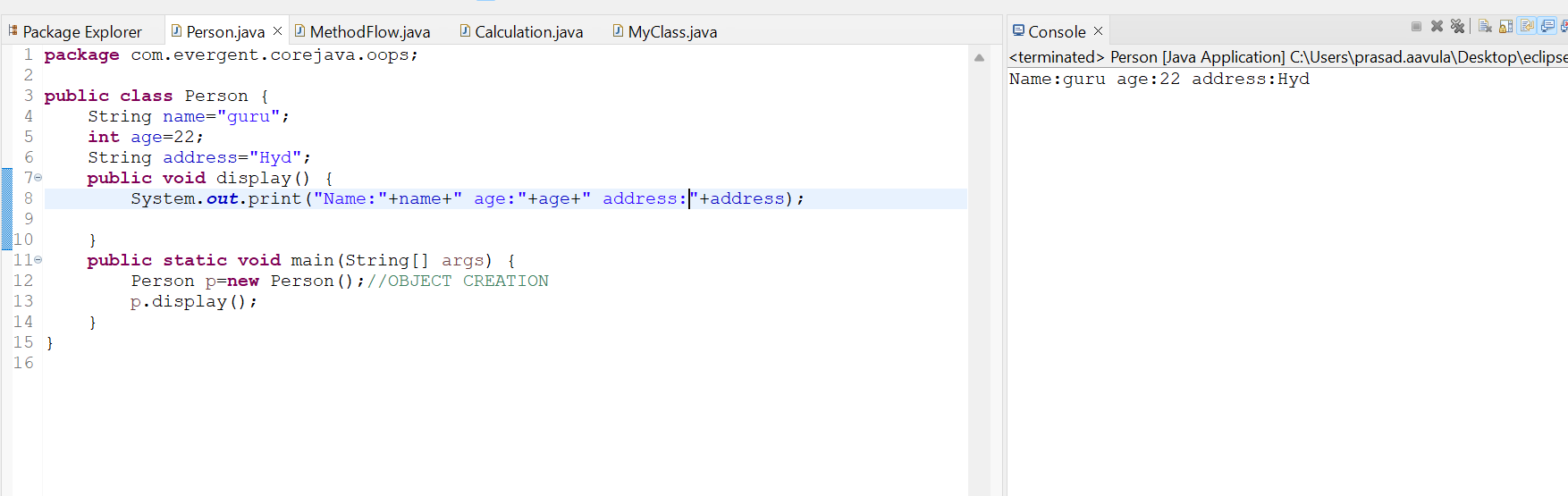
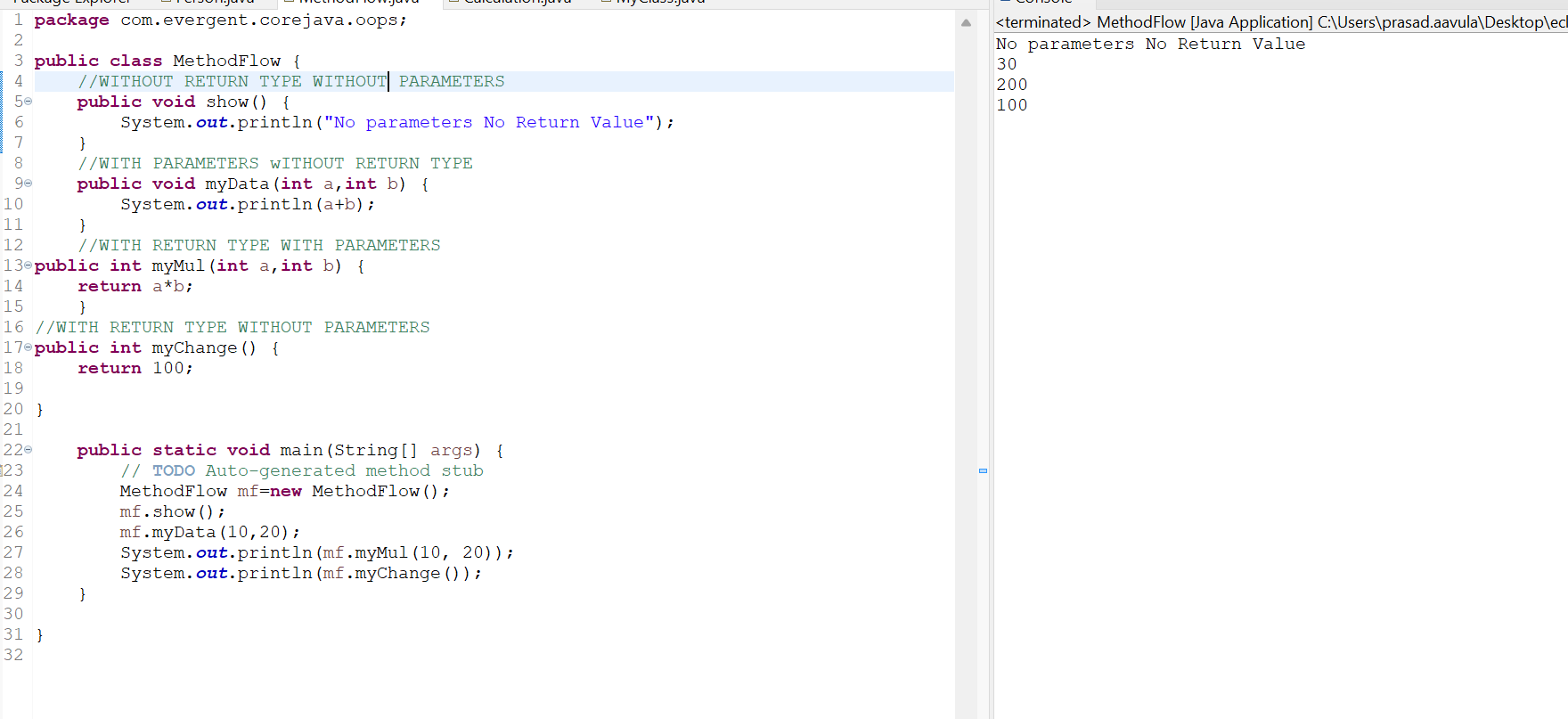
**Core Java Index**

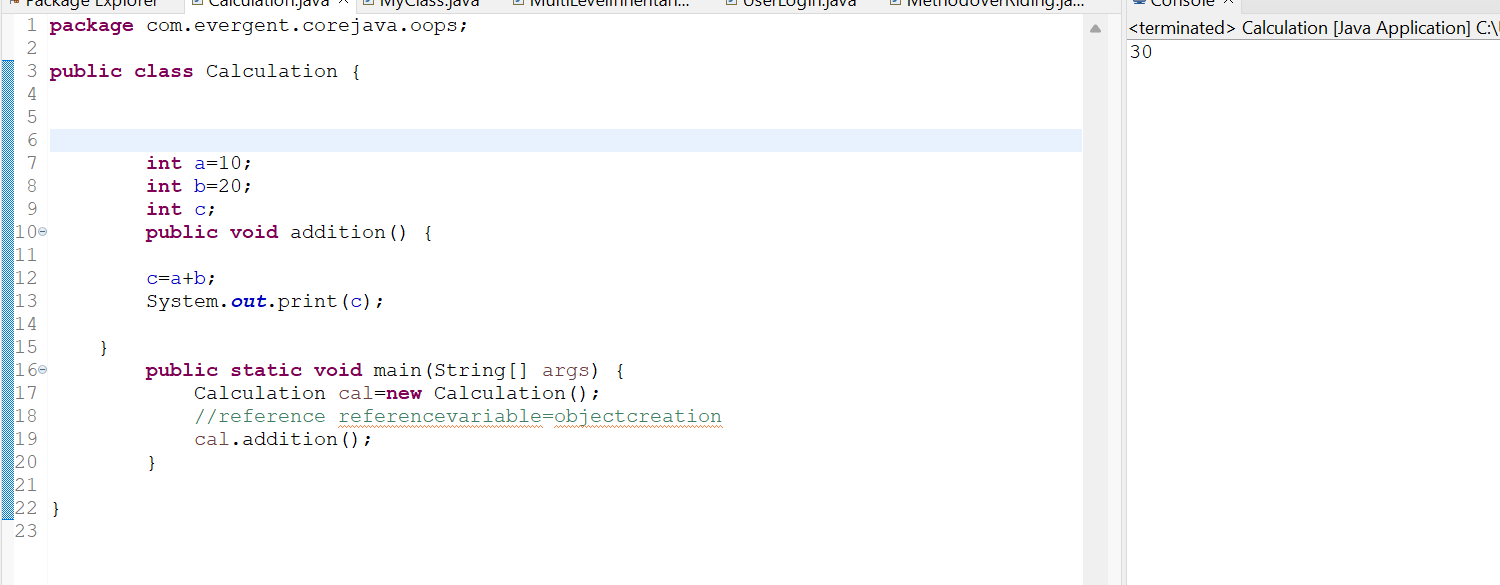
1. **Oops Concepts** **8-8-24 Day-4**
   1. **Encapsulation**  :Encapsultion is a mechanism of binding up of data(attributes and methods) together in class and can access attributes and methods through object creation and reference.

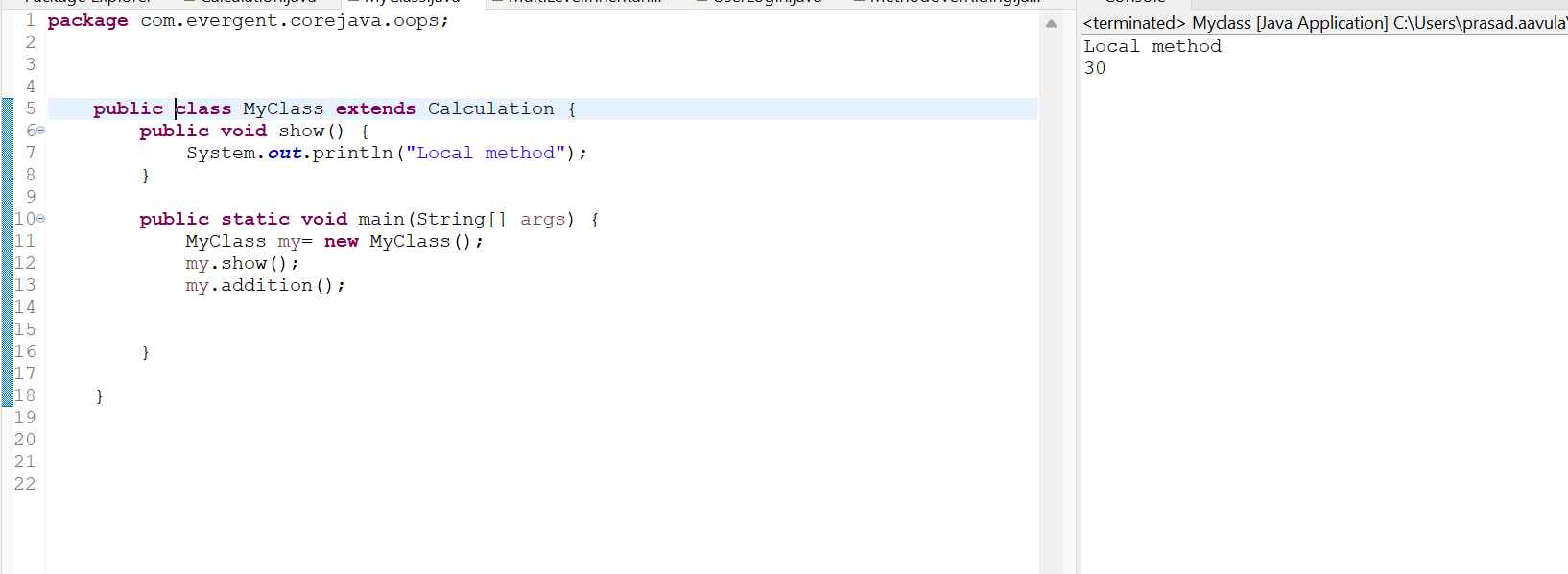


* + 1. Method Types or Method Flows
       1. No Parameters with no return type
       2. Parameters with no return type
       3. Parameters with return type
       4. No Parameters with return type

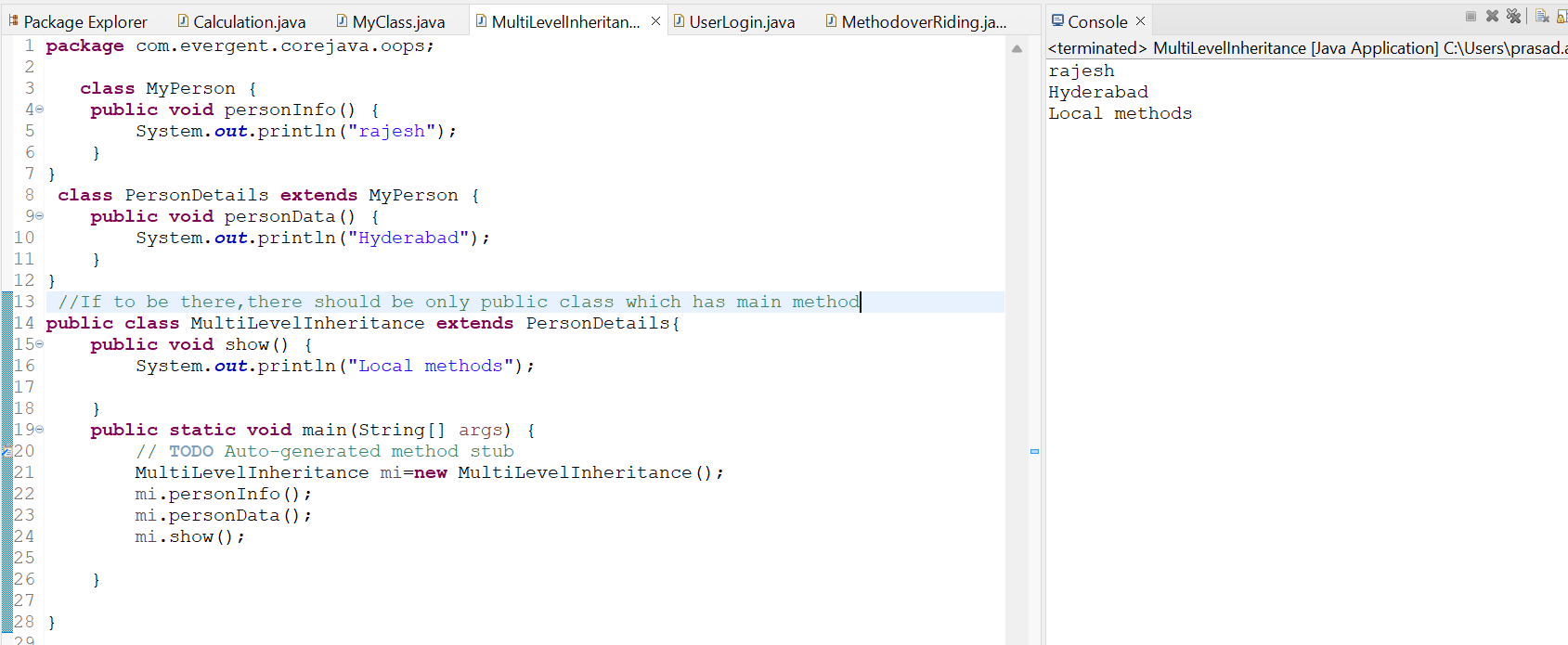


* 1. **Inheritance:**It is the reusability of existing functionalities from super class to sub class is inheritance.extends is a keyword used to inherit the properties of superclass.





* **MultilevelInheritance**:Multilevel inheritance in Java occurs when a class inherits from a class, and then another class inherits from that derived class.



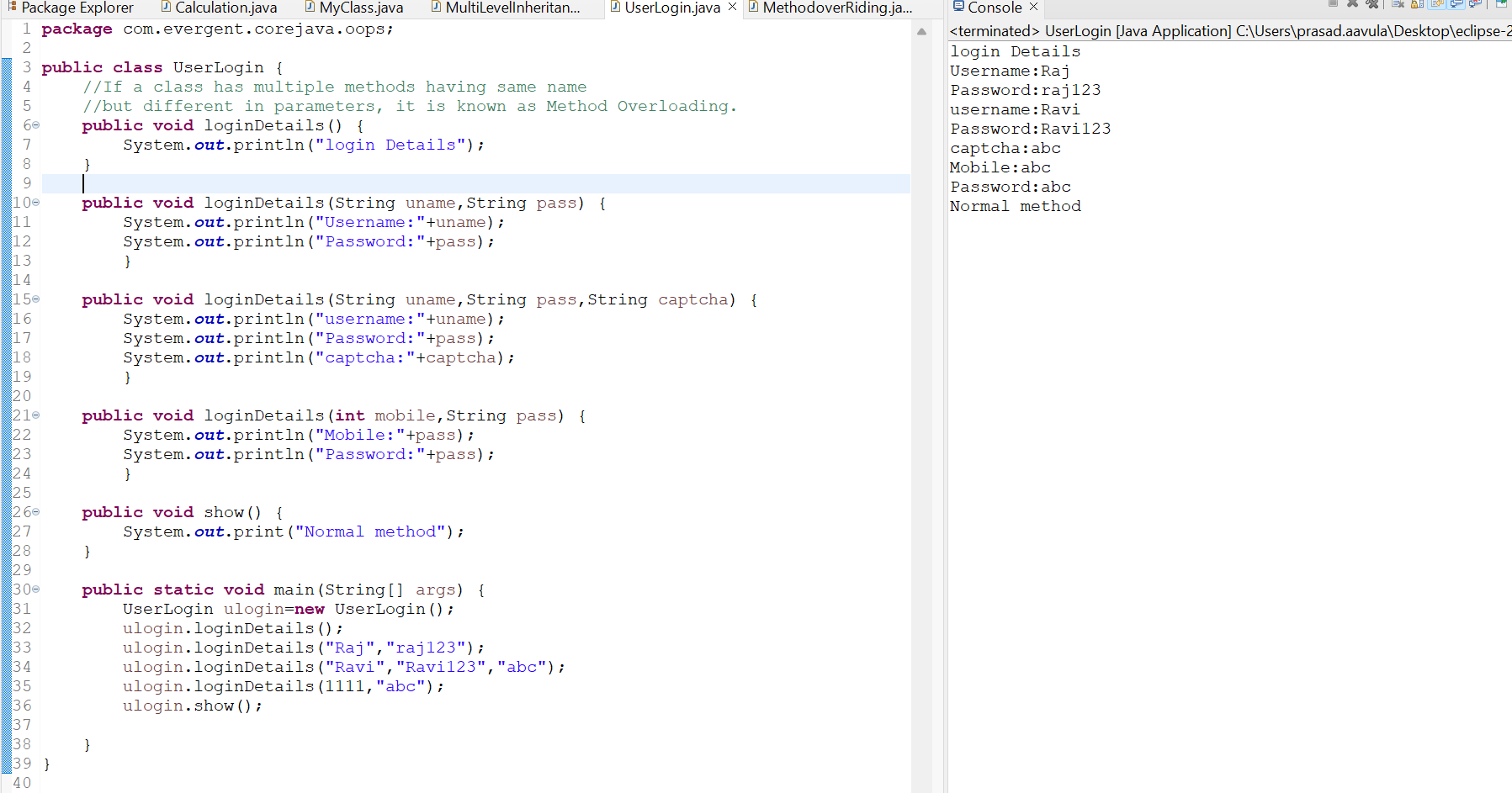
* 1. **Polymorphism:**Polymorphism in Java is a concept by which we can perform a single action in different ways.

Types of polymorphism:

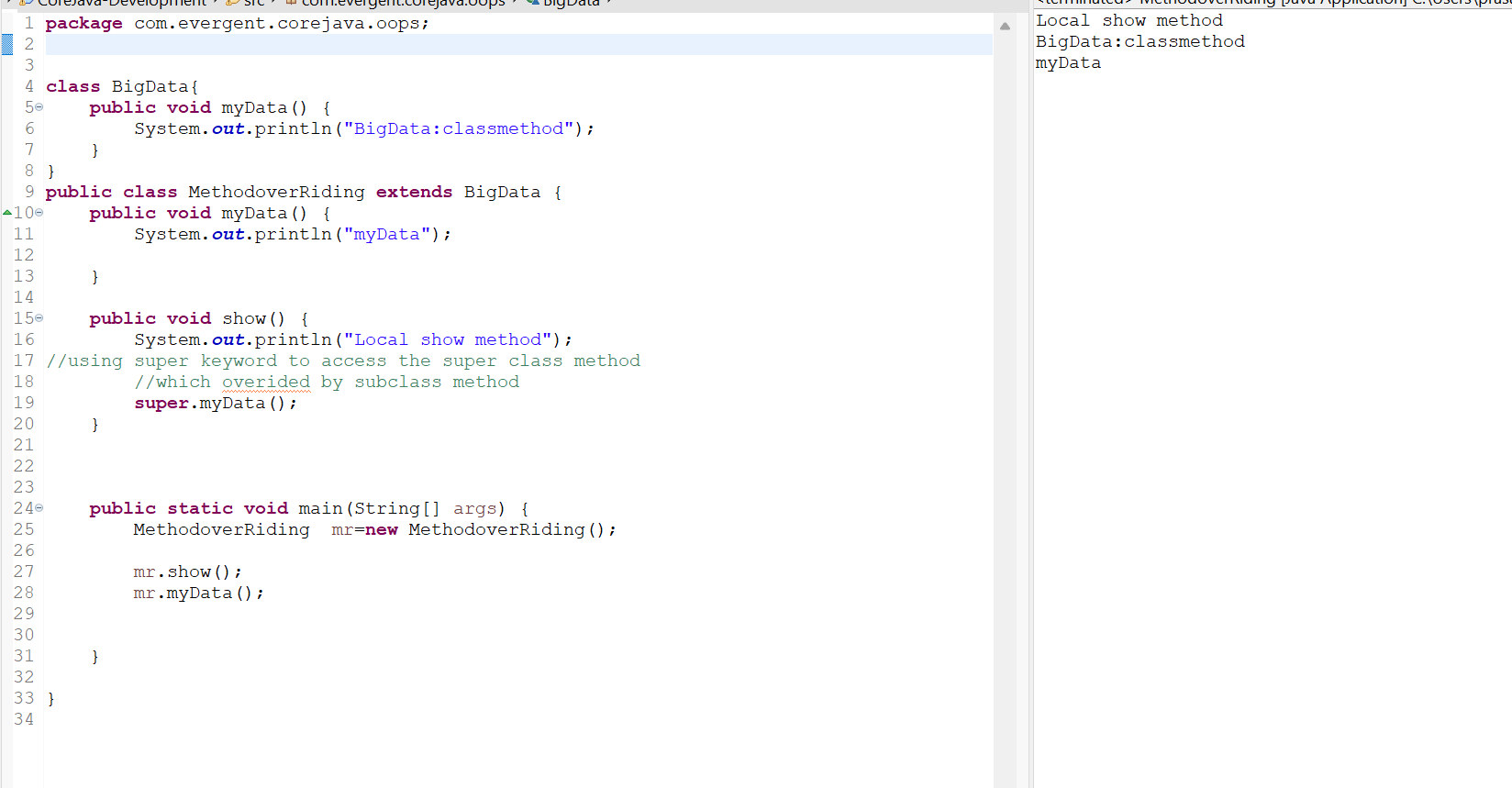
Compiletime polymorphism: Compile-time polymorphism is achieved when there are multiple methods with the same name but different parameters in a class. The appropriate method is determined based on the arguments used during compilation.

Runtime Polymorphism: (Dynamic Method Dispatch) is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

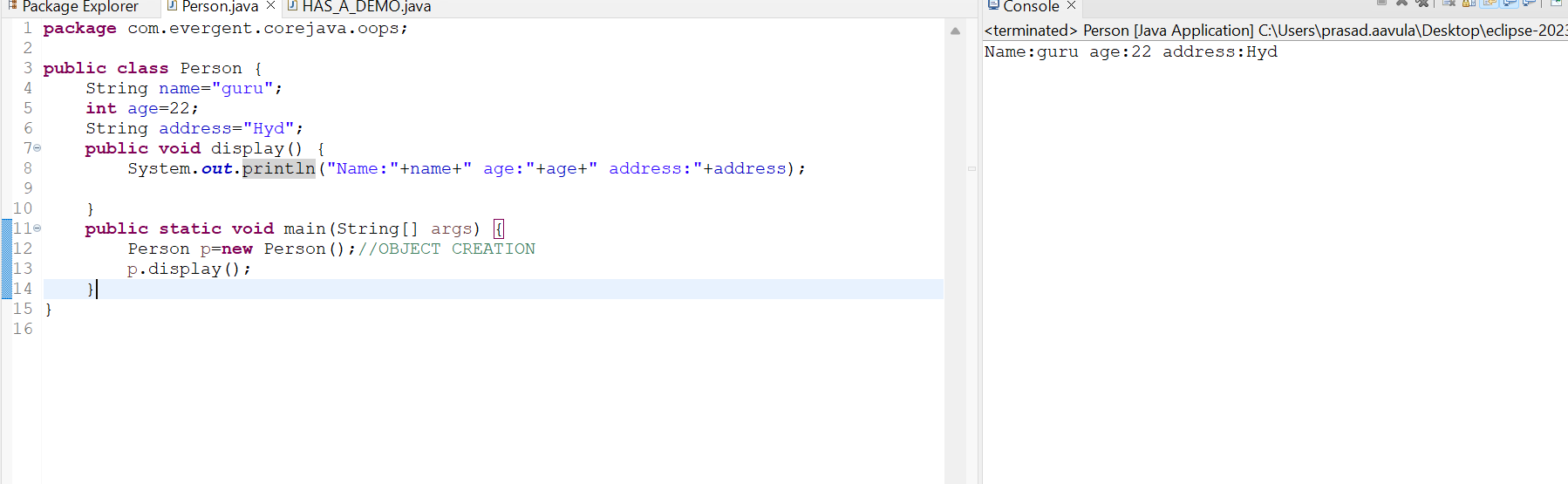
* + 1. Method Overloading: If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

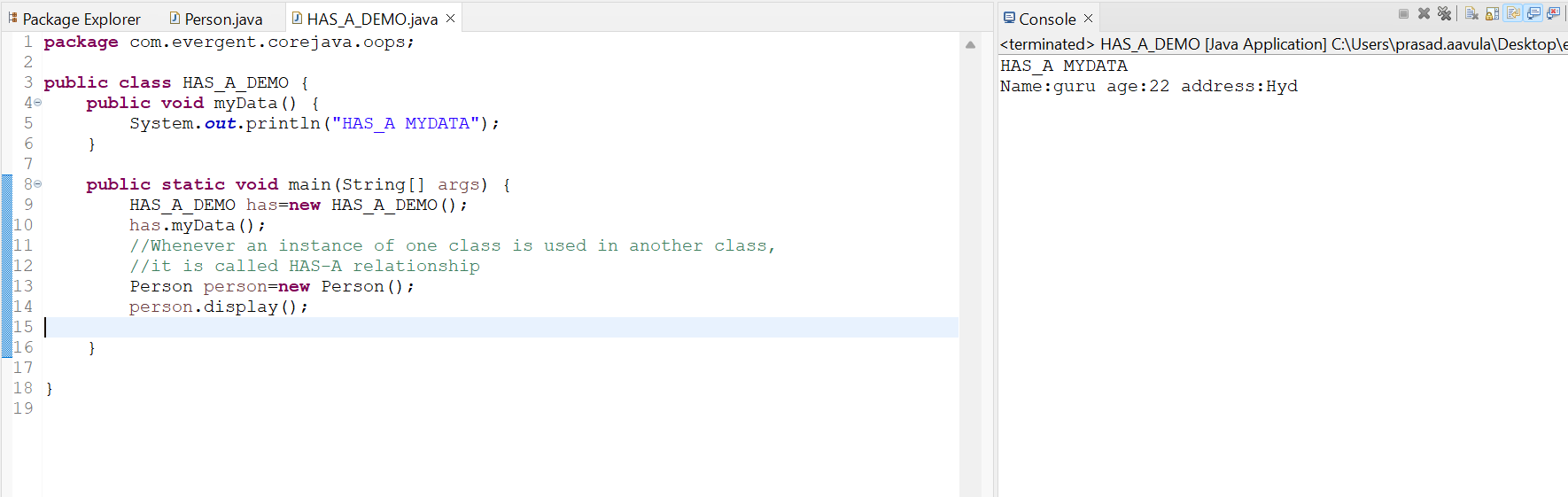


* + 1. Method Overriding:if subclass (child class) has the same method as declared in the parent class, it is known as method overriding.

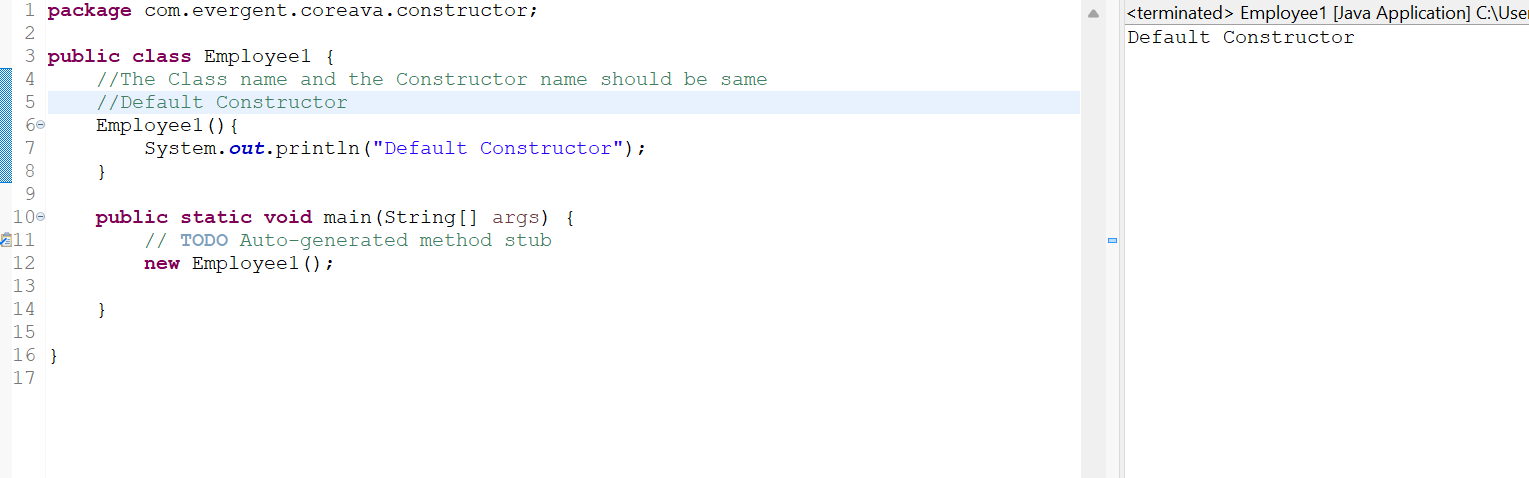


* 1. **Abstraction**:Abstraction in Java is the process in which we only show essential details/functionality to the user. The non-essential implementation details are not displayed to the user.It can be achieved by using interfaces.
* IS-A: Whenever one class inherits another class, it is called an IS-A relationship.
* HAS-A:Whenever an instance of one class is used in another class, it is called HAS-A relationship



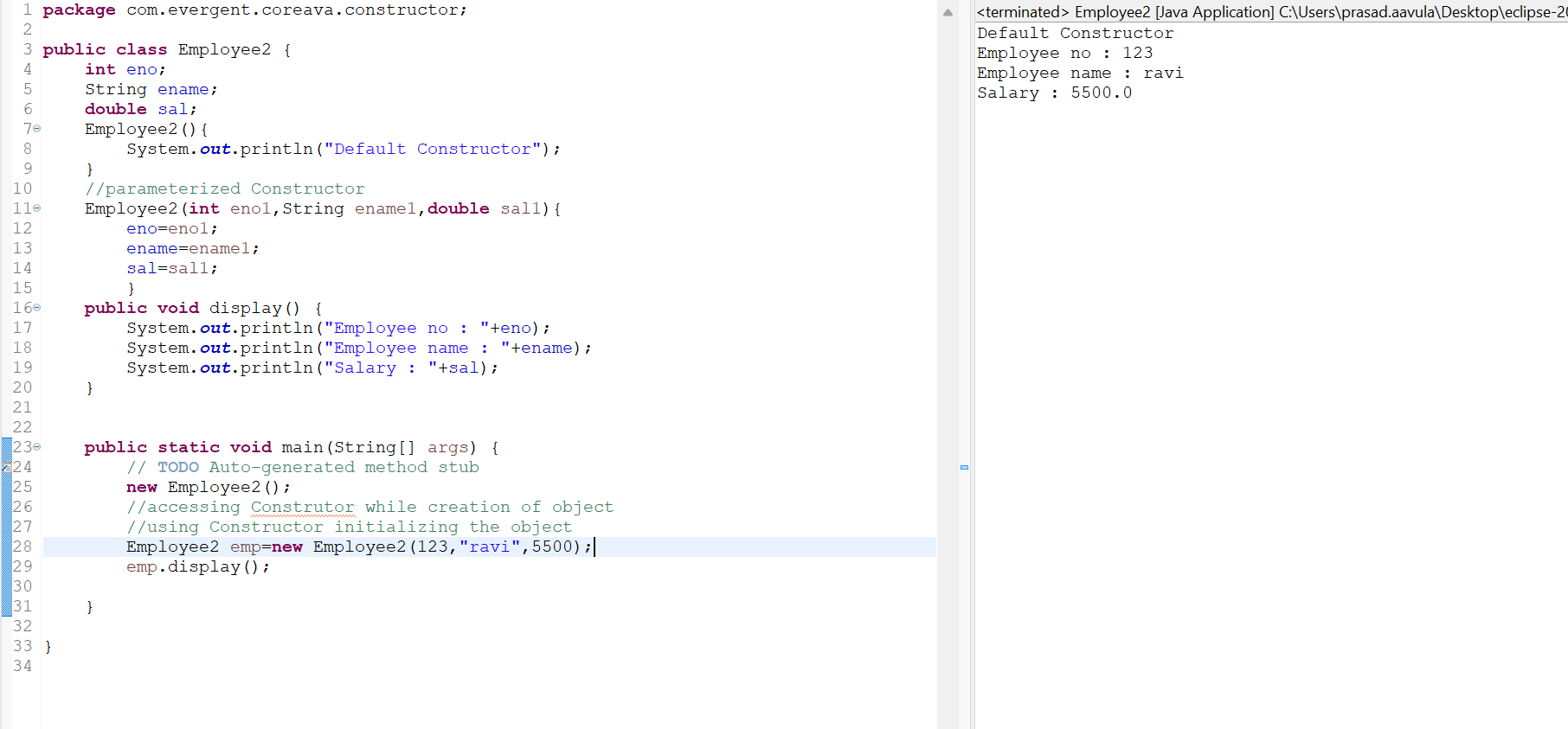


1. **Constructors -In Depth**
2. The Class name and the Constructor name should be same
3. There are two types of Constructors
4. Default Constructors
5. Parameterized Constructor
6. Object Copy Constructor
7. We can access the constructor while creation of object
8. Constructors are mainly for initialization of object
9. Constructors does not have any return type not even void also,If we give void it treats it as a method
10. Every class need atleast one default constructor.
11. The Class name and the Constructor name should be same
12. here are two types of Constructors, One is Default Constructor



And Other type is Parameterized Constructor

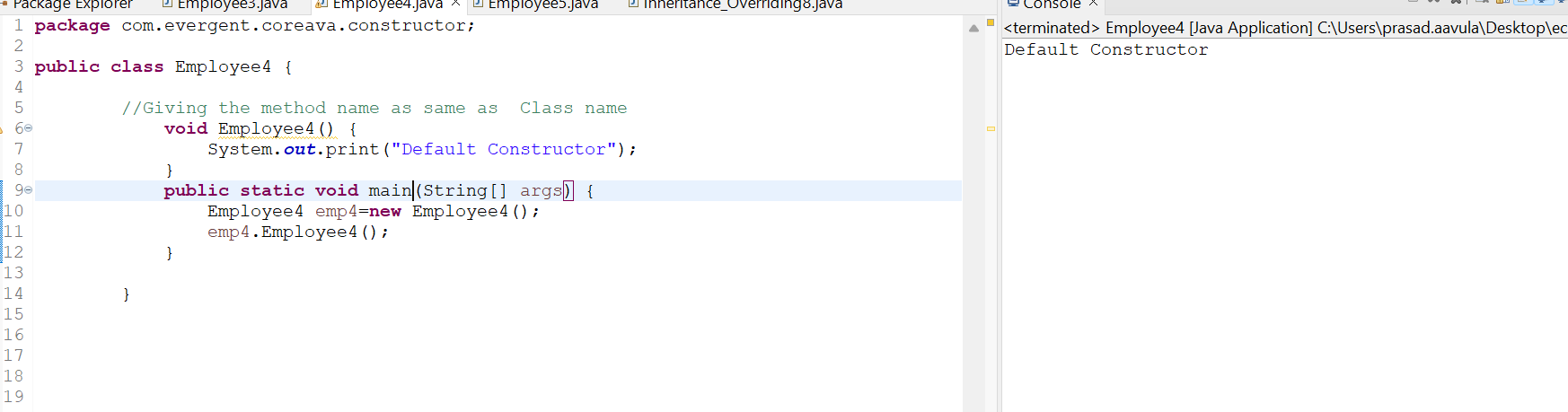
1. We can access the constructor while creation of object



1. Constructors are mainly used for initialization of objects



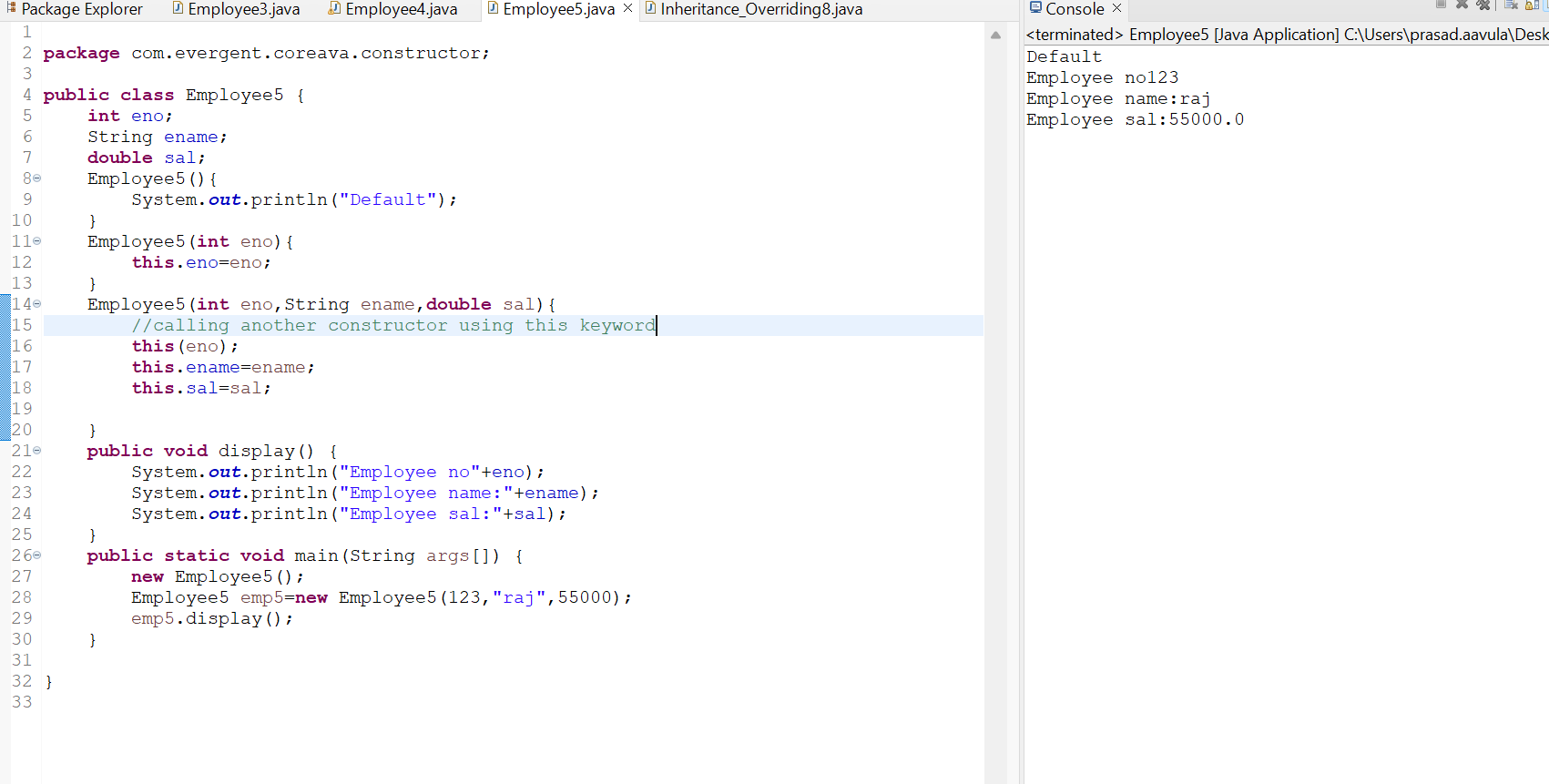
1. Constructors doesn’t have any return type and void,If we Declare as a void it will consider as normal method not a constructor



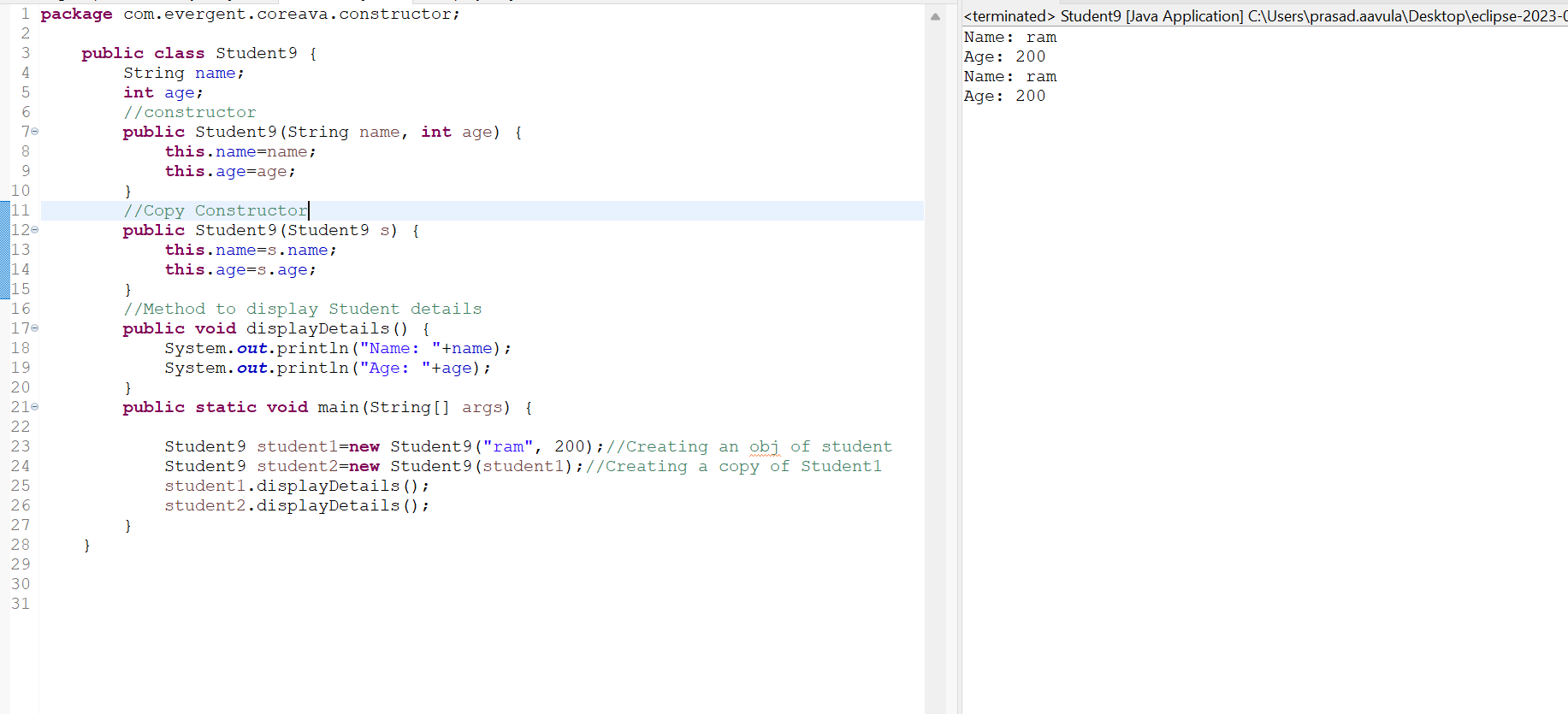
1. Every class need’s at least on default constructor either the user will create or default constructor created while object creation
2. Always Constructors are Overloaded



1. We can call one constructor in another constructor



1. The Other type of Constructor is CopyConstructor.copy constructor is a special type of constructor that creates an object using another object of the same Java class.



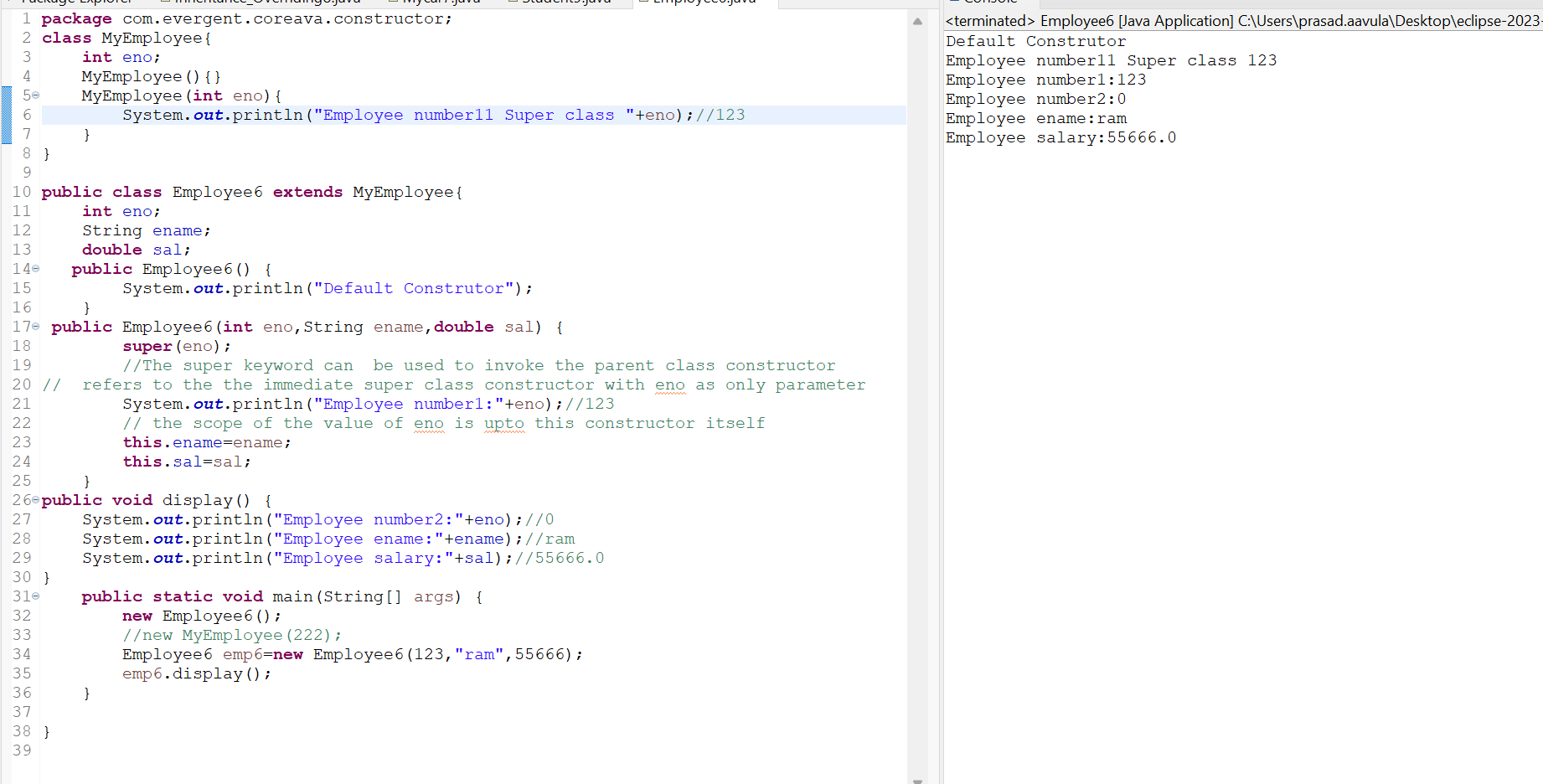
**This Keyword:-**

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

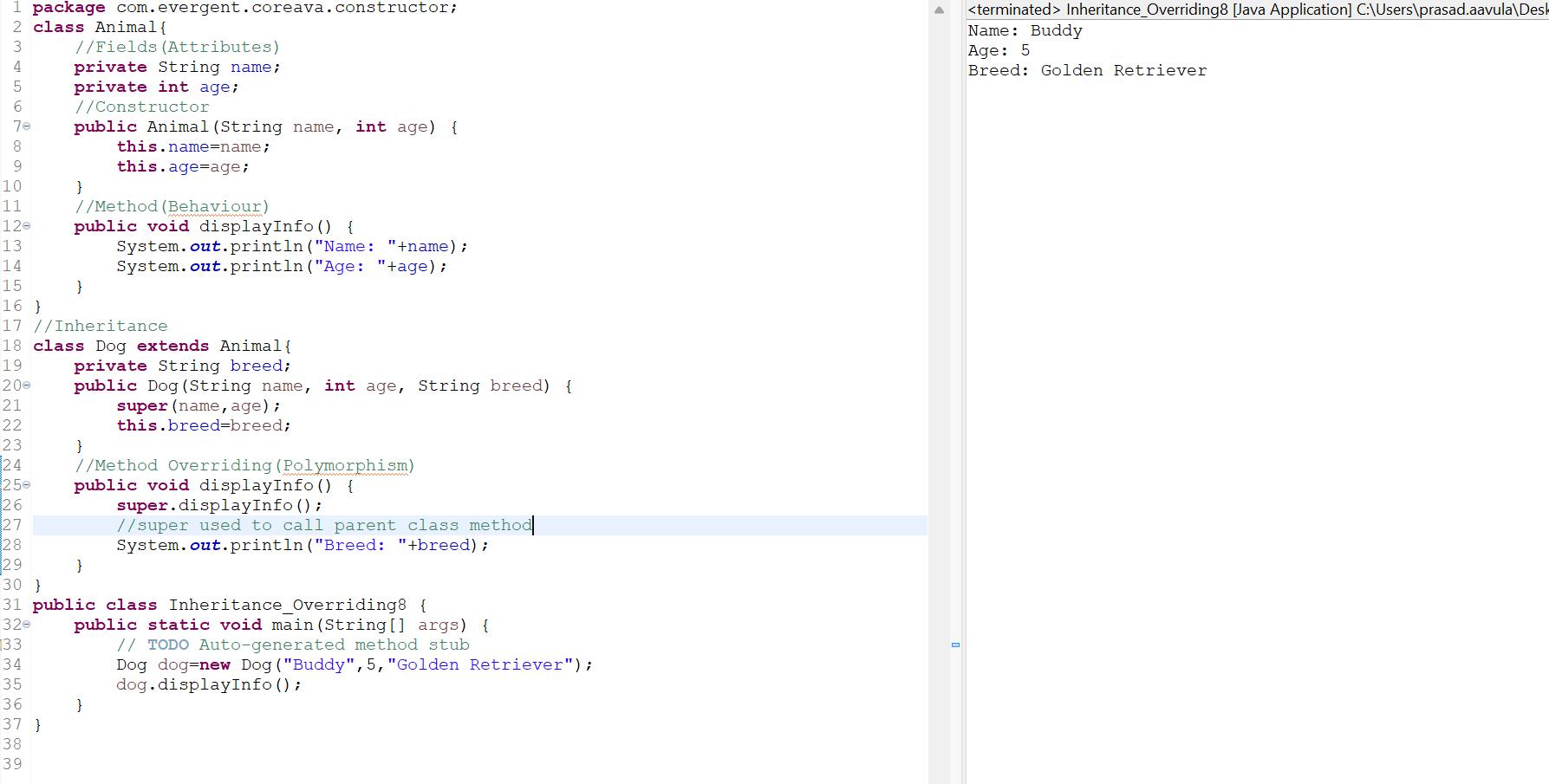


**Super Keyword:-**

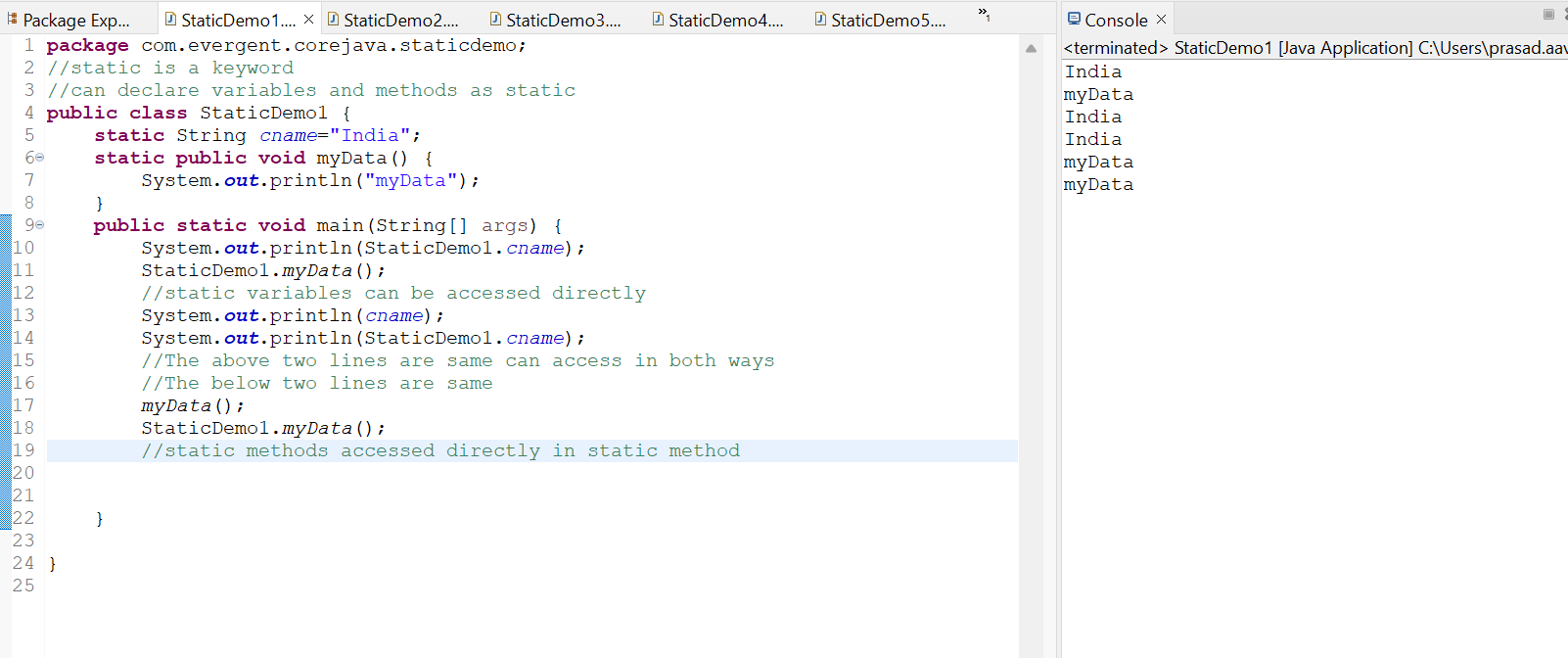
* Super is a keyword which is used to call the super class constructor from the base class constructors



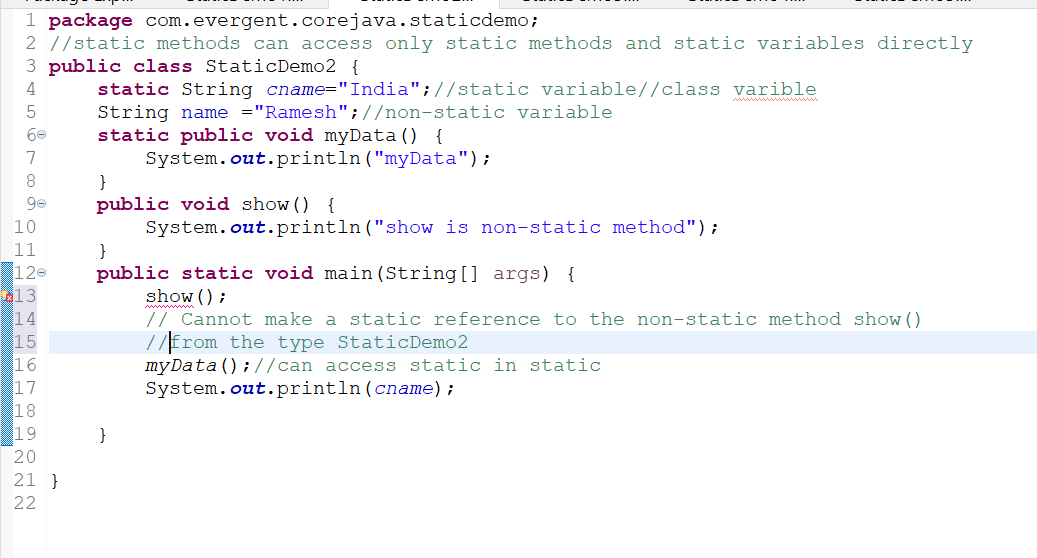
* The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.similarly to retrieve the parent class variable we can use super.variablename if and only if child class also has a varible with same name.

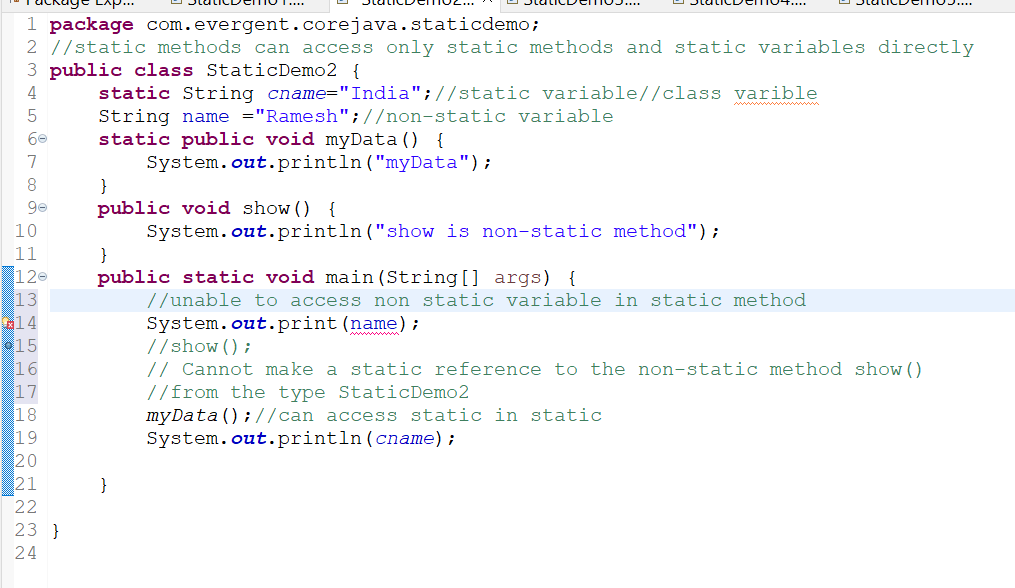


1. **static Keyword In Depth** **9-8-24 Day-5**
   1. static is keyword
   2. We can declare the static as variables and methods
   3. We can access the static methods and variables directly through className.method or className.variable.
   4. Static methods can access static variables and static methods only by directly.
   5. Static methods can not access non static methods and non static variables directly
   6. Non static methods can access static variables and static methods directly.
   7. static block:-When class is loaded in to JVM static block get initialized first
   8. static variables get reflected when one make changes in method (If we declare a variable as a static ,if modified the new value remains).
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

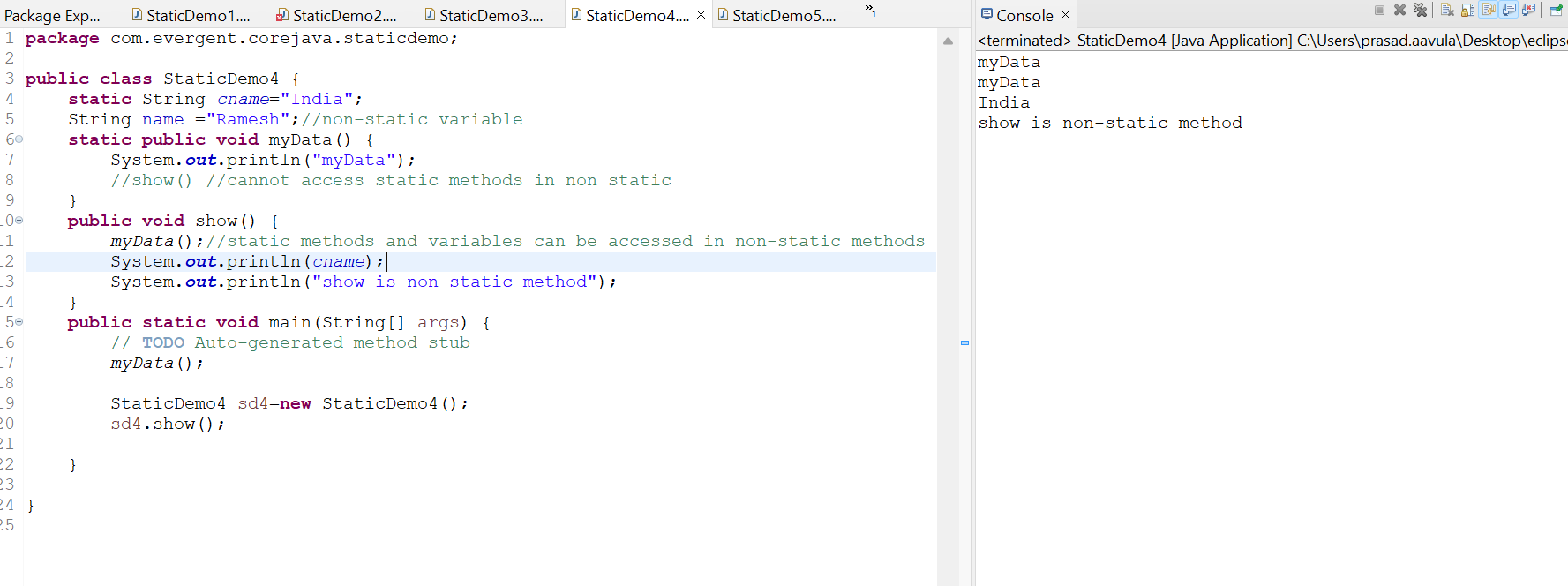


1. Static methods can access static variables and static methods only directly but not non static methods and non static variables.

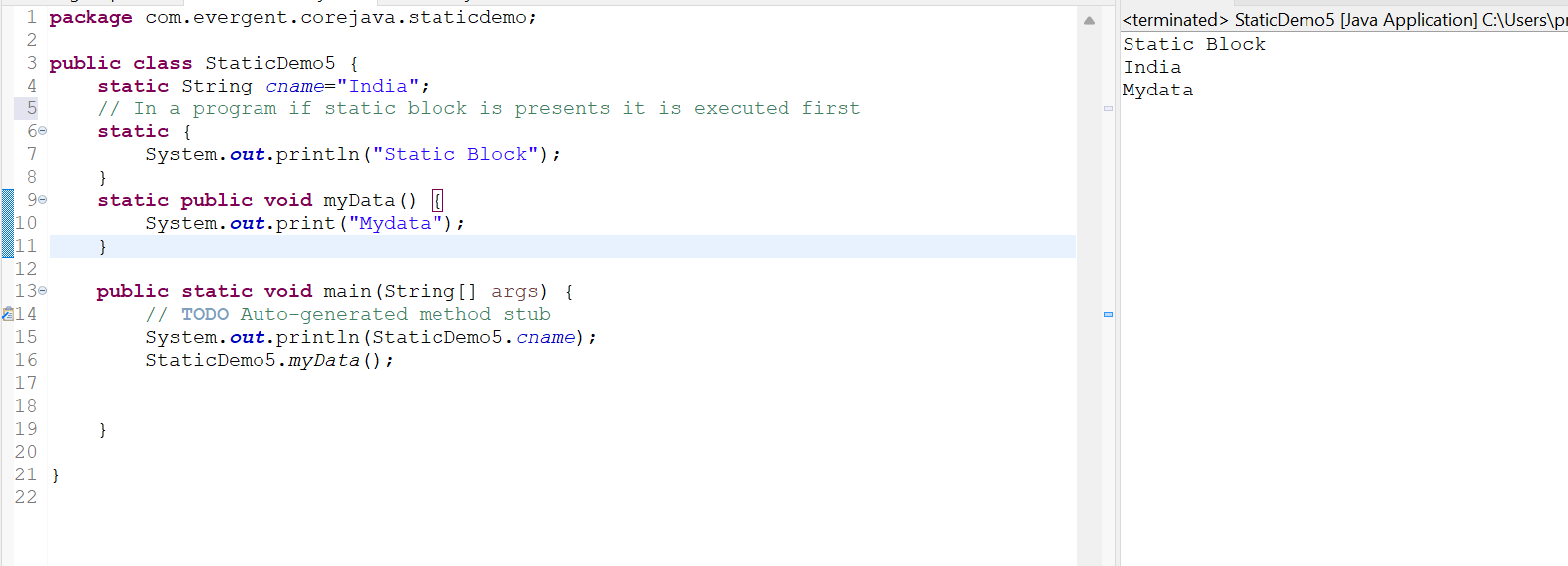




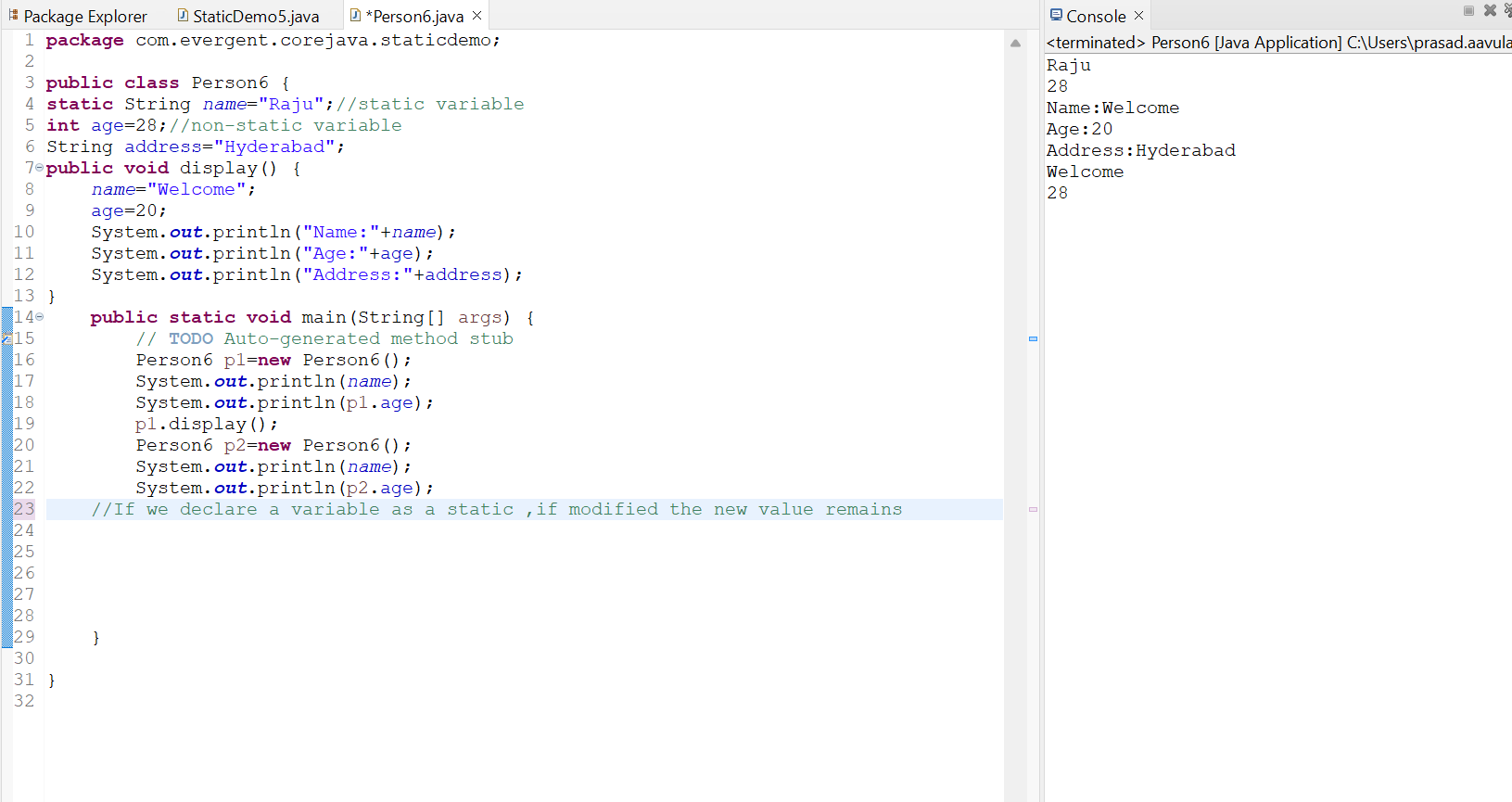
1. Non-static method can access the static methods and static variables directly



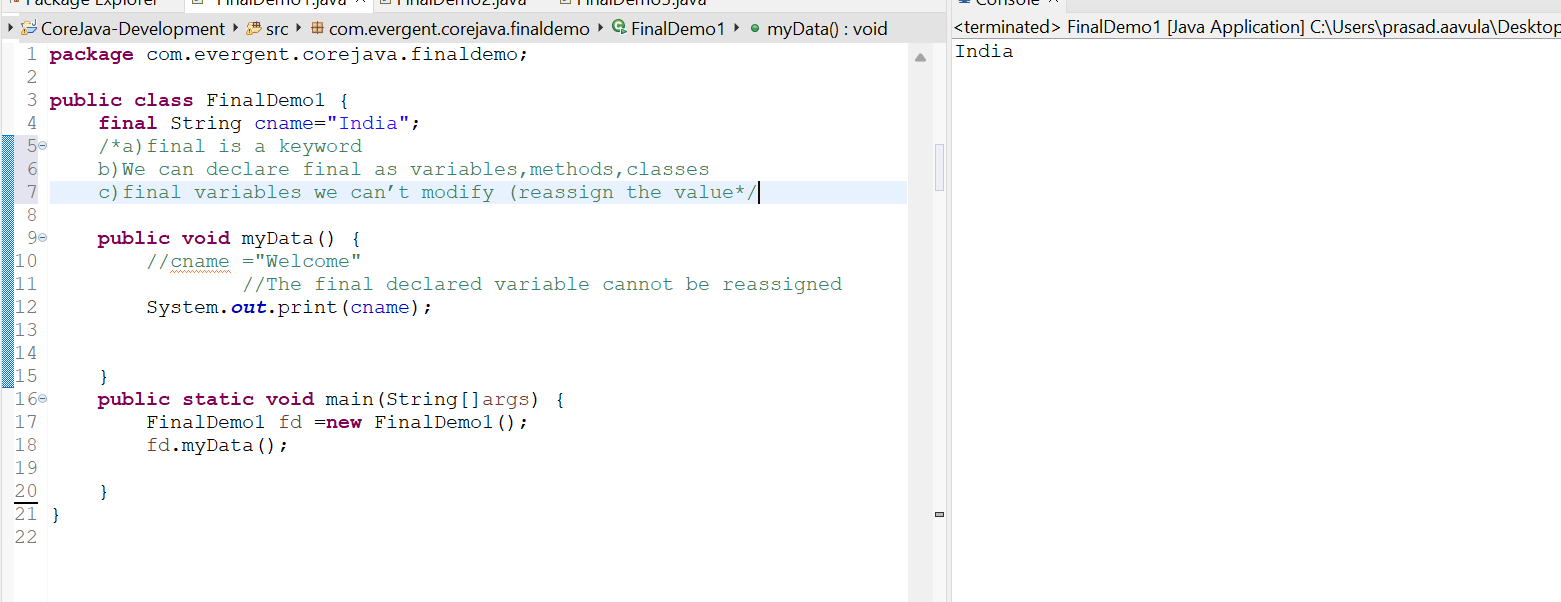
1. static block:-When class is loaded in to JVM static block get initialized first



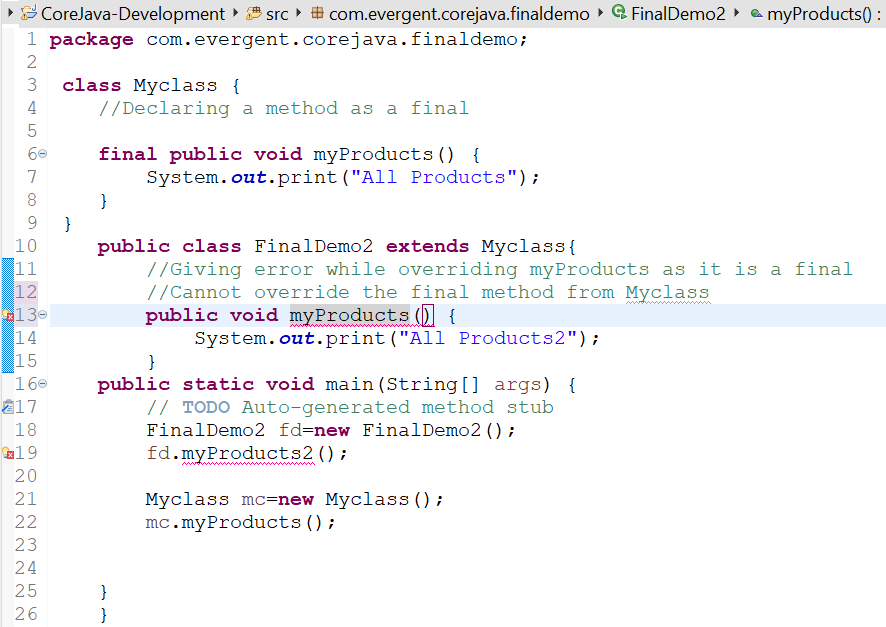
1. static variables get reflected when one make changes in method



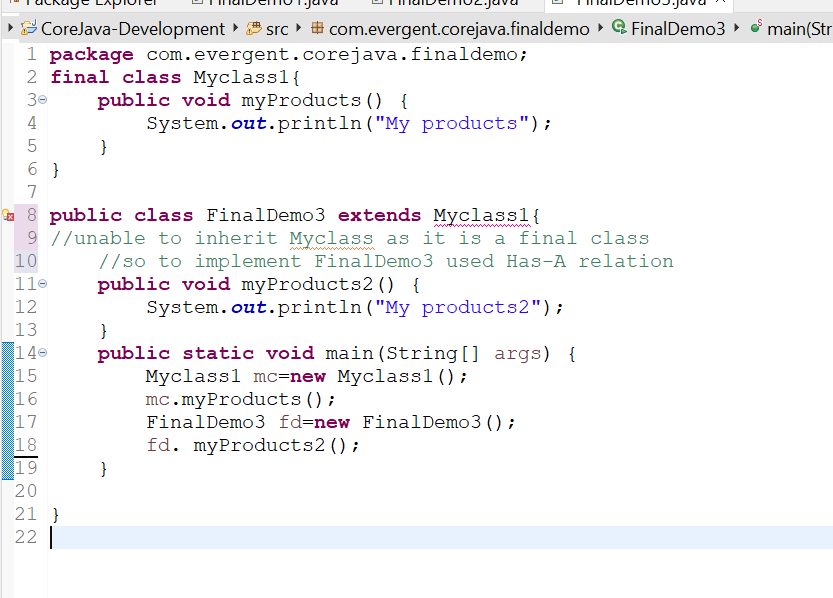
1. **final Keyword in Depth**
   1. final is a keyword
   2. We can declare final as variables,methods,classes
   3. final variables we can’t modify (reassign the value)
   4. final methods can’t be overridden
   5. Final class cannot be inherited
   6. Final variables can be initialized during constructor calling.
2. final is a keyword
3. We can declare final as variables,methods,classes
4. final variables we can’t modify (reassign the value)



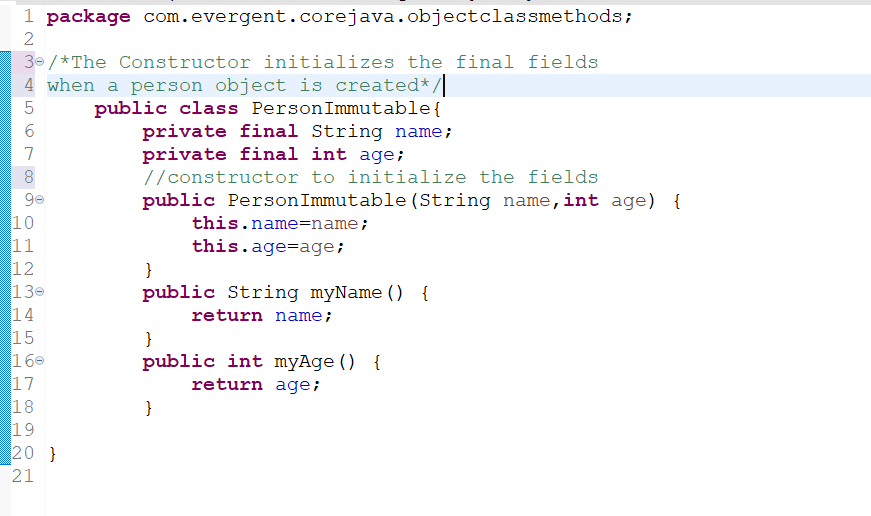
1. final methods can’t be overridden



1. final class cannot be inherited



1. final fields can be intialized during contructor calling.





1. **Strings** **12-8-24 Day-6**
   1. String class is final class
   2. String are immutable
   3. String methods are non-synchronized
   4. In java ,a string is a sequence of characters often used to represent text
   5. 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:-

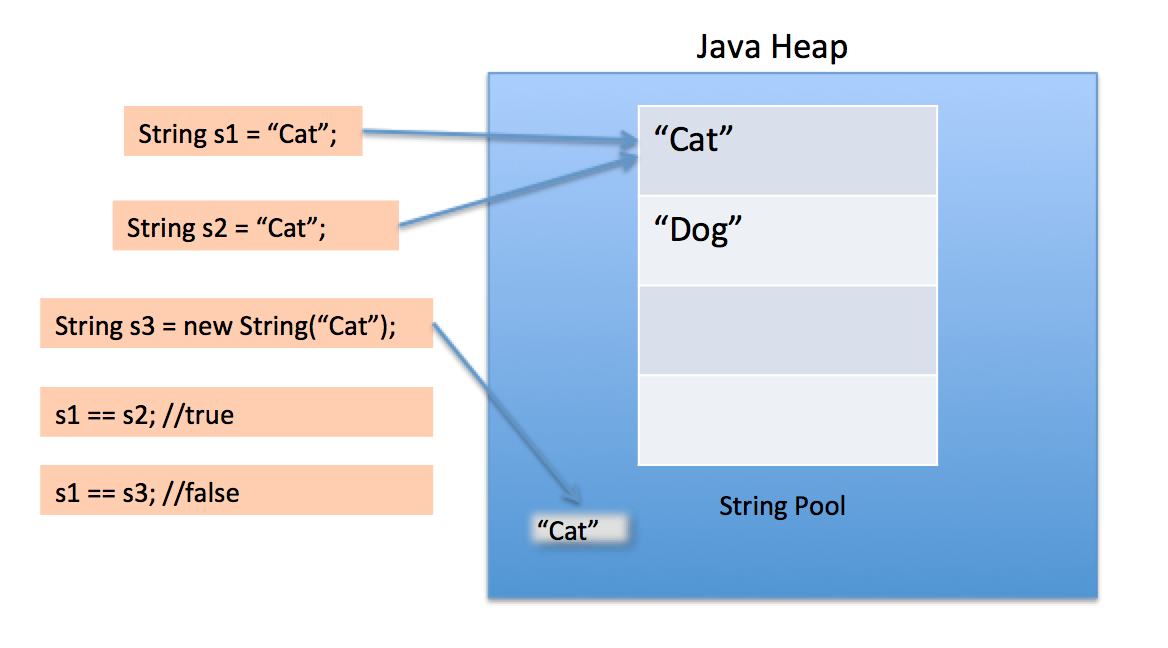
String name=new String(“Hello”);

name=”Welcome”;

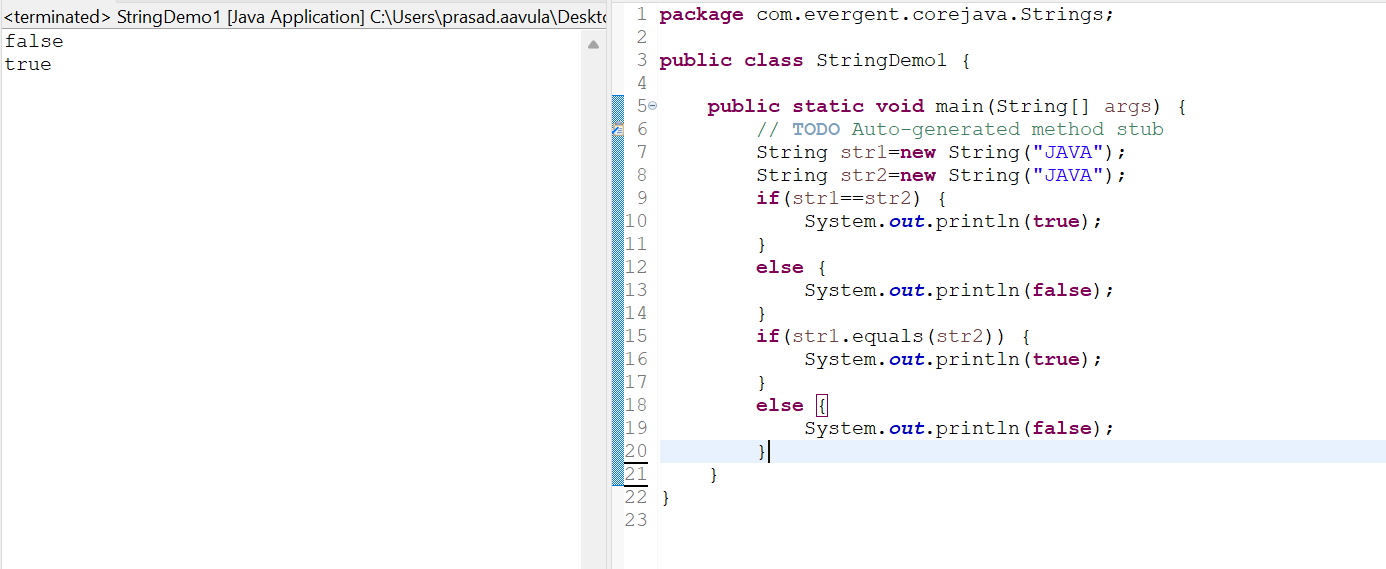
* 1. String objects in java and are instances of the string class ,which is part of the java.lang.package
  2. Key Features of strings in java :

Immutable:-Once a string object is created it can’t be modified Any modifications to a string creates a new string object

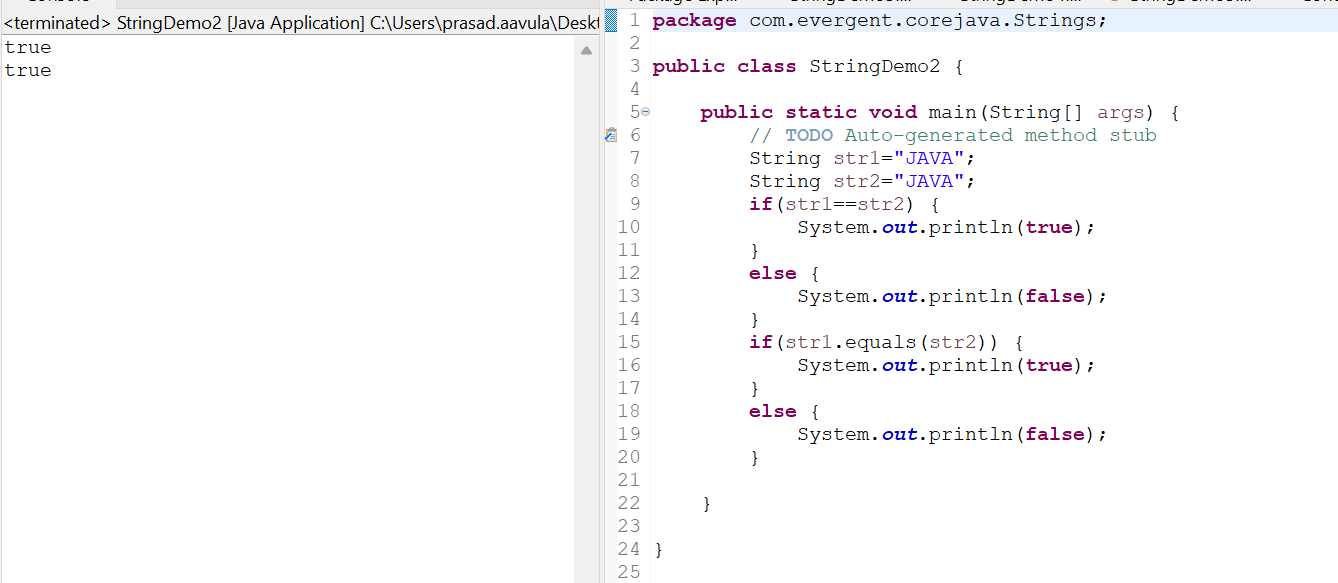
* 1. Java Optimizes memory usage by storing in s special area of memory known as the string pool
  2. If two Strings have the same value and are created without using new keyword,they will reference the same object in the string pool
  3. Storage of Strings in java done in two ways
     1. Heap Storage 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



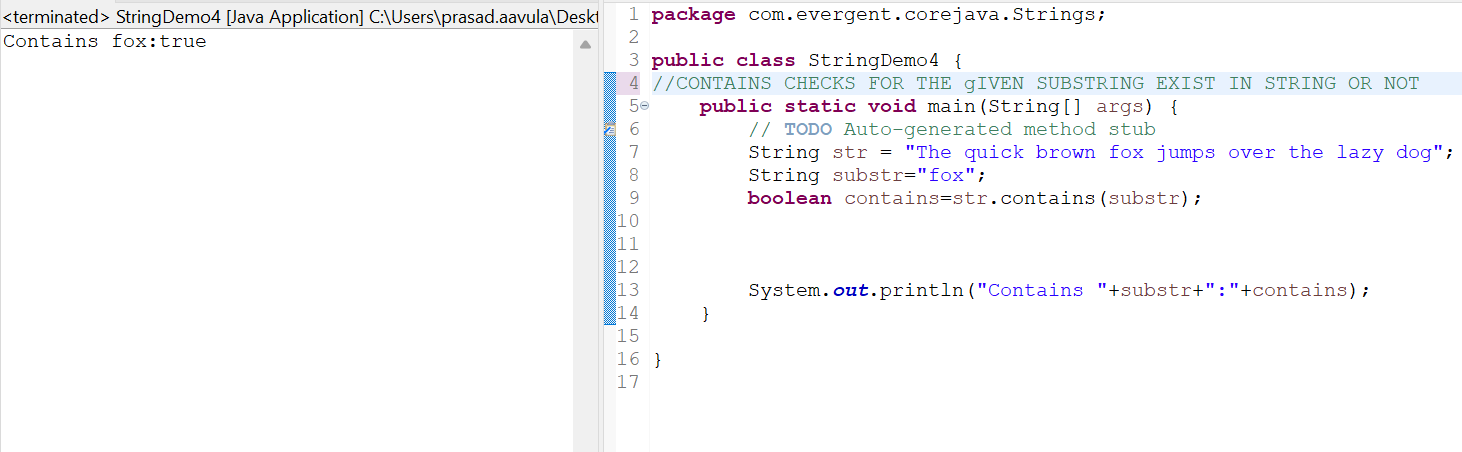
* 1. String Program without using the new keyword



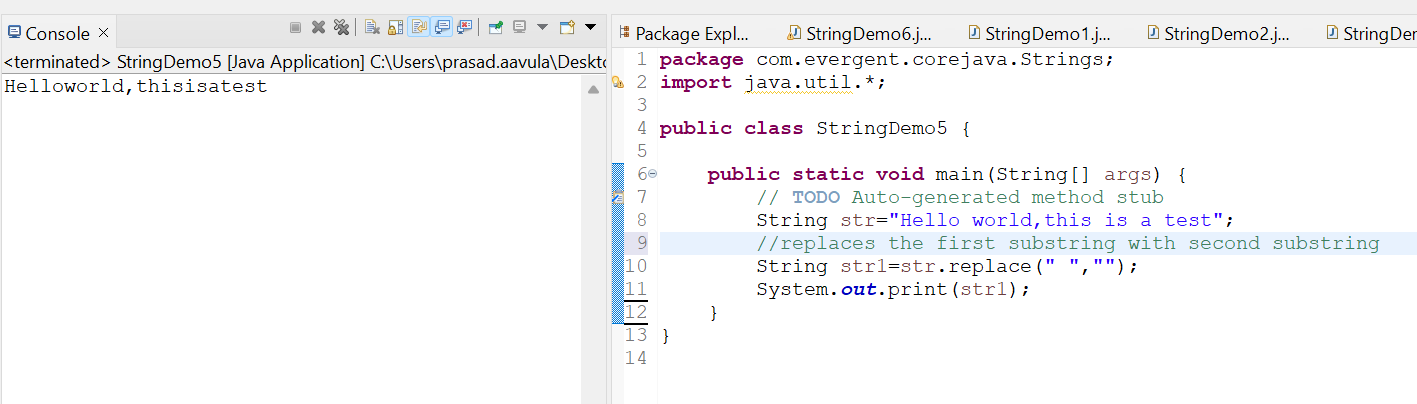
* 1. Different types of String methods include:
     1. Length();
     2. toLowerCase();
     3. toUpperCase();
     4. trim();
     5. startsWith();
     6. endsWith();
     7. equals();
     8. Contains();
     9. toCharArray();
     10. indexOf();
     11. lastIndexOf();
     12. subString();
     13. split();



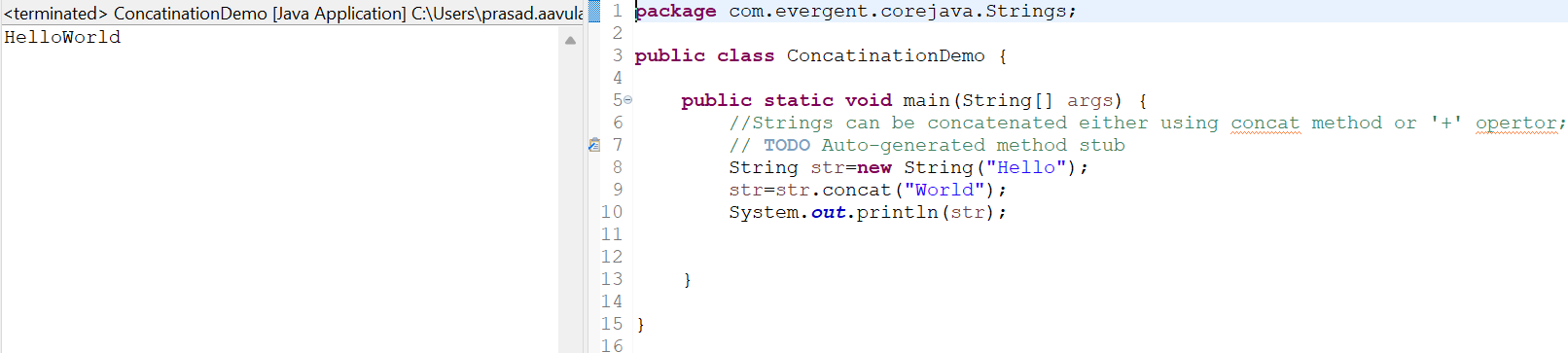
Contains():



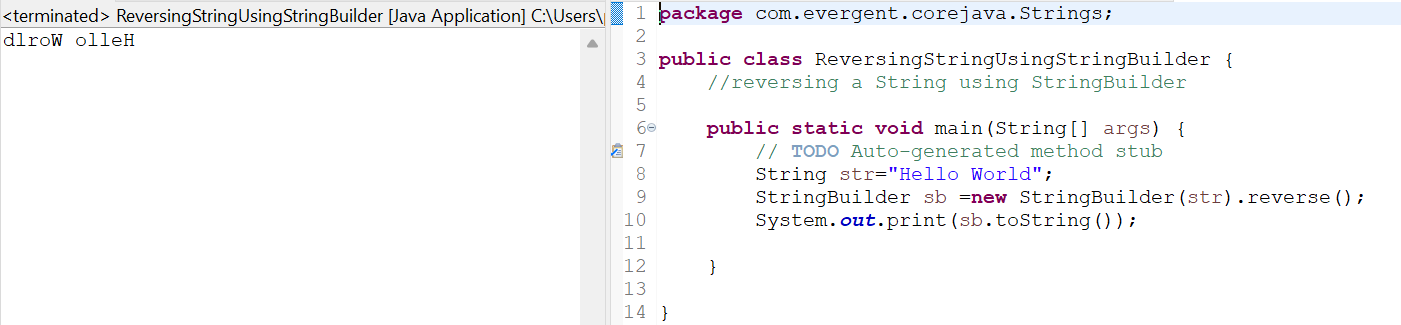
replace():



concat():



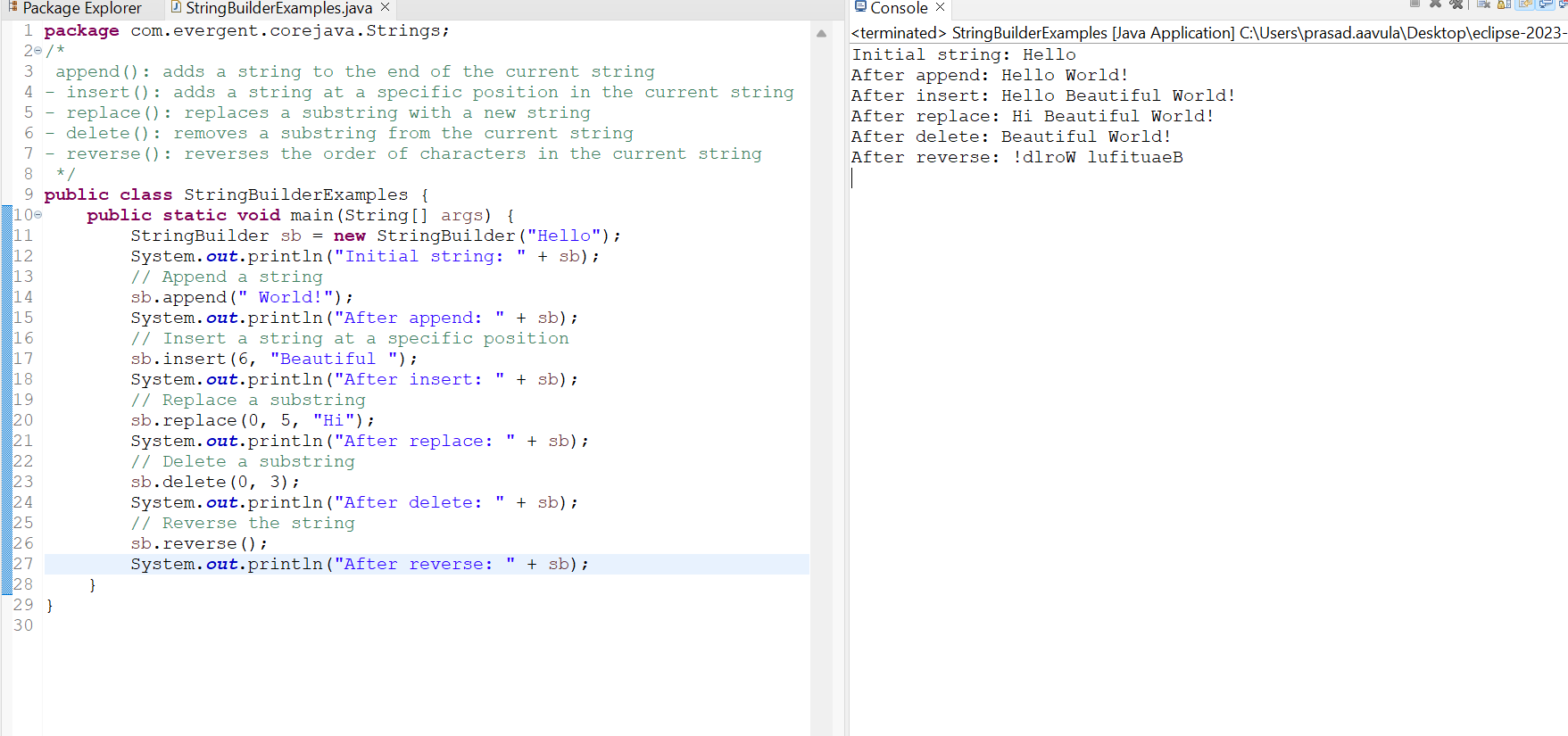
reverse():



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

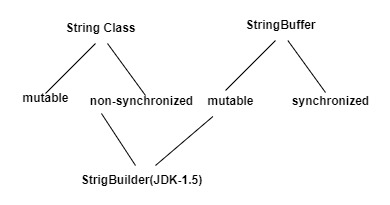


1. **StringBuilder:-**In Java,StringBuffer is a final class and mutable and it has some methods include
   1. Append:-adds a string to the end of the current string
   2. Insert:-adds a string at a specific position in the current string
   3. Replace:-replaces a substiring with new string
   4. Delete:-removes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

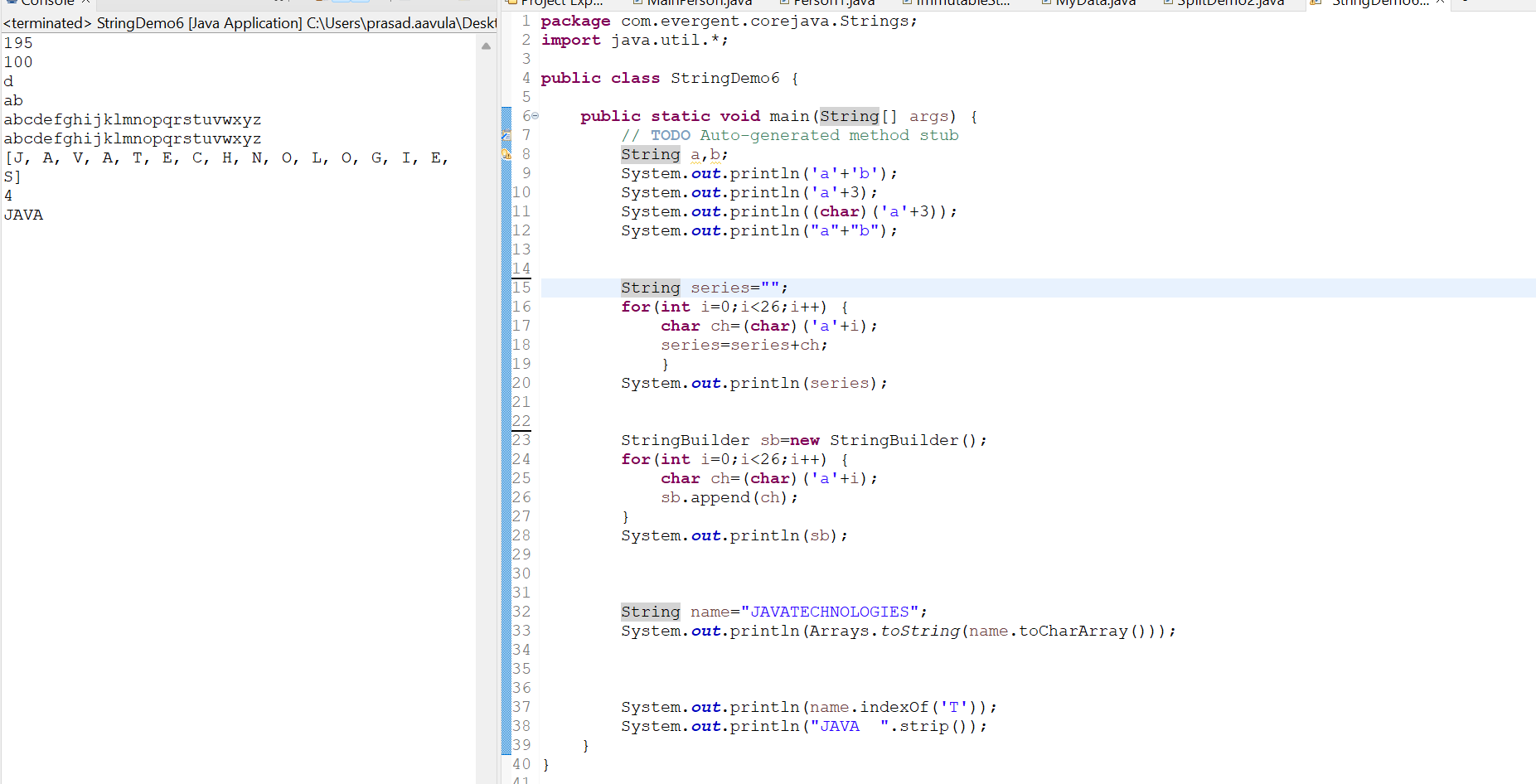


1. Difference Between **String ,StringBuffer,and StringBuilder**

|  |  |  |
| --- | --- | --- |
| **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**



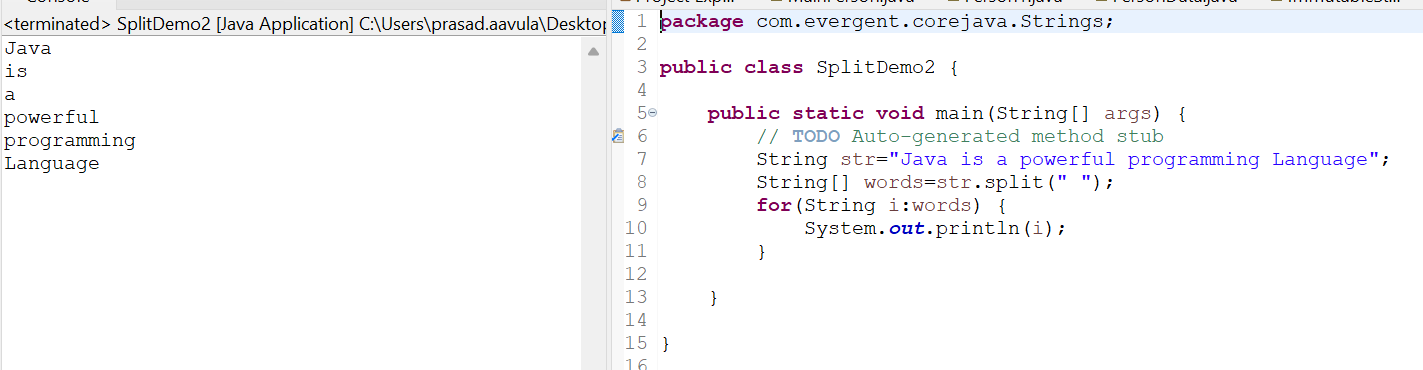
**13-8-24 Day-6**

1. Split method

The string split() method breaks a given string around matches of the given regular expression. After splitting against the given regular expression, this method returns a string array.

Generally split takes two parameters regex and limit

Limit has defuly value,if required if we can give



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 inherited

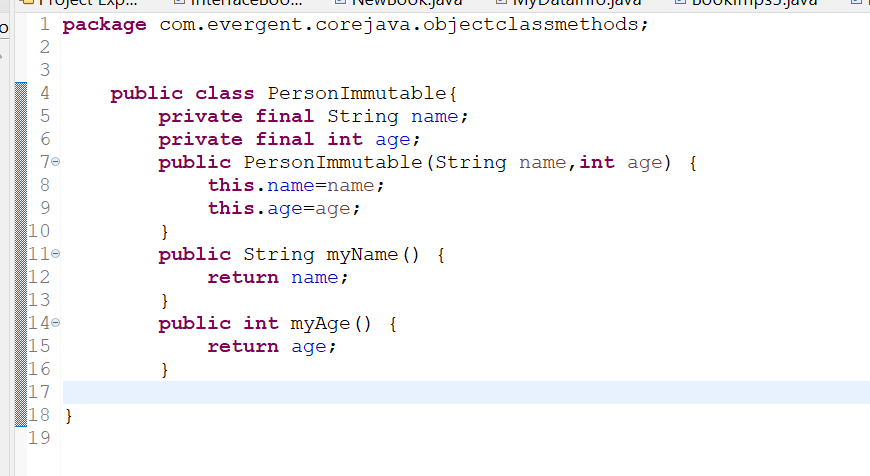
**Private Final Fields:-**

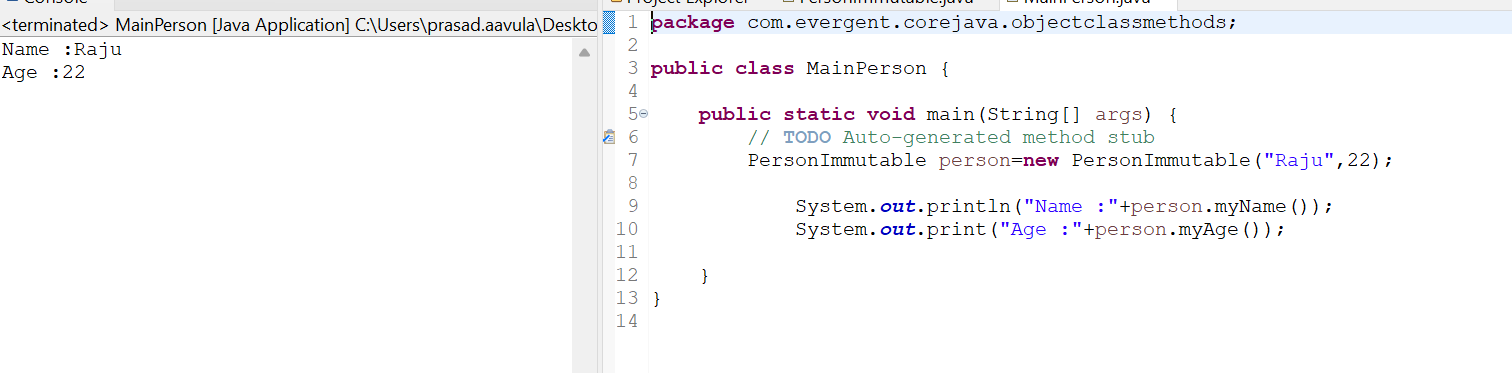
**-** We can declare class as final

- The field and age are private & final

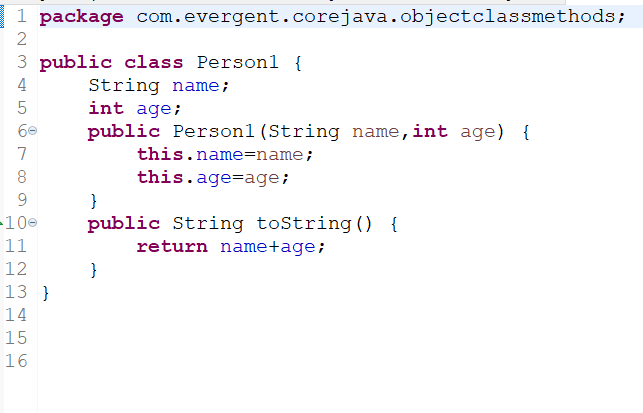
**Constructor:-**

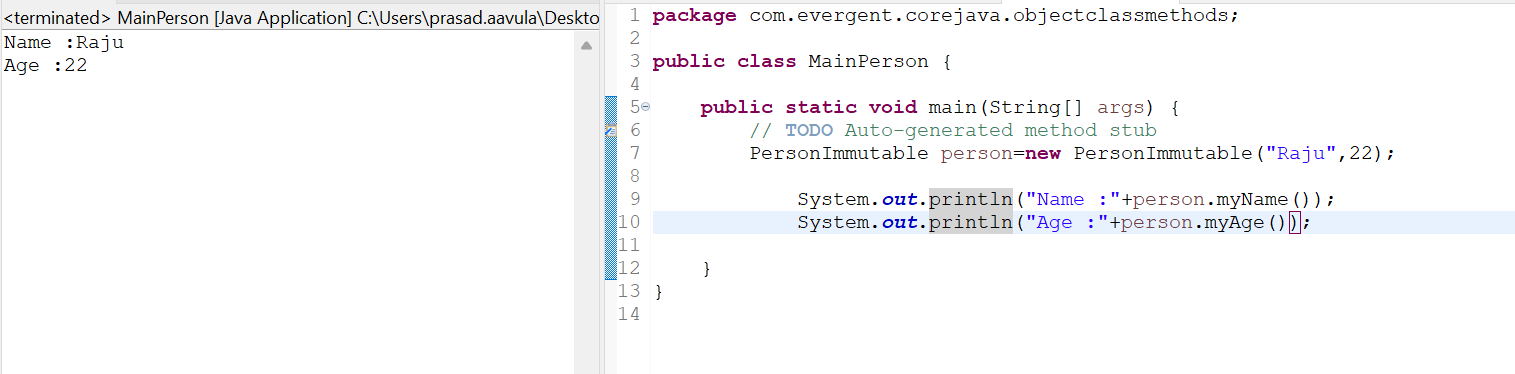
- The constructor initializes the final fields when a person object is created



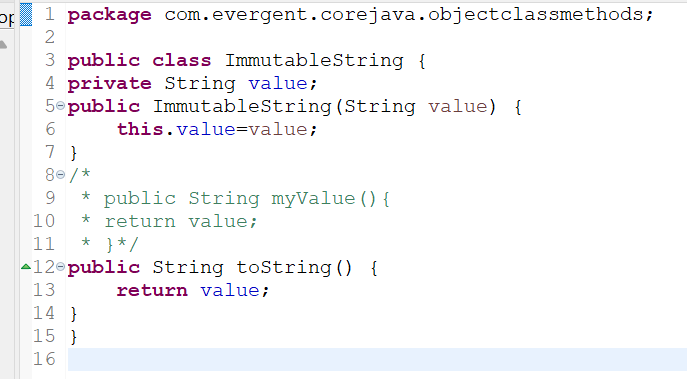


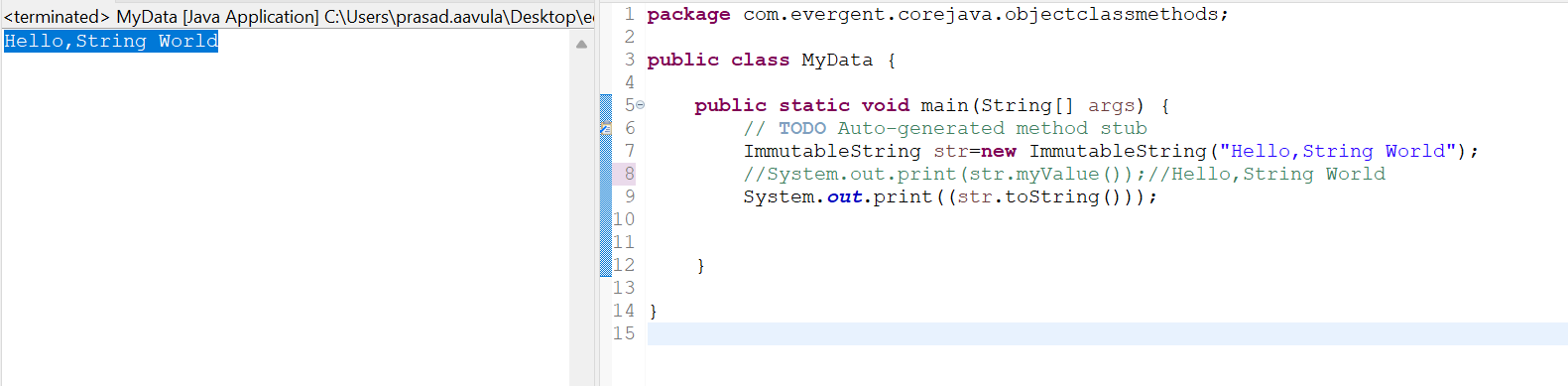
* Different ways of using toString





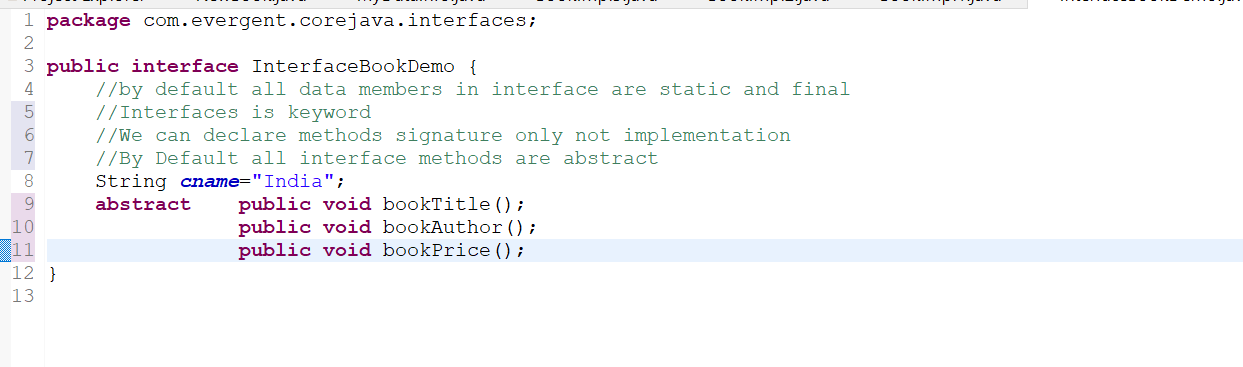
* we can use tostring or directly getter method to retrieve the value

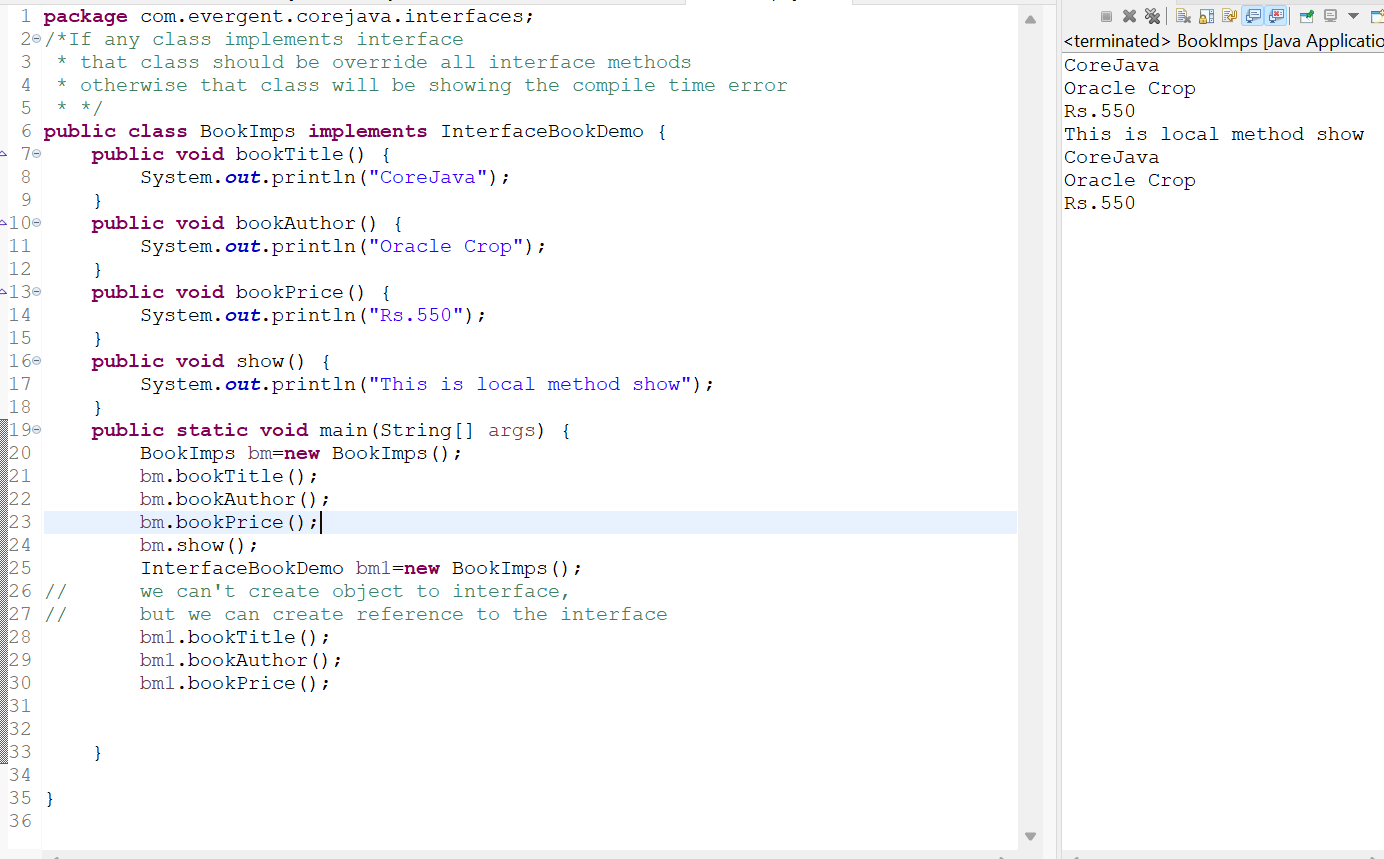




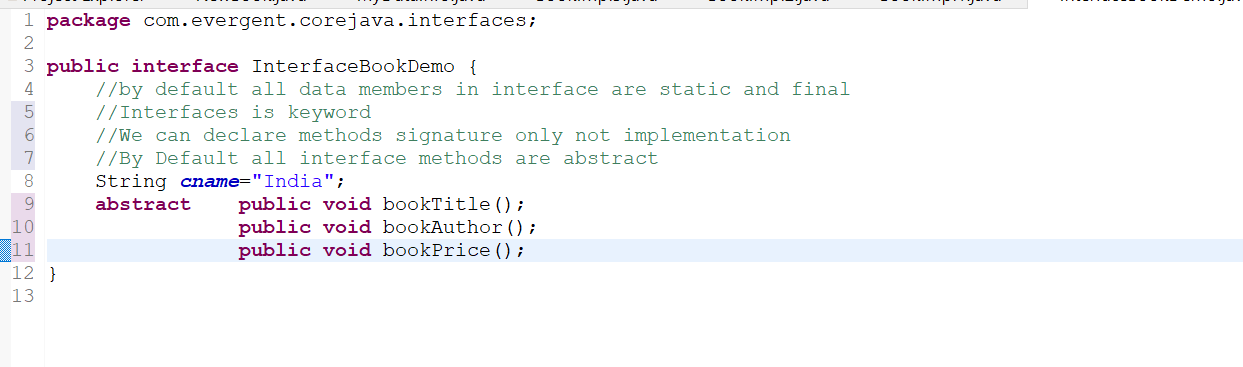
1. **Interface’s -In Depth**
   1. Interfaces is keyword
   2. We can declare methods signature but not implementation
   3. By Default all interface methods are abstract
   4. If any class implements interface that class should be override all interface methods otherwise that class will be shown with compile time error
   5. We can’t create object to interface,but we can create reference to the interface
   6. We can declare variables inside interfaces and all variables inside interfaces are final and static by default
   7. Java will support multiple inheritance through Interfaces
   8. One class can implements interfaces
   9. One Interface can extends other interfaces.

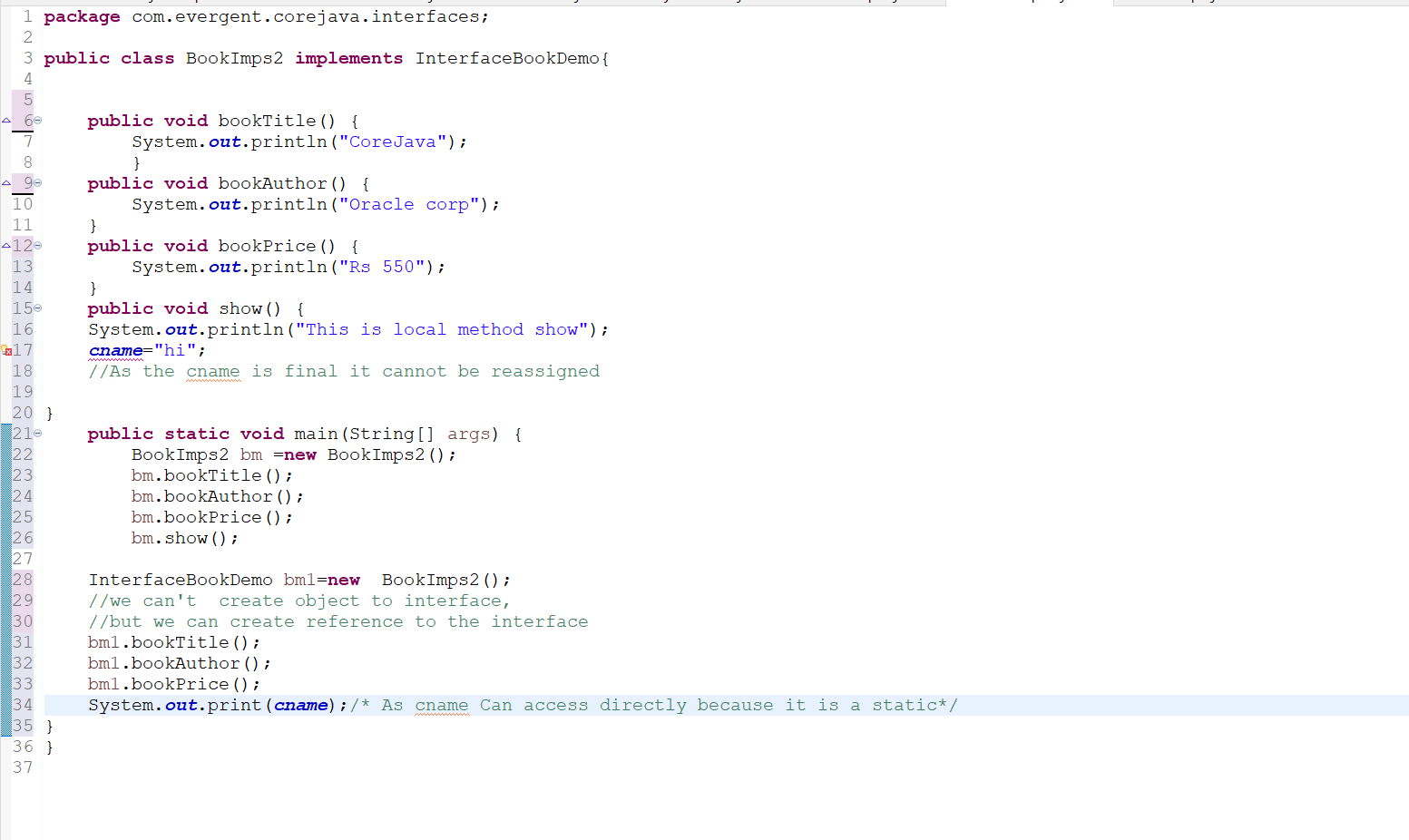
In below interface given only method signatures but not implementation





As all data members in interface are static and final by default

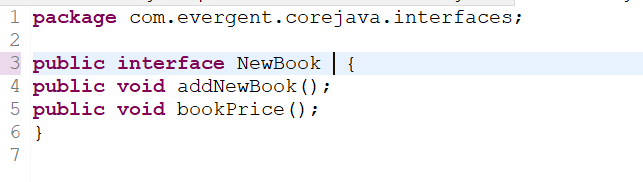


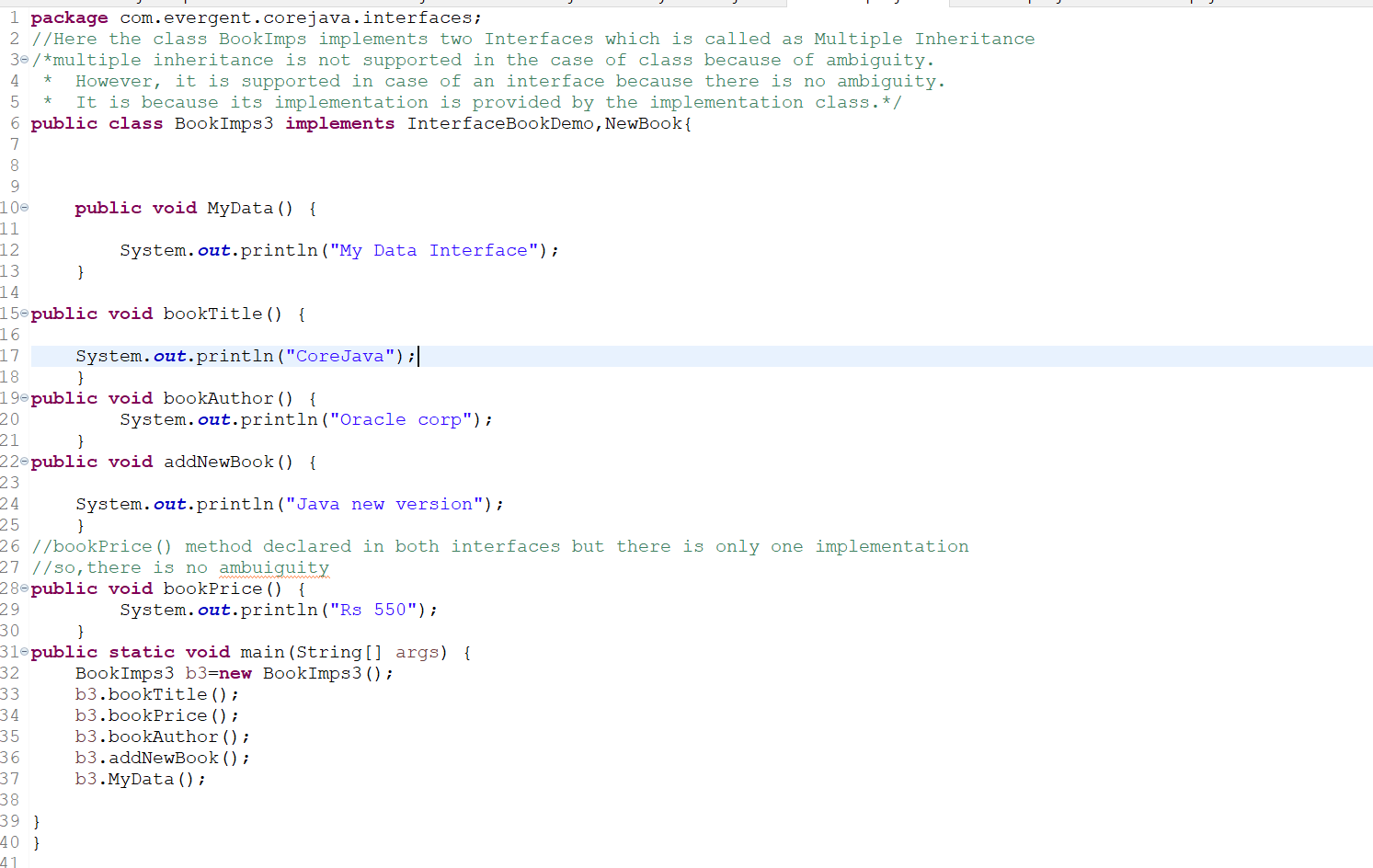


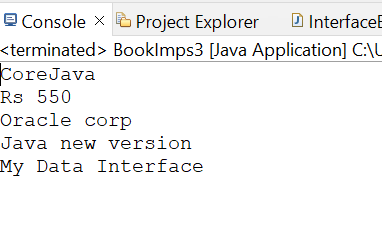
**we can't access the local methods with object reference of the interface**



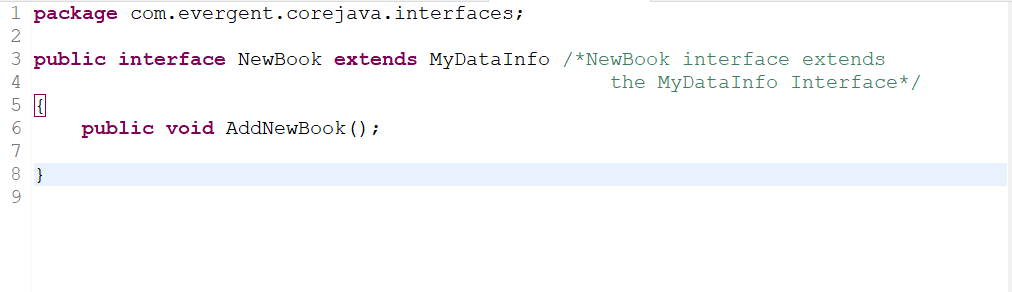
* 1. Java will support multiple inheritance through interfaces



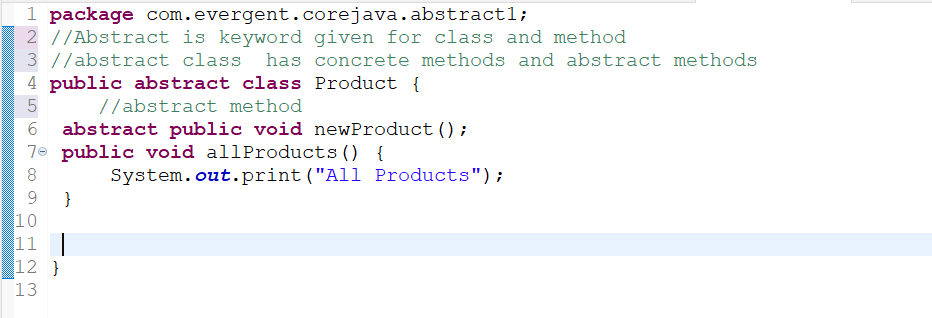




* 1. One class can implements interfaces
  2. One interface can extend other interfaces

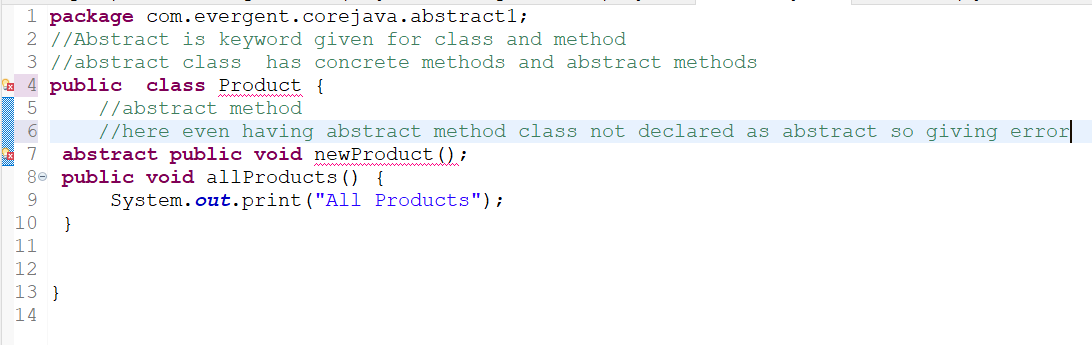


1. **Abstract Classes:**
2. Abstract is a Keyword
3. Abstract classes have abstract methods and concrete (Implemented ) methods
4. If any class having one abstract then that class to be declared as abstract keyword otherwise It it shows compile time error
5. If any class extends abstract class then that class should be override all abstract methods otherwise the class will show show compile time error.
6. We cannot create object for abstract class but we can create reference to abstract class.

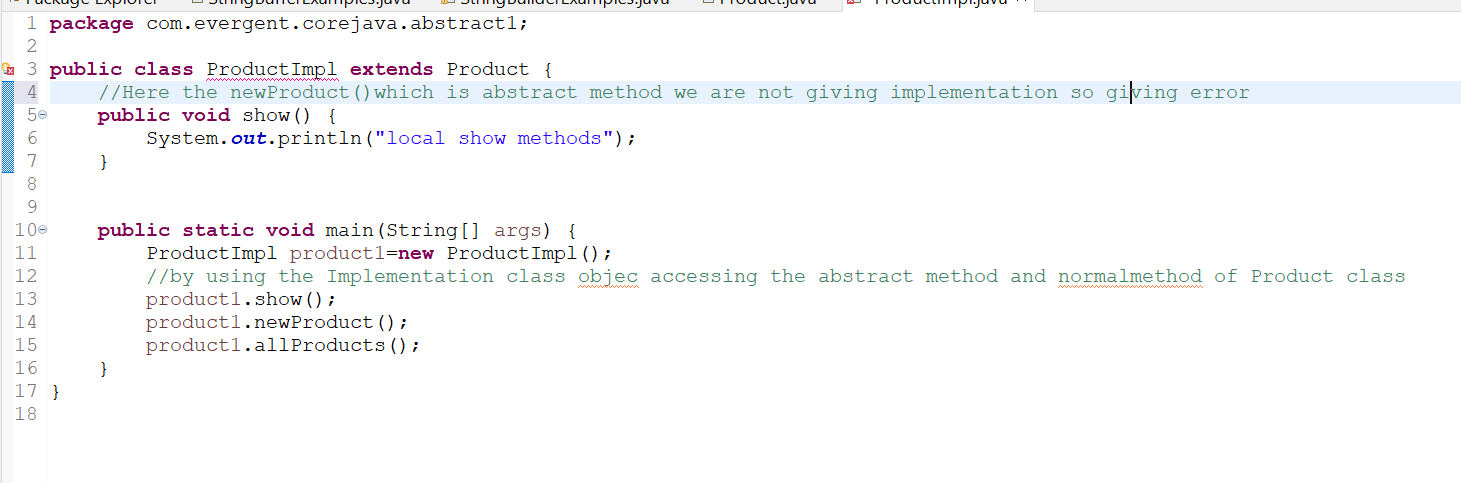




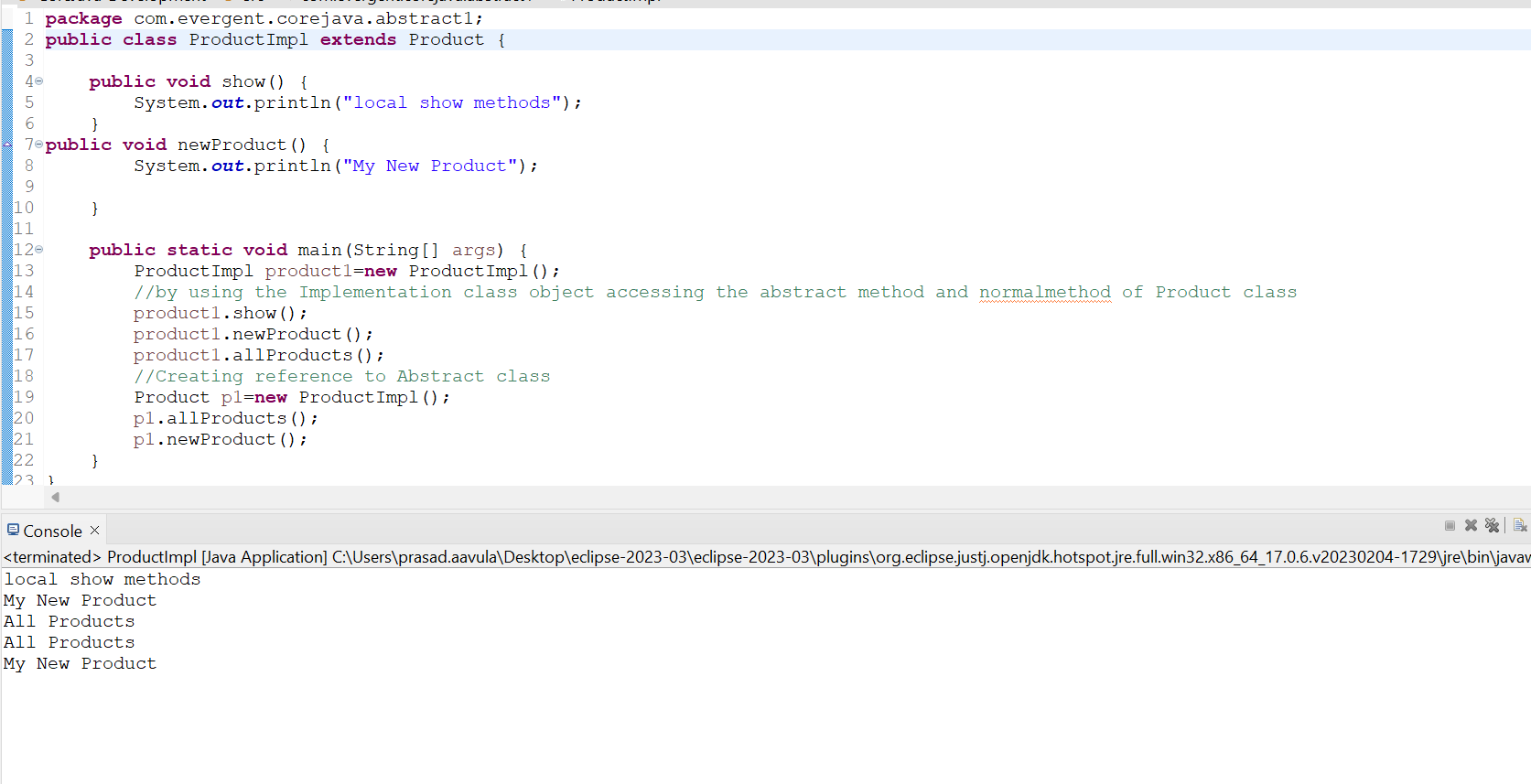
If any class having one abstract then that class to be declared as abstract keyword otherwise It it shows compile time error



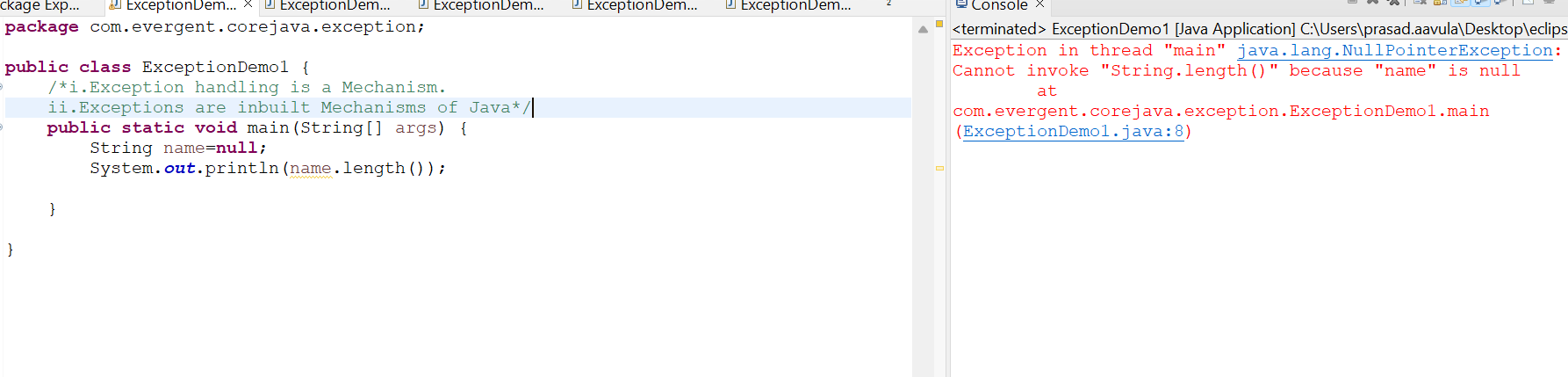
If any class extends abstract class then that class should be override all abstract methods otherwise the class will show show compile time error.



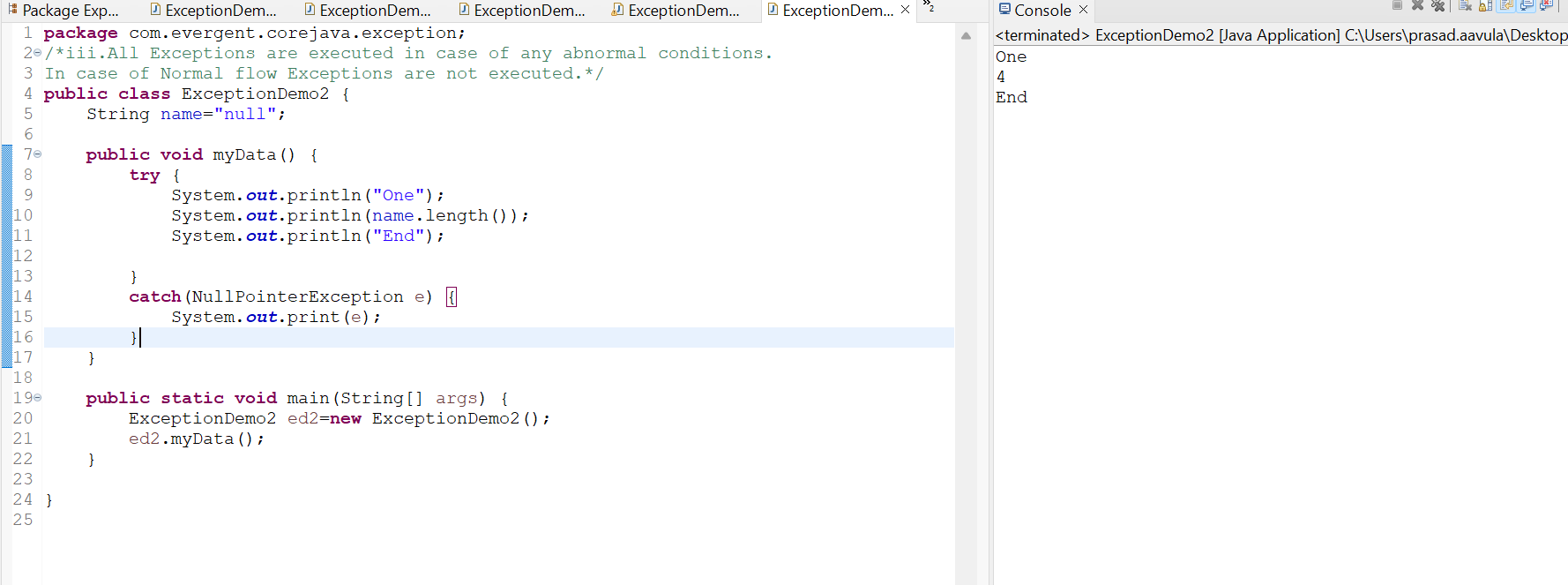
We cannot create object for abstract class but we can create reference to abstract class.



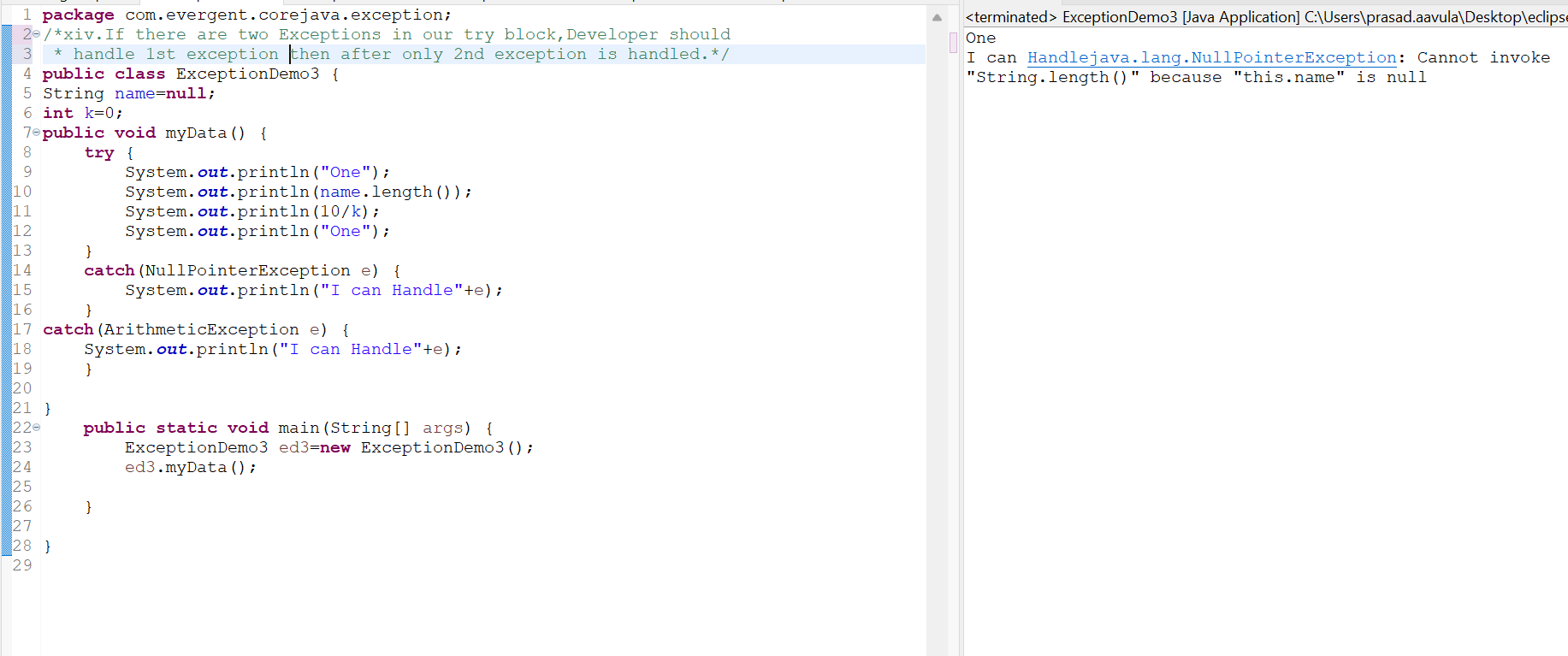
1. **Exception Handling:**
2. Exception handling is a Mechanism.
3. Exceptions are inbuilt Mechanisms of Java
4. All Exceptions are executed in case of any abnormal conditions.
5. In case of Normal flow Exceptions are not executed.
6. Once any Exception catched in java,the remaining lines of code is unreachable.
7. java.lang.Throwable is super class for Exceptions and Errors.
8. There are two types of Exceptions in java:
9. Checked Exceptions
10. UnChecked Exceptions
11. All Checked Exceptions are compile time Exceptions
12. All Unchecked Exceptions are Runtime Exception
13. There are five keywords in Exception Handling:
14. Try
15. Catch()
16. Finally
17. throws
18. throw
19. try is for business logic
20. Catch for handling Exceptions.
21. Finally is a block,irrespective of the Exception finally block is executed.
22. Using throws an Exception can be executed method by method.
23. Throw is for runtime exceptions.it calls predefined Exception classes.
24. try followed by either catch block or finally block
25. We should follow Exception hierarchical.
26. We can create our own(user defined) exceptions.
27. UserDefined Exceptions extends Exception or Runtime Exception.
28. All Exception are into java.lang.packages.
29. If there are two Exceptions in our try block,Developer should handle 1st exception then after only 2nd exception is handled.
30. We can Exceptions but not Errors.
31. Exception handling is a Mechanism.
32. Exceptions are inbuilt Mechanisms of Java



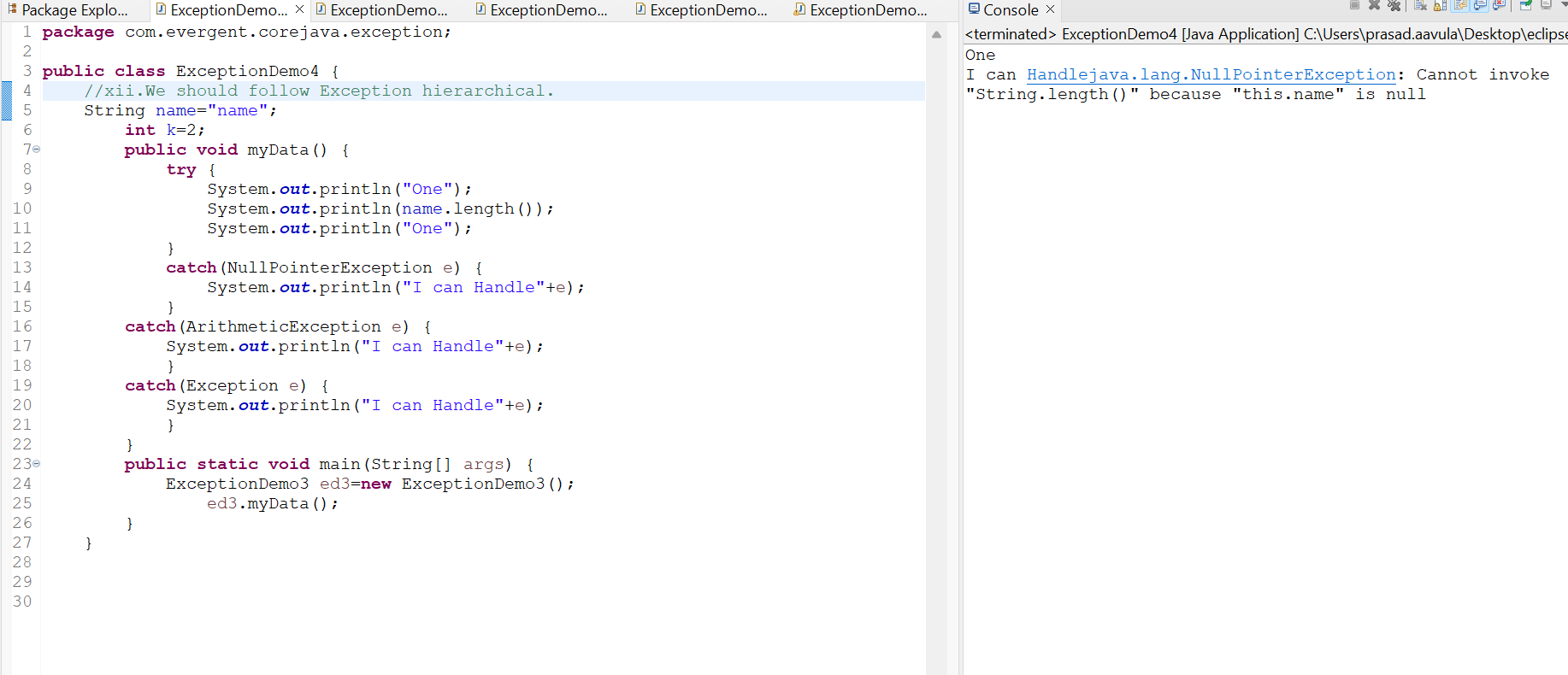
1. All Exceptions are executed in case of any abnormal conditions.
2. In case of Normal flow Exceptions are not executed.



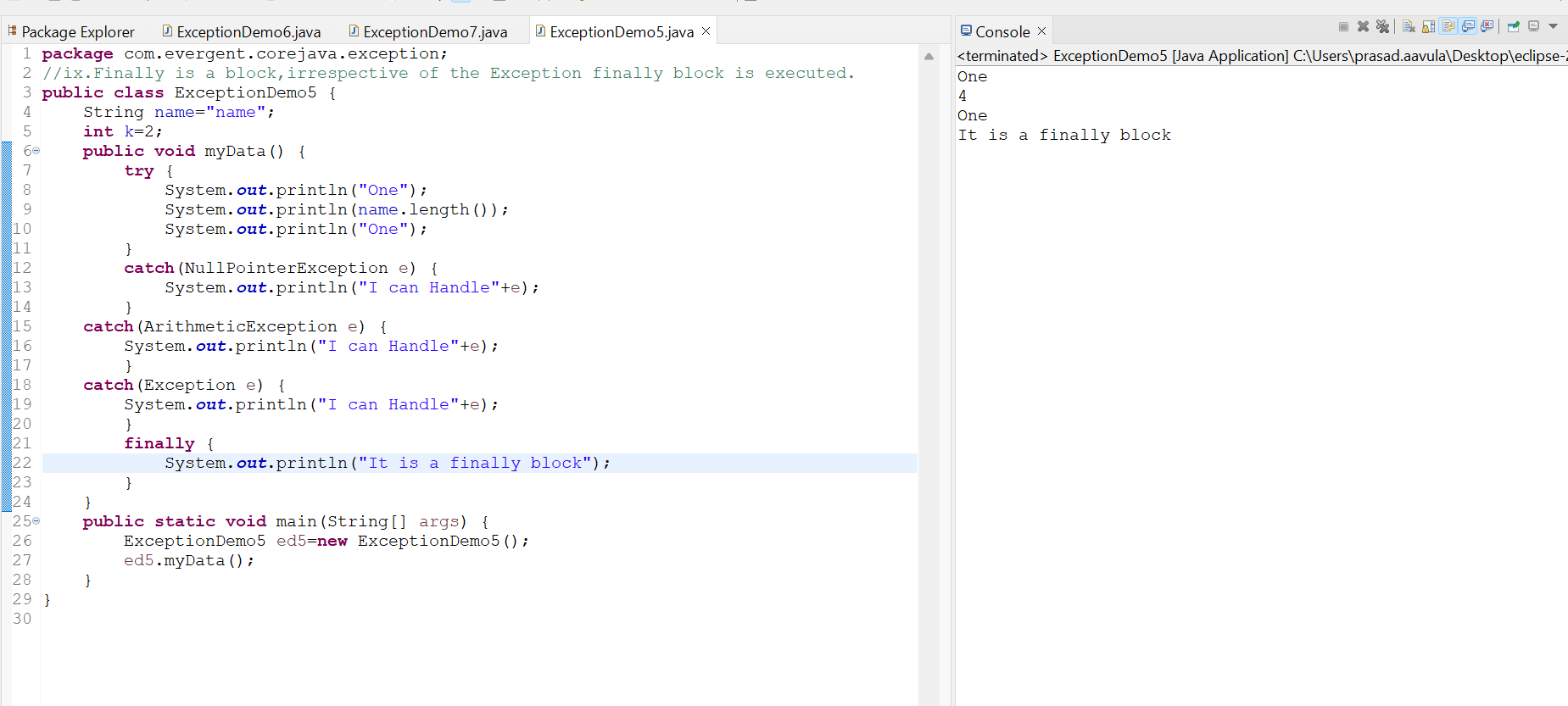
If there are two Exceptions in our try block,Developer should handle 1st exception then after only 2nd exception is handled.



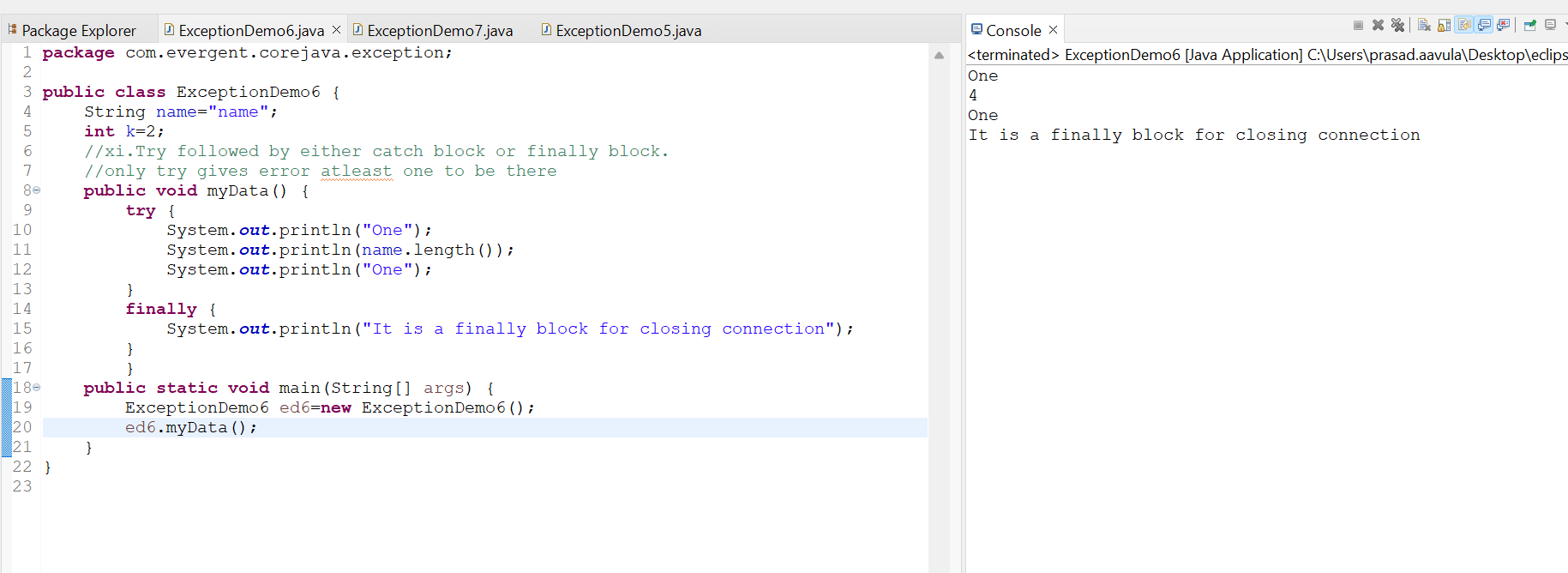
We should follow Exception hierarchical.



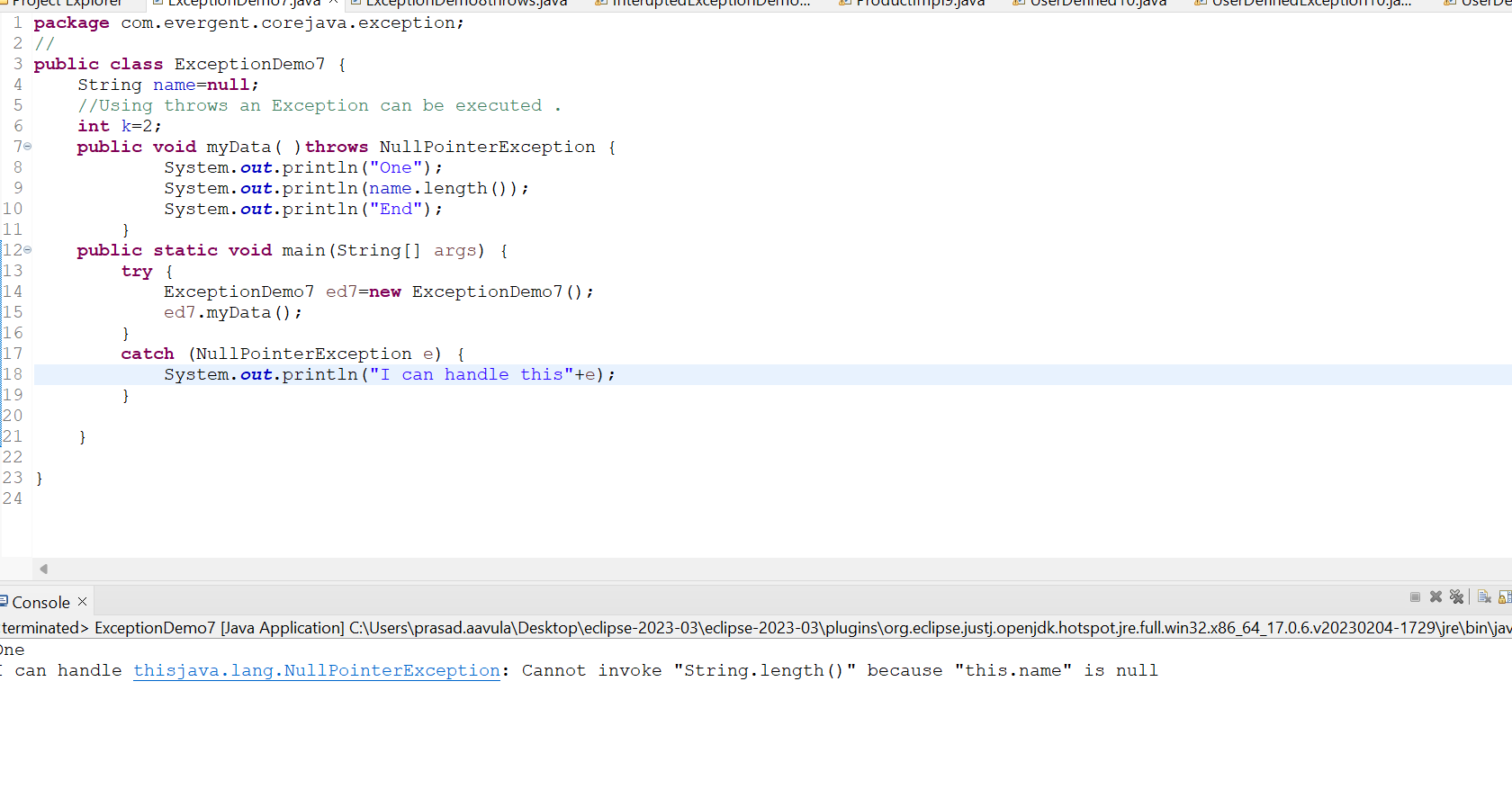
Finally is a block,irrespective of the Exception finally block is executed.



try followed by either catch block or finally block



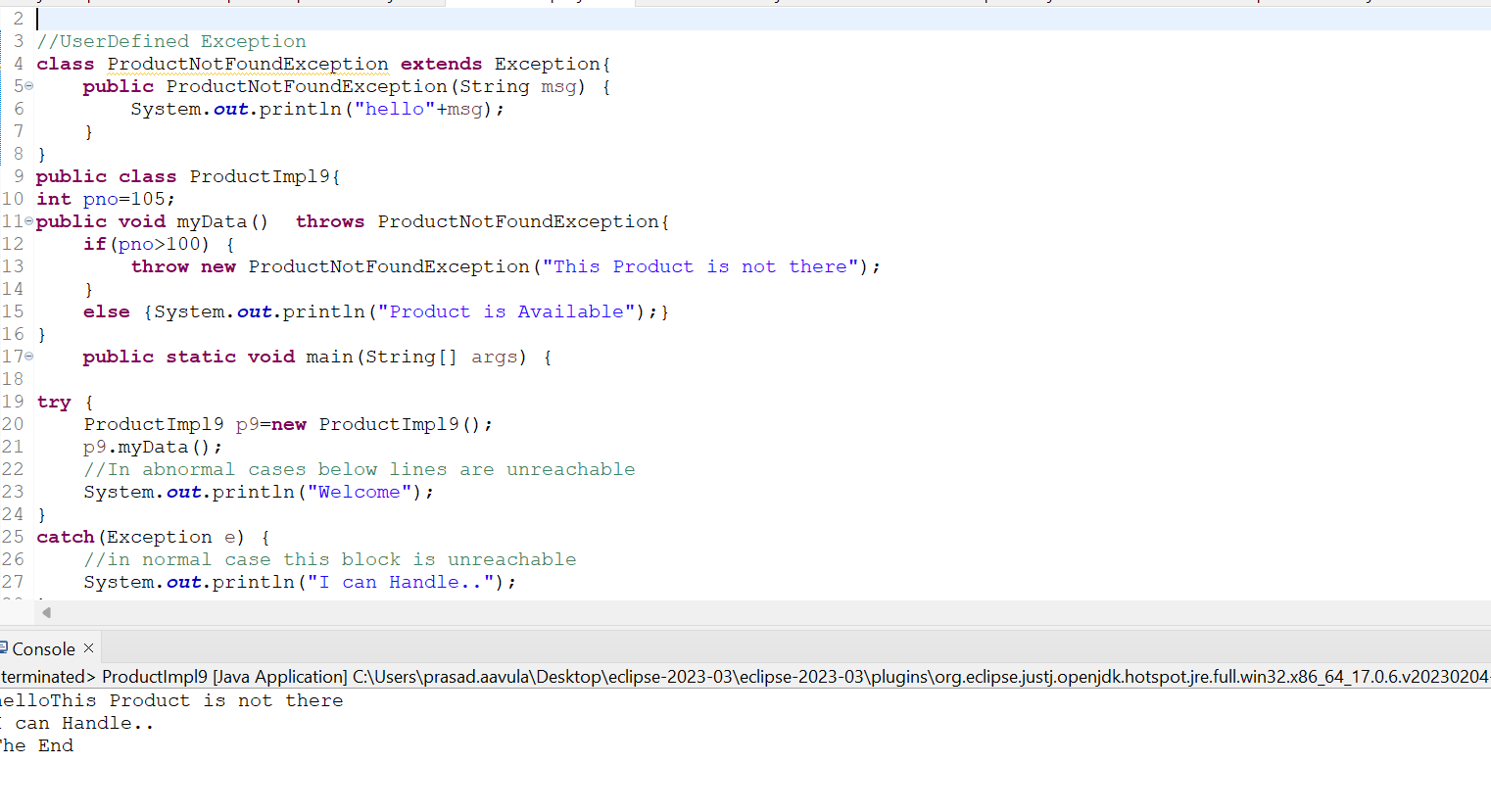
Using throws an Exception can be executed method by method.

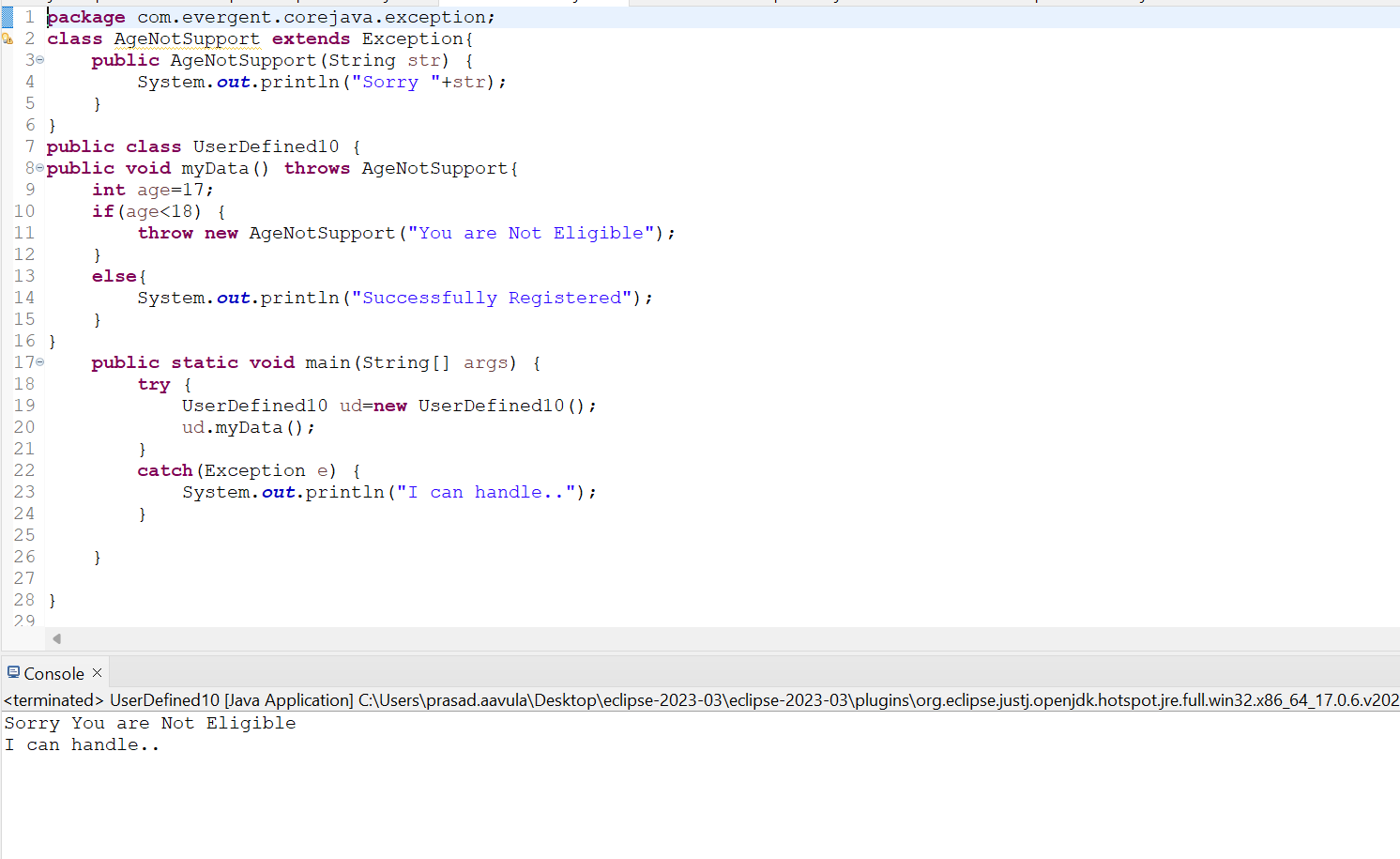




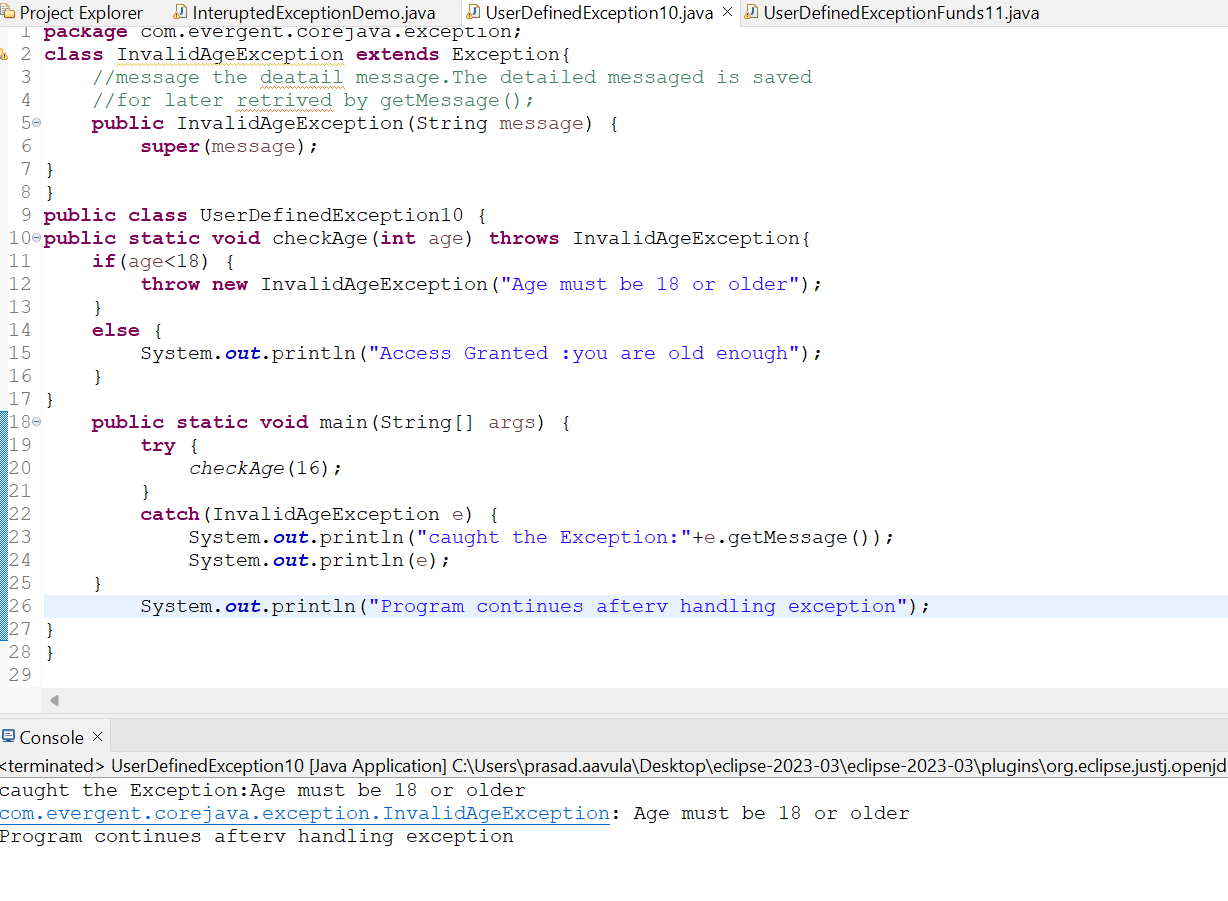
We can create our own(user defined) exceptions.

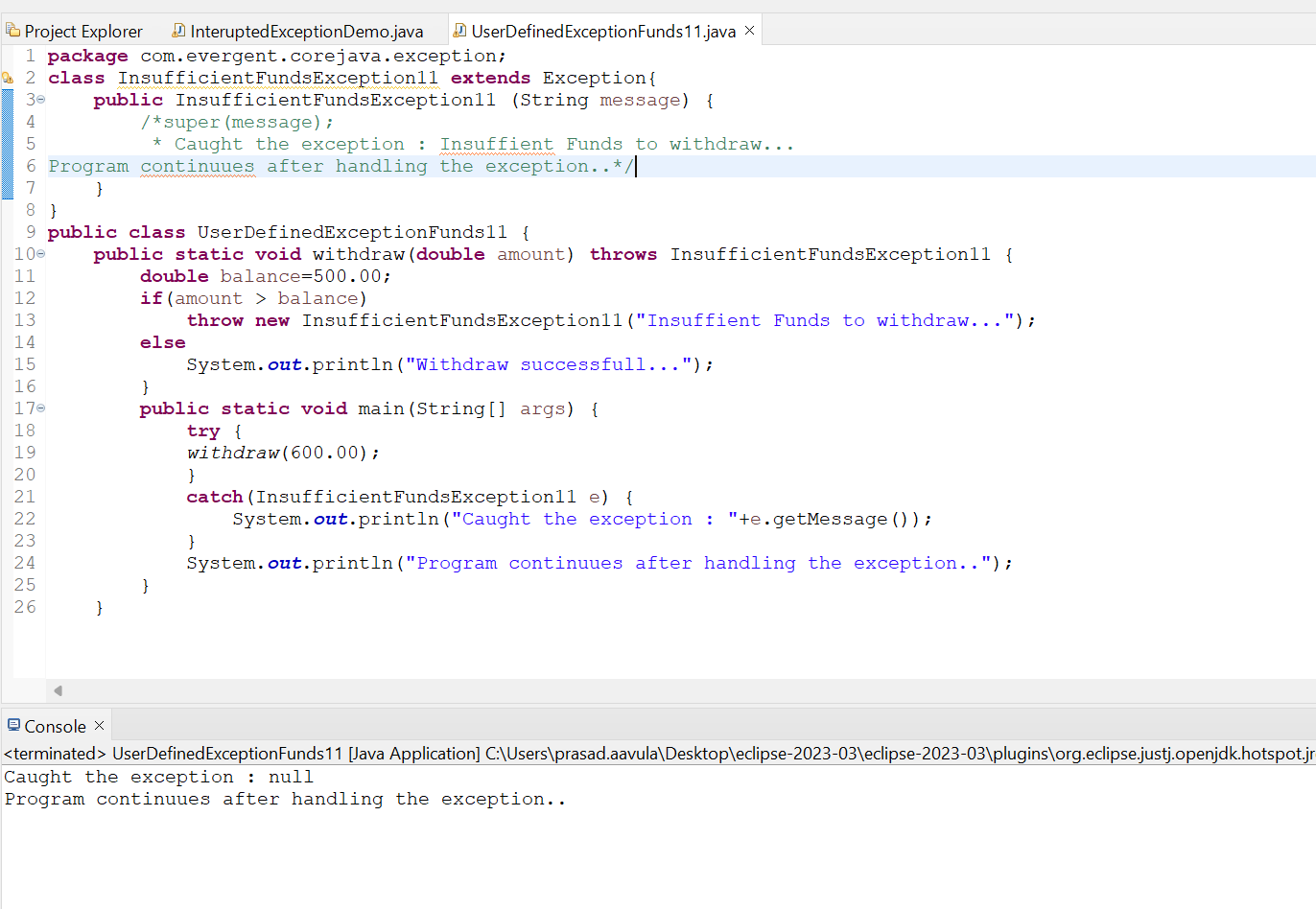
UserDefined Exceptions extends Exception .



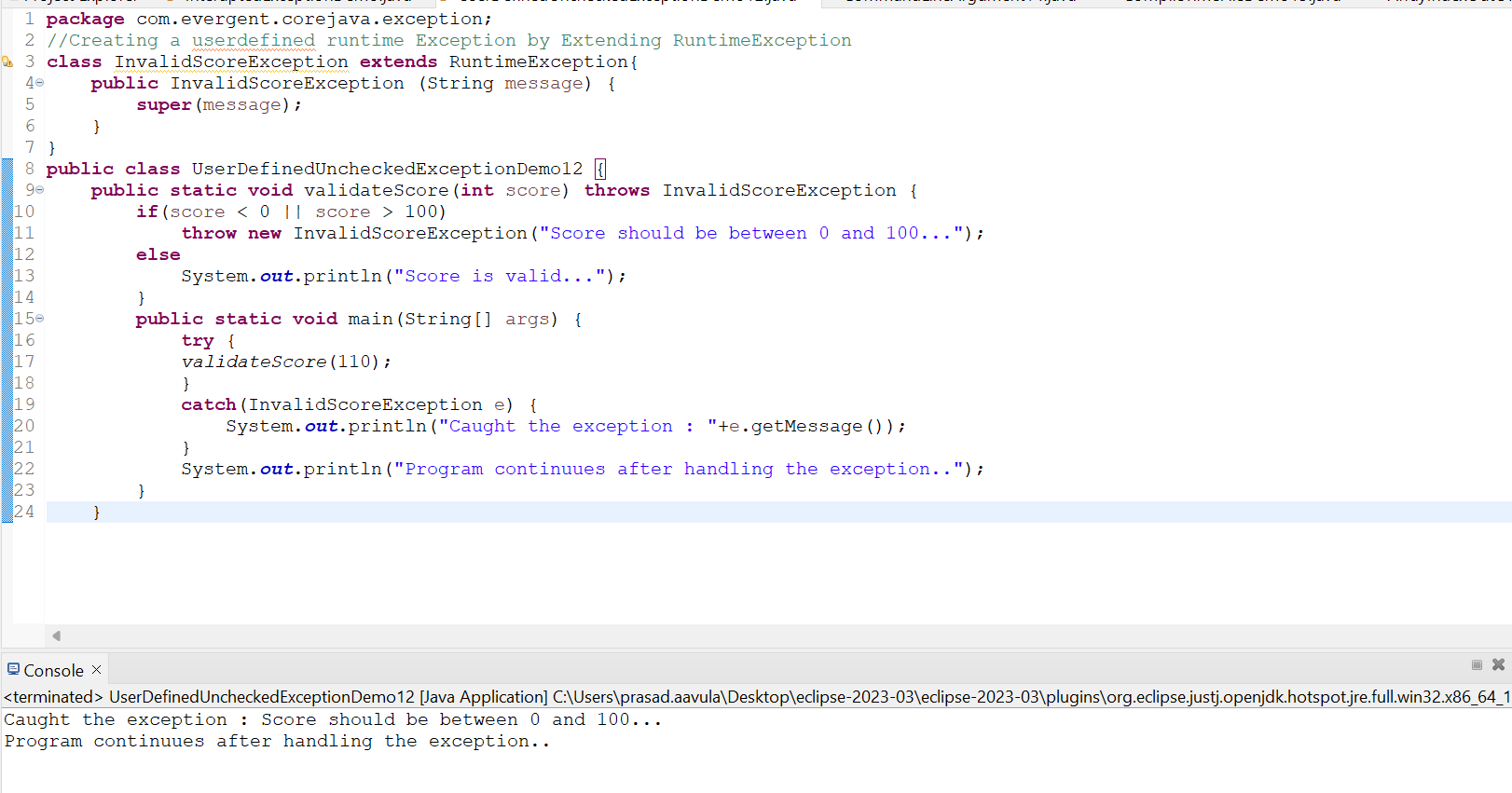


message the detail message.The detailed messaged is saved for later retrived by getMessage()





Creating a userdefined runtime Exception by Extending RuntimeException



ArrayIndexOutOfBound Exception( Predefined RuntimeException)

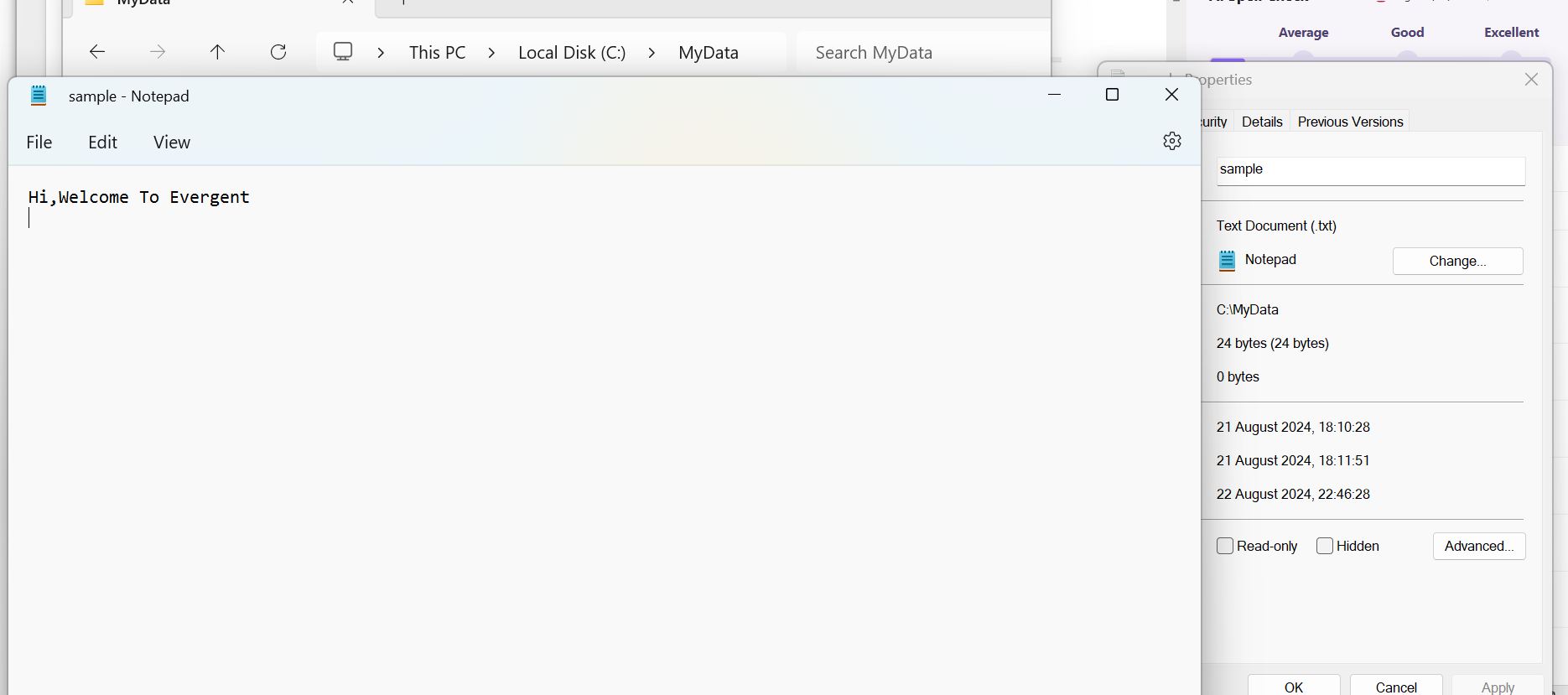


CommanLine Arguments:Arguments passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input.



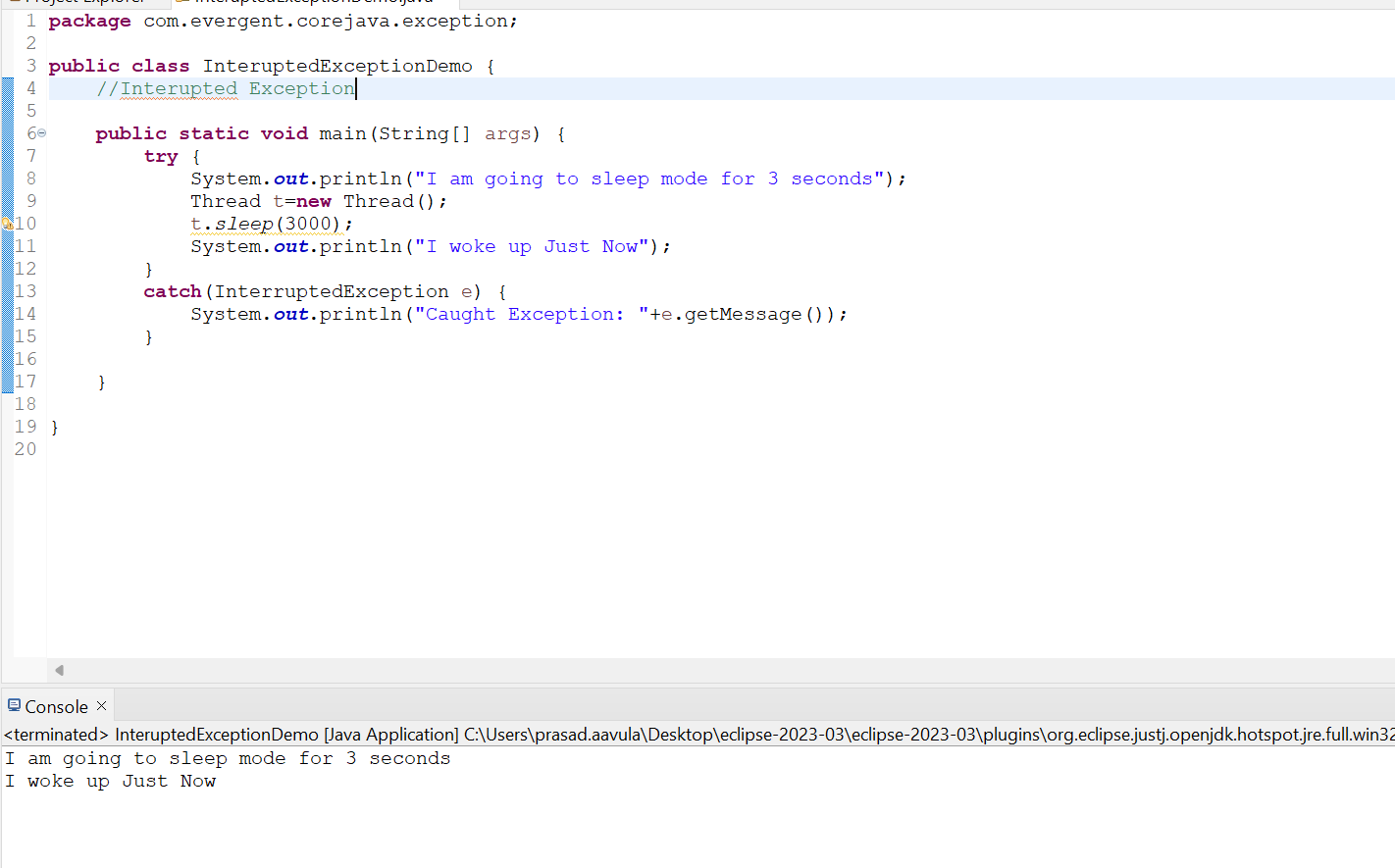
Checked Exception:

FileNotFound Exception

Sample.txt:

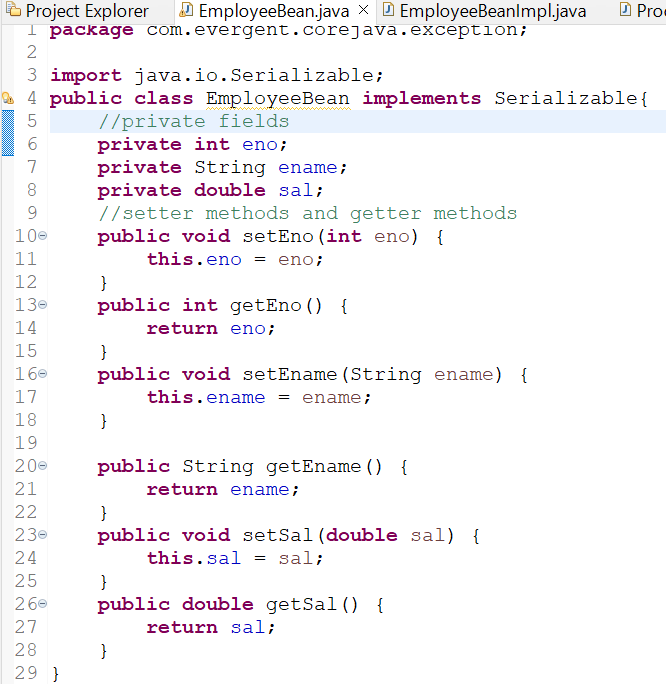


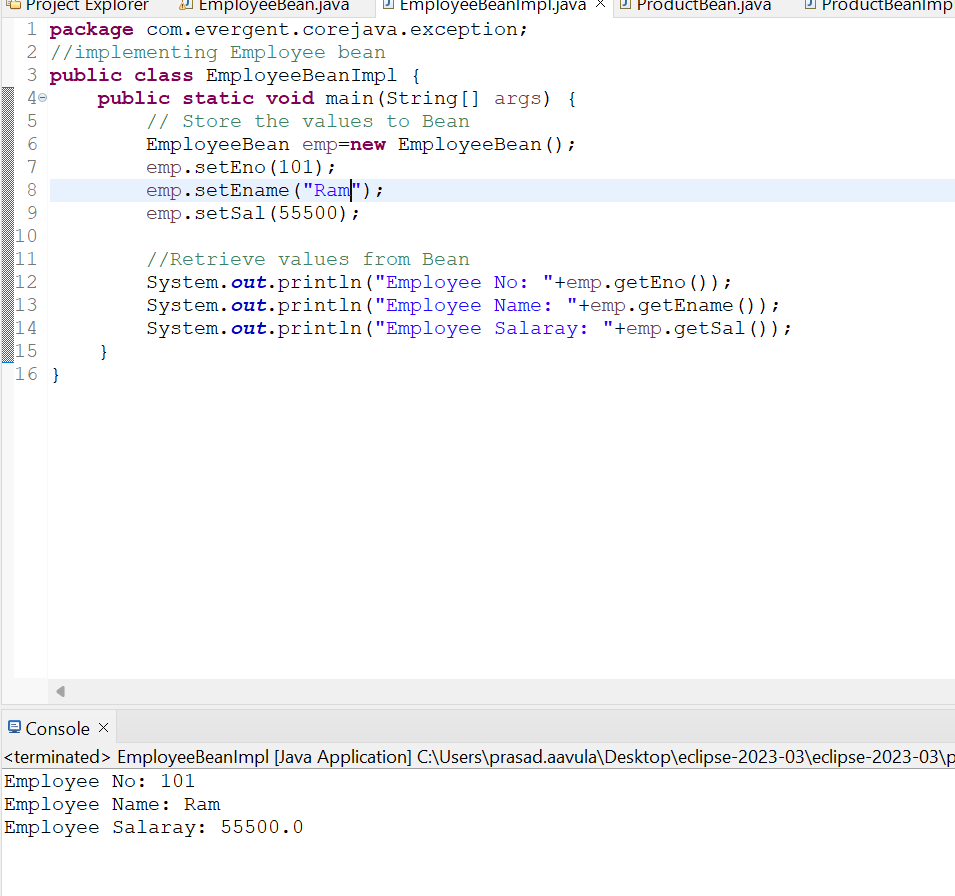
Interupted Exception:



1. **Java Beans:**
2. Java Bean is a mechanism and it is a Light weight.
3. All attributes are private
4. Alll setters and getters methods are public
5. Implements java.io.Serializable interface
6. We can acheive tightly encapsulation using Java beans.

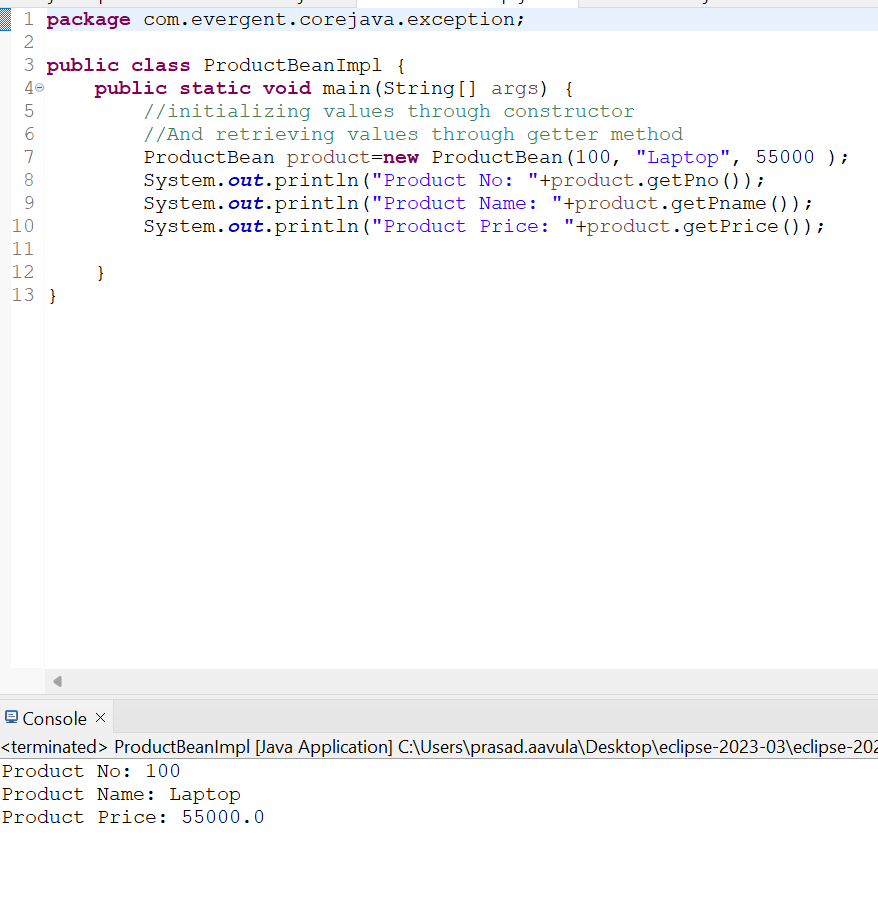
Example1:here the bean create using the setters and getter methods





Example 2:Here initializing the values using constructor and reteiving through getters methods





Example3:Here Retriving the valus using toString() method

