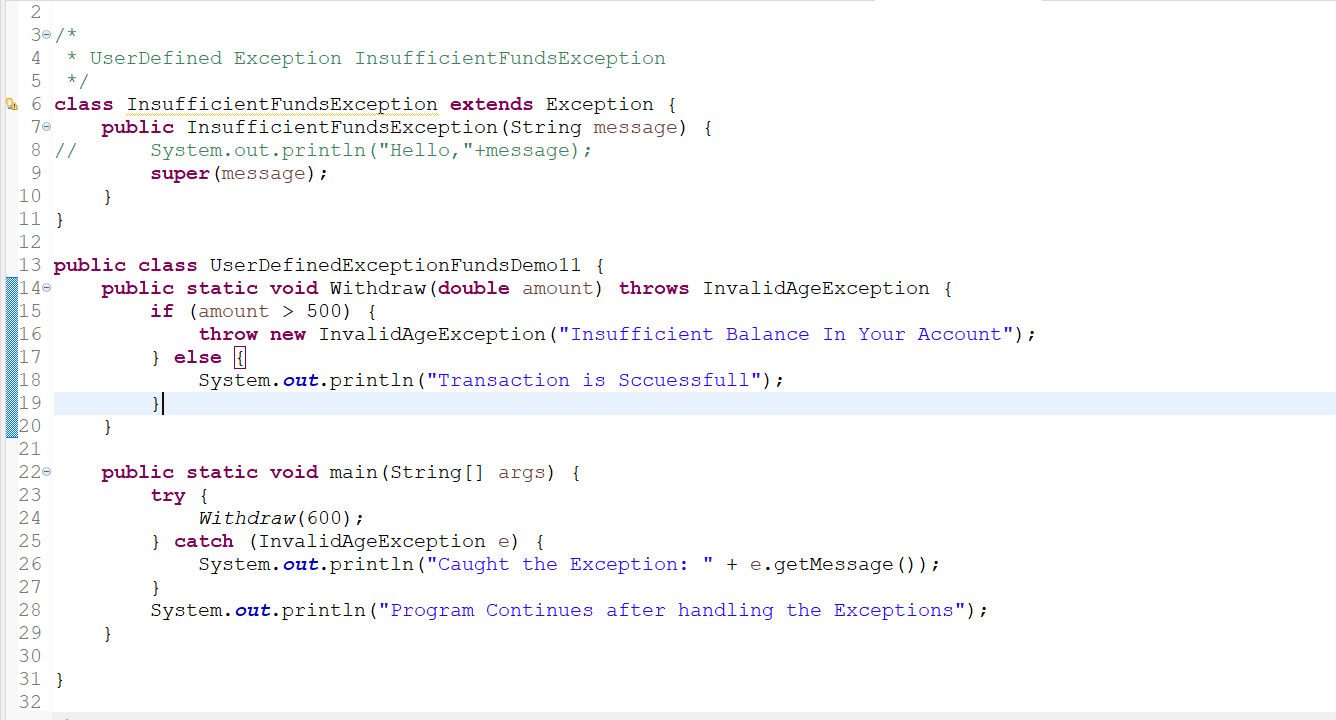
**Program:** Throws a exception layer by layer or method by method Demo2



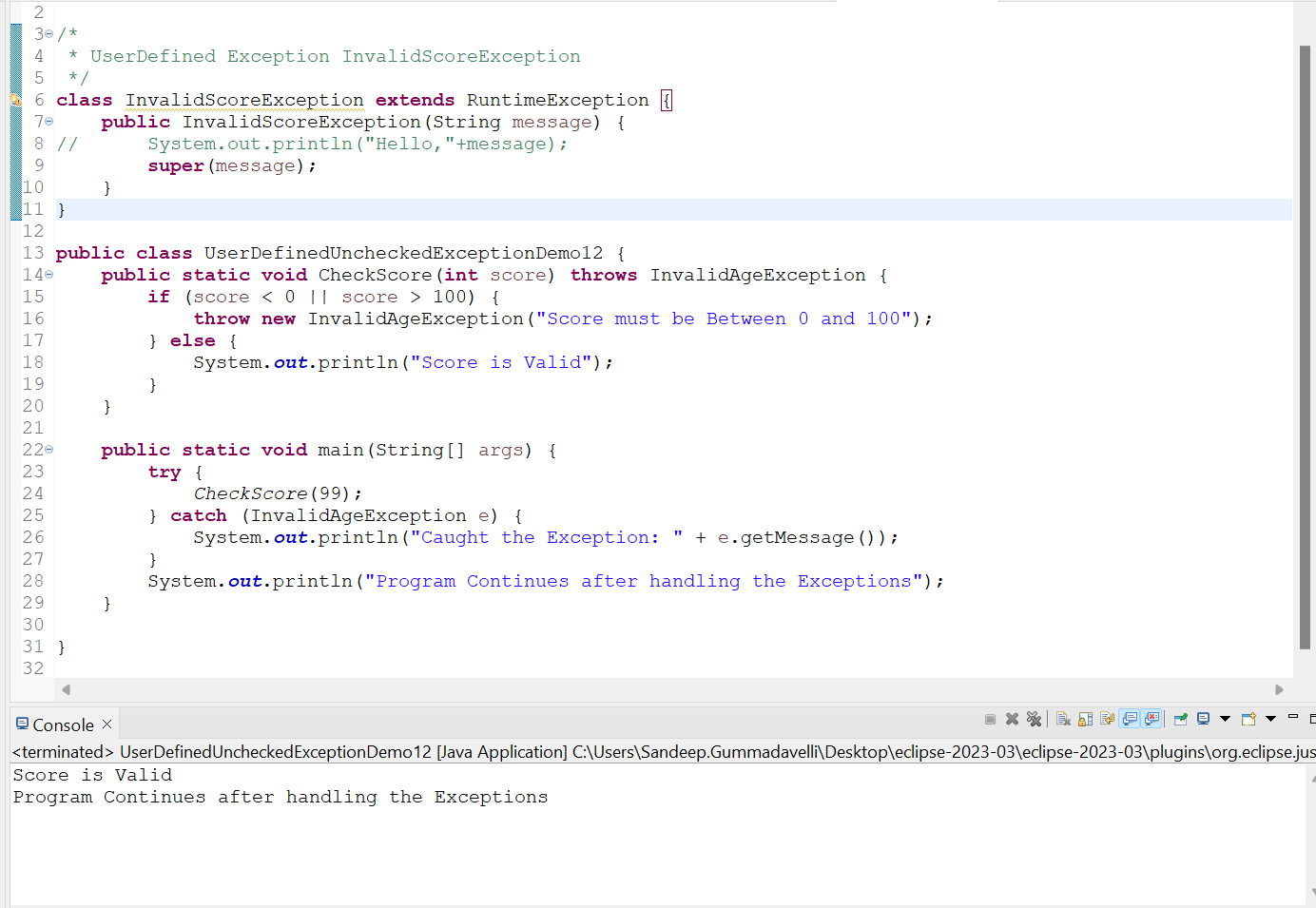
**Program:** UserDefined Exception ProductNotFoundException



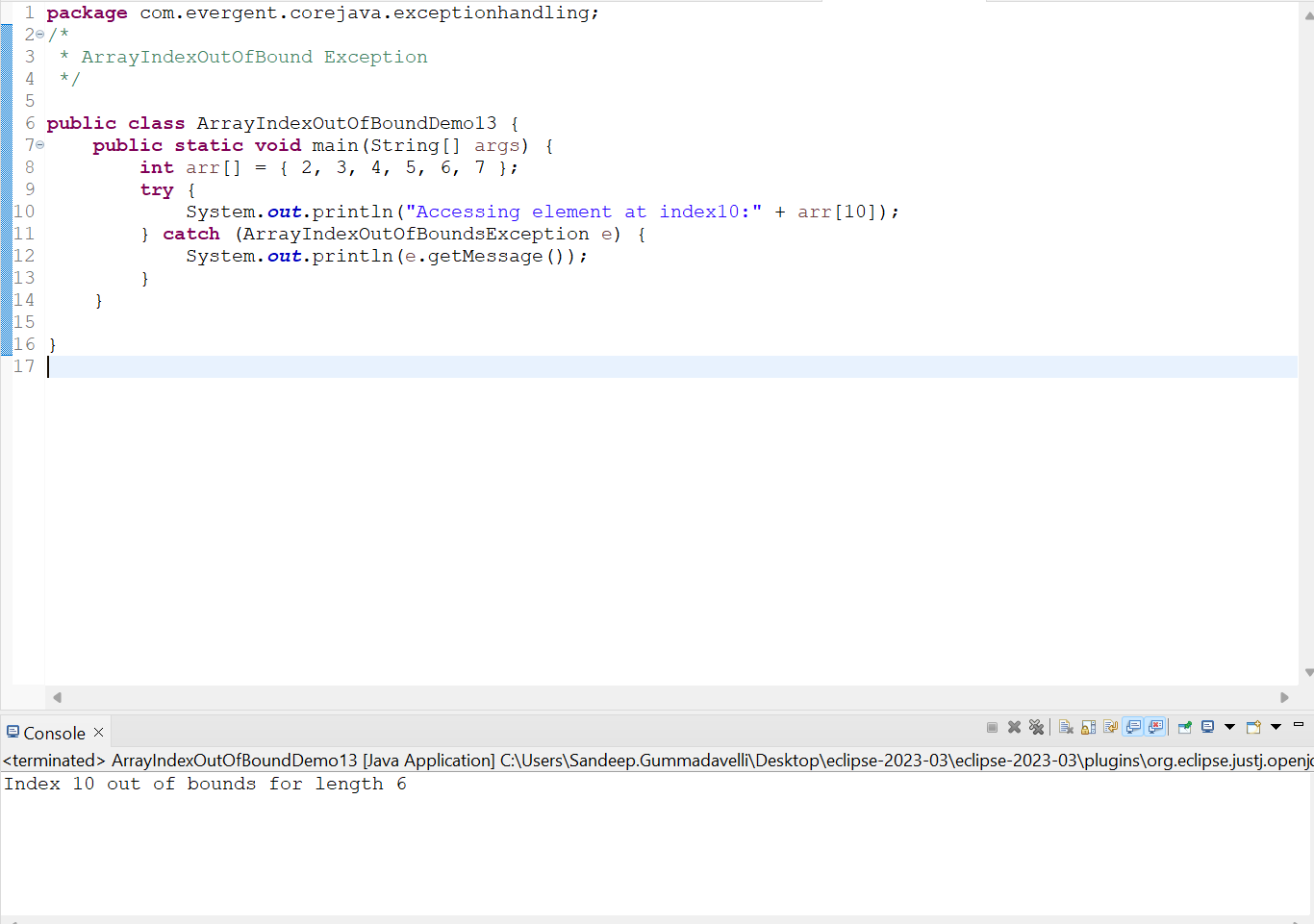
**Program:** UserDefined Exception InvalidAgeException



**Program:** UserDefined Exception InsufficientFundsException



**Program:** UserDefined Exception InvalidSoreException



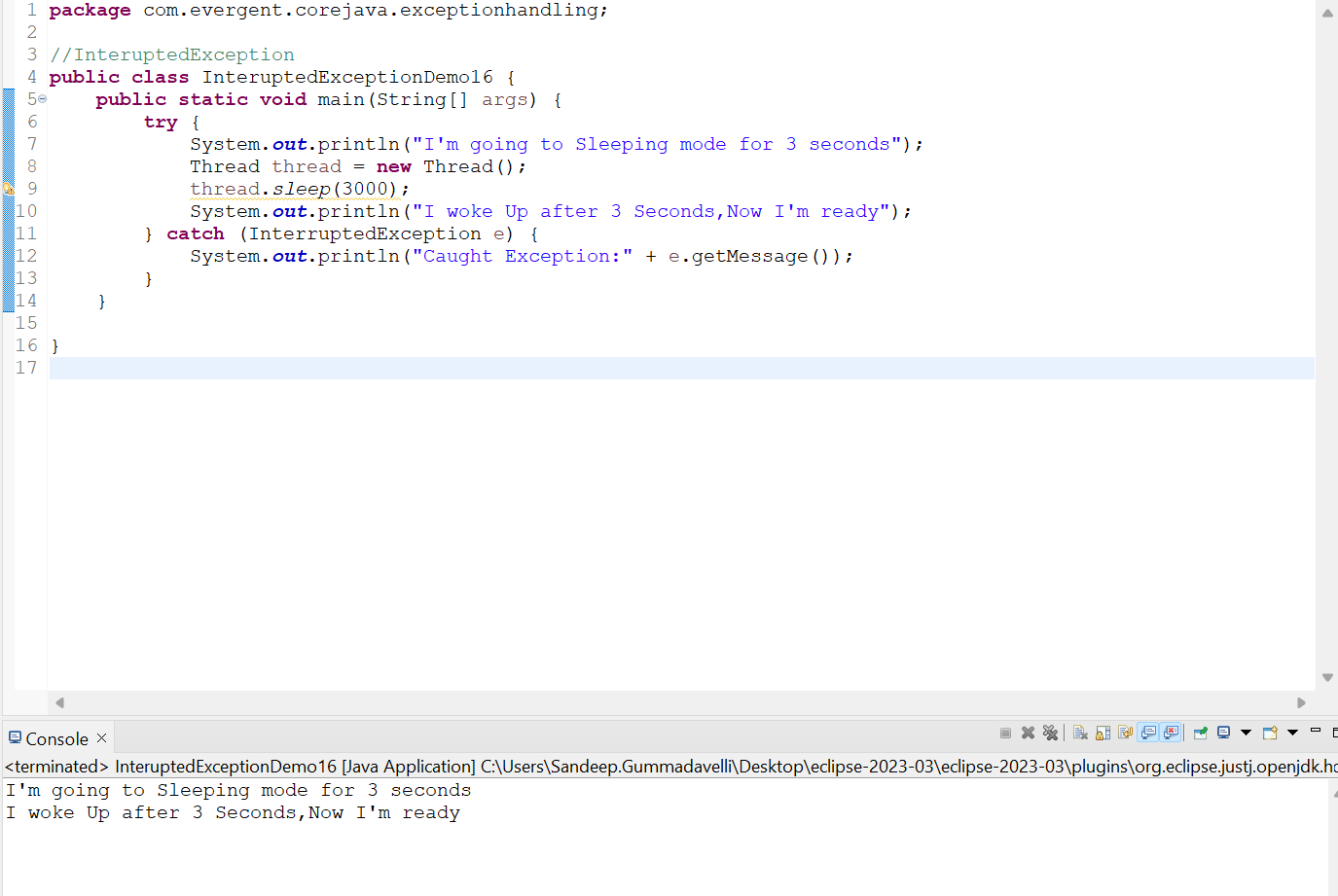
**Program:** ArrayIndexOutOfBoundsException



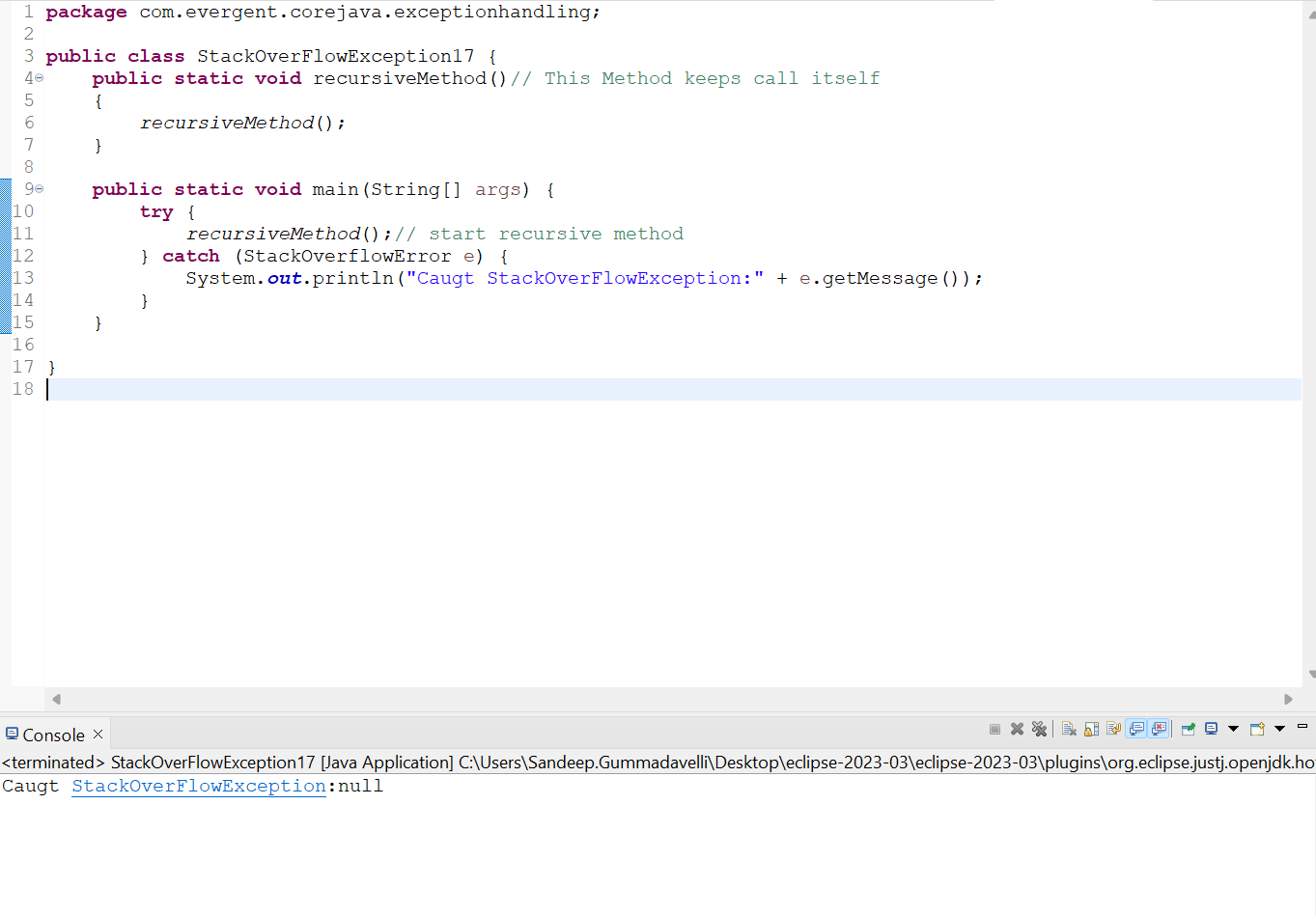
**Program:** CommandLineArugements



**Program:** CompileTimeFileDemo(FileNotFoundException)

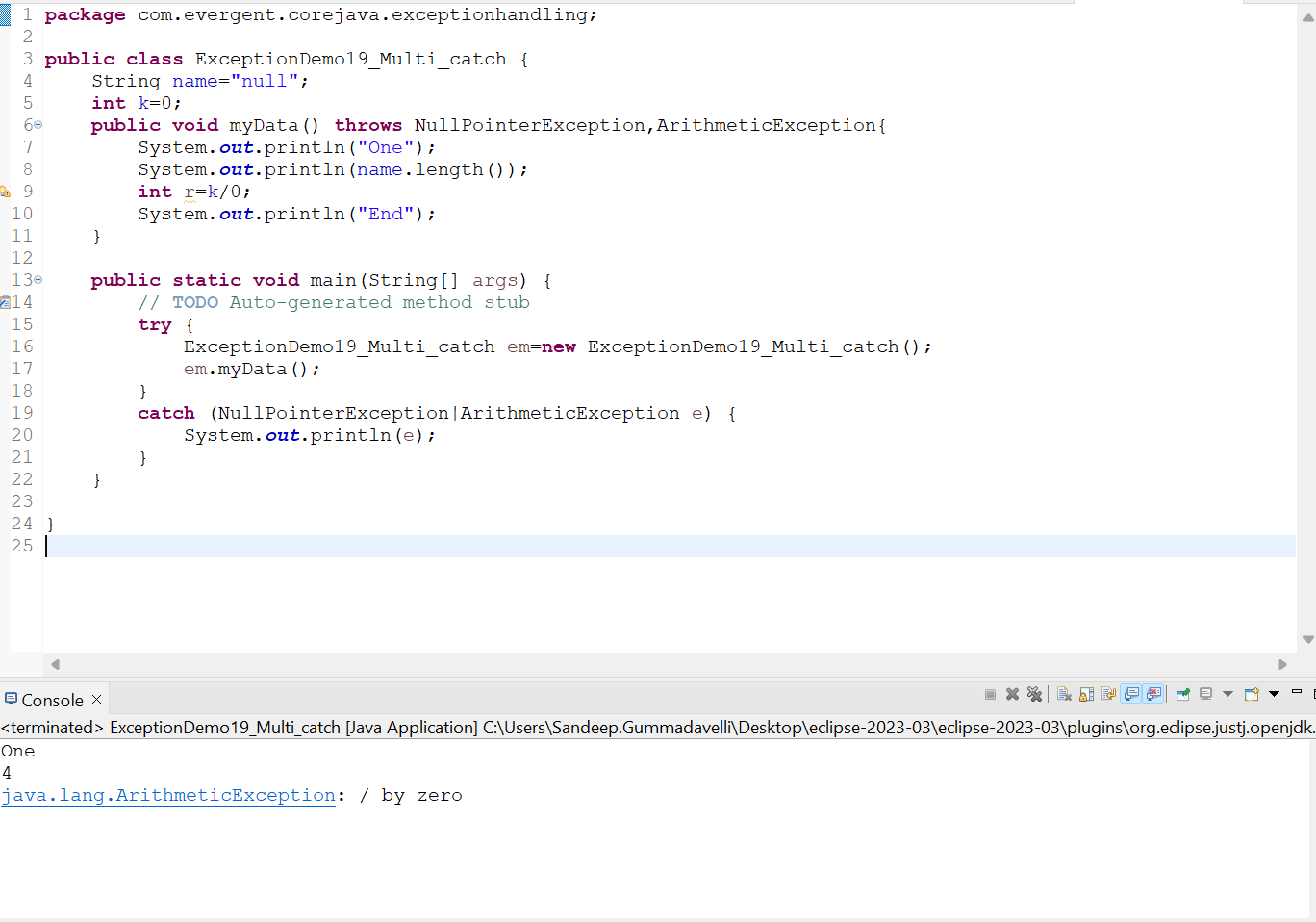


**Program:**InteruptedExceptionDemo16

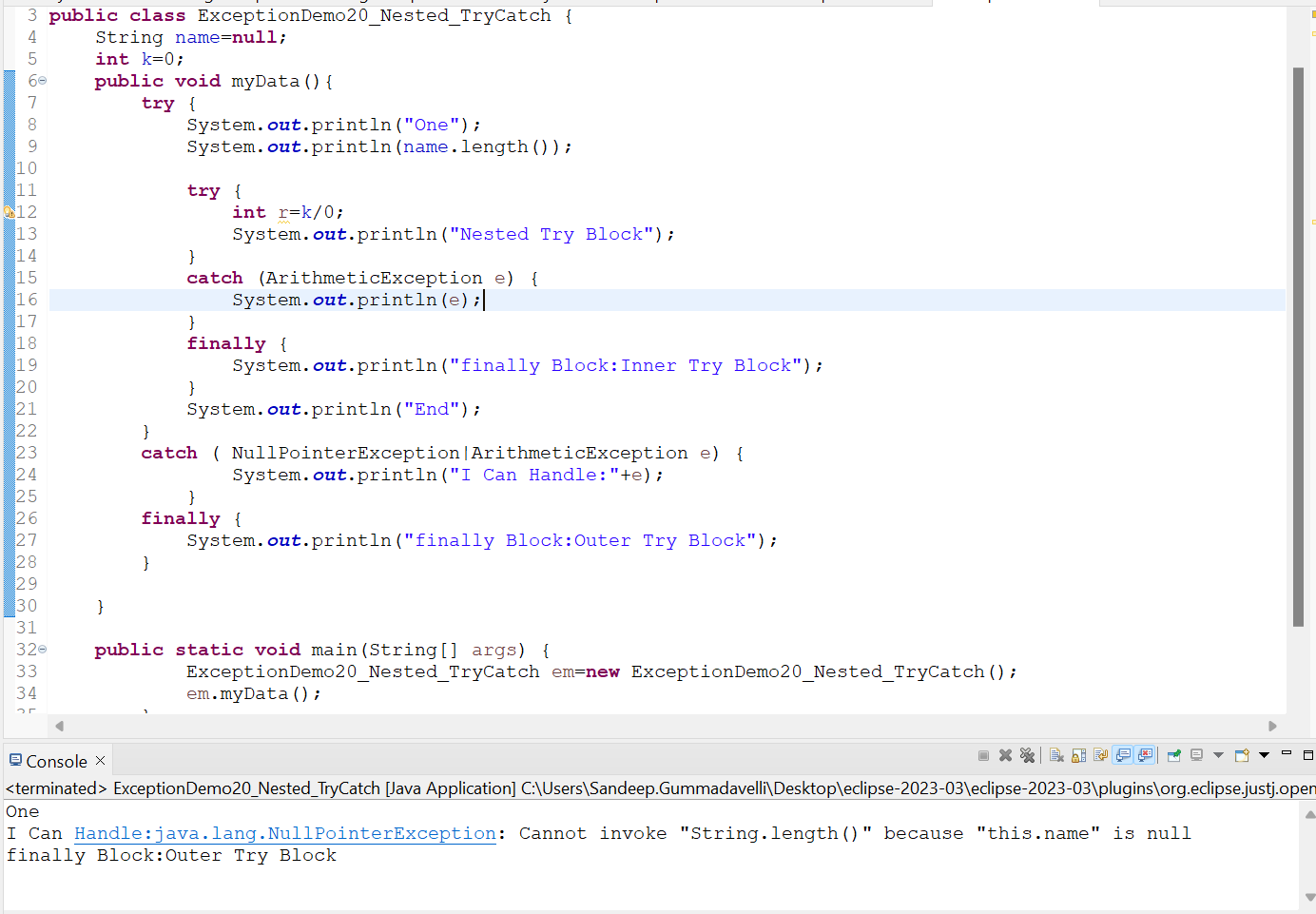
**Program:**StackoverFlowException



**Program:**throws with multiple exception



**Program:**Catch with multiple Exceptions



**Program:**nested try catch block