**ETPL708**

**EVERGENT TECHNOLOGIES PRIVATE LIMITED**

**CORE JAVA TRAINING**

**INDEX**

**05-08-2024(Day-1)**

1. Languages and Applications
2. **Java Features**

**a.**Why Java is platform independent

**b.**OOPS

**c.**Exceptional handling

**d.**Multi Threading

**e**.Security

**f.**Open source

**g.**Networking

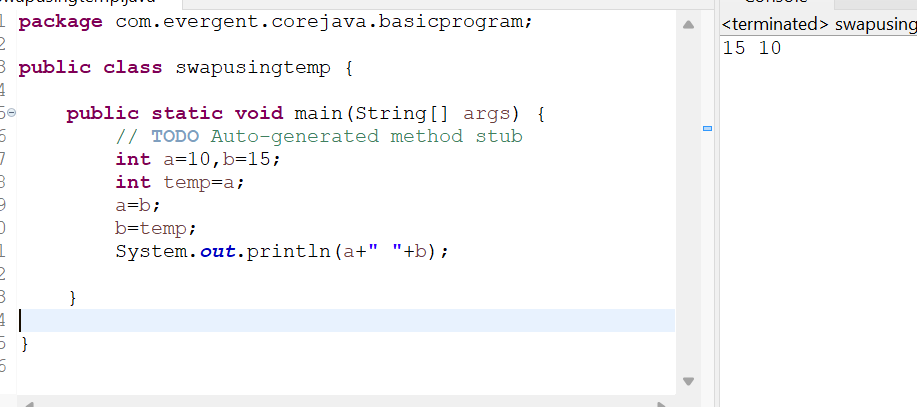
**h.**Memory management

**i.**Web application

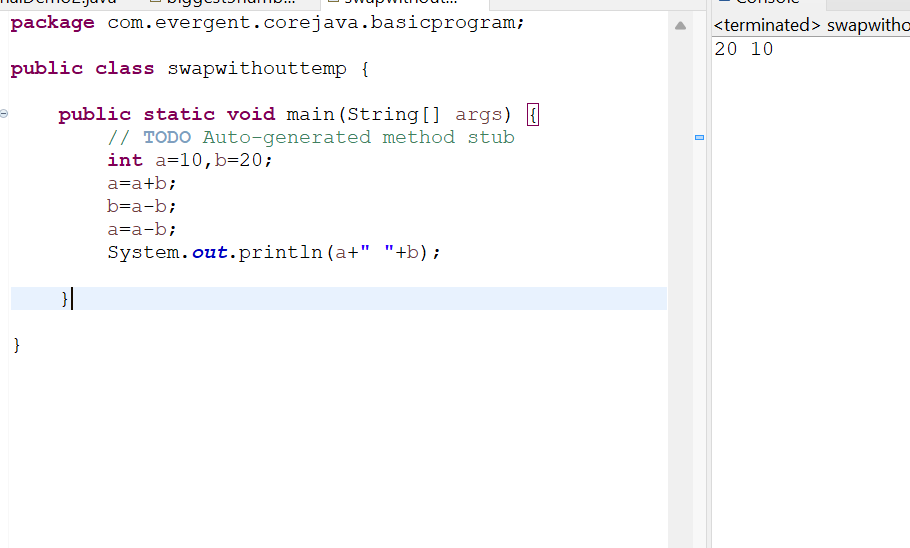
1. **JDK,JRE,JVM**
2. **Basic Java Programming**
3. **Packages**

**PROGRAMS:**

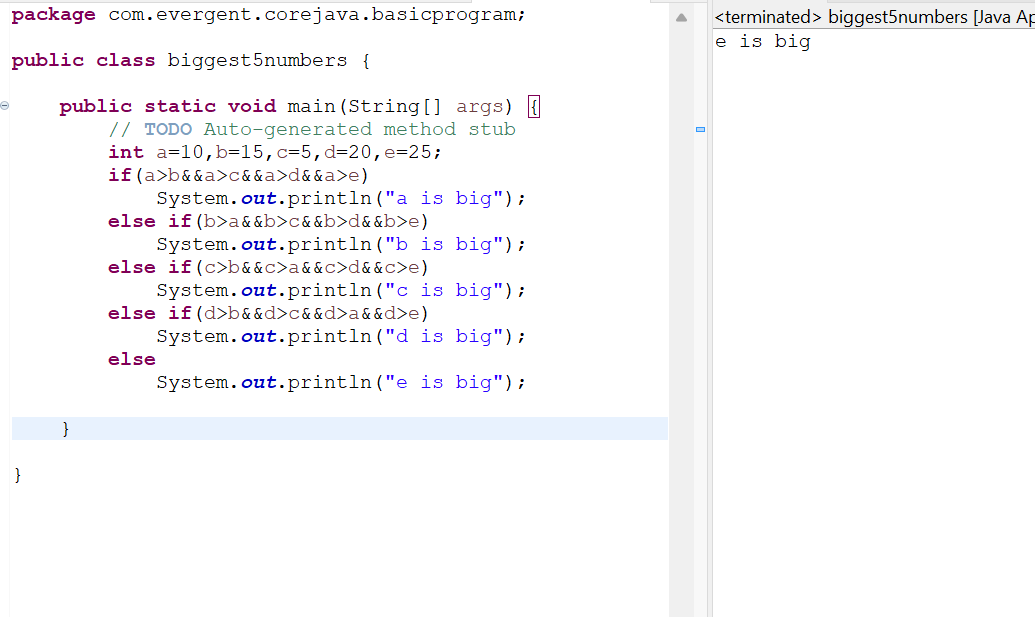
**Program-1:**Swapping using temp



**Program-2:**Swapping without temp



**Program-3:**Finding big num from 5 numbers



**06-08-2024(Day-2)**

**1.**Nested loops

**2.**Arrays(1-Dimensional)

**3.**Arrays(2-Dimensional)

**4.**Logical Programming

**5.**Switch Case

**6.**java.lang package

**a.**Object class methods

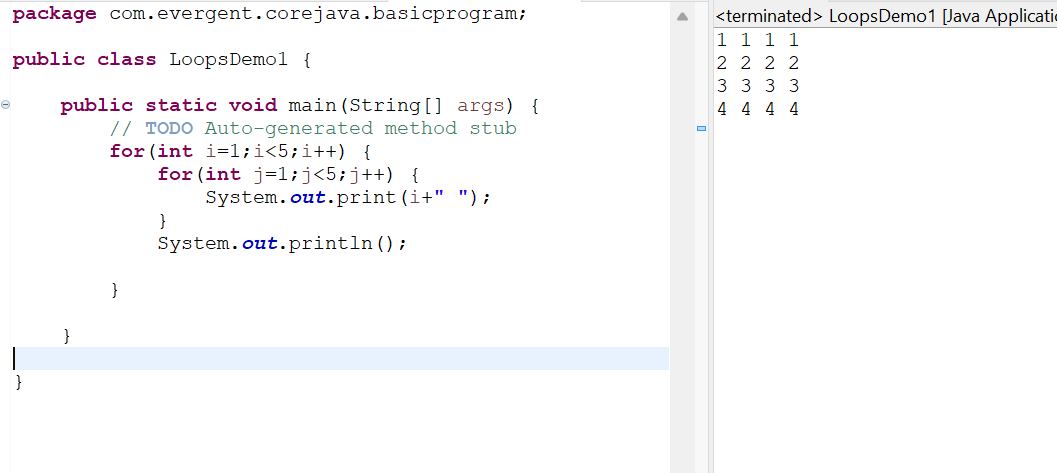
**7.**Scanner class

**8.**enum

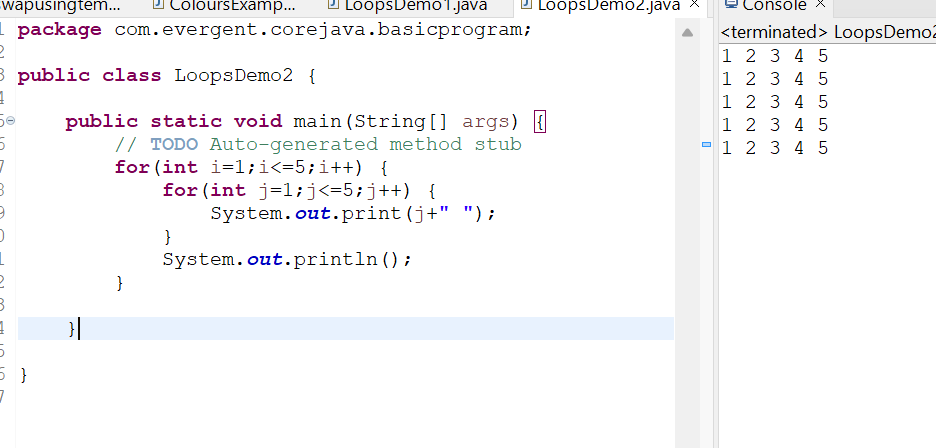
**9.**Event Management Applications

**PROGRAMS:**

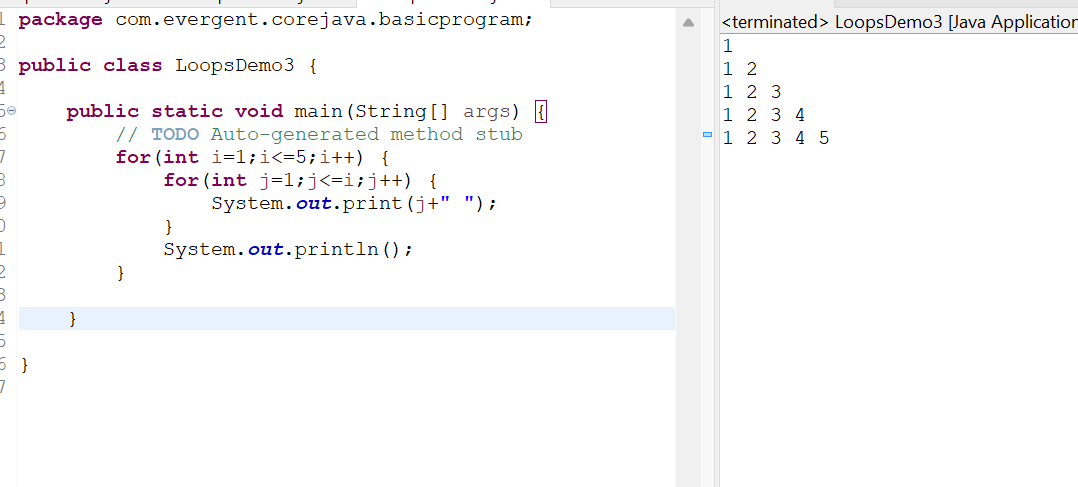
**Program-1:**



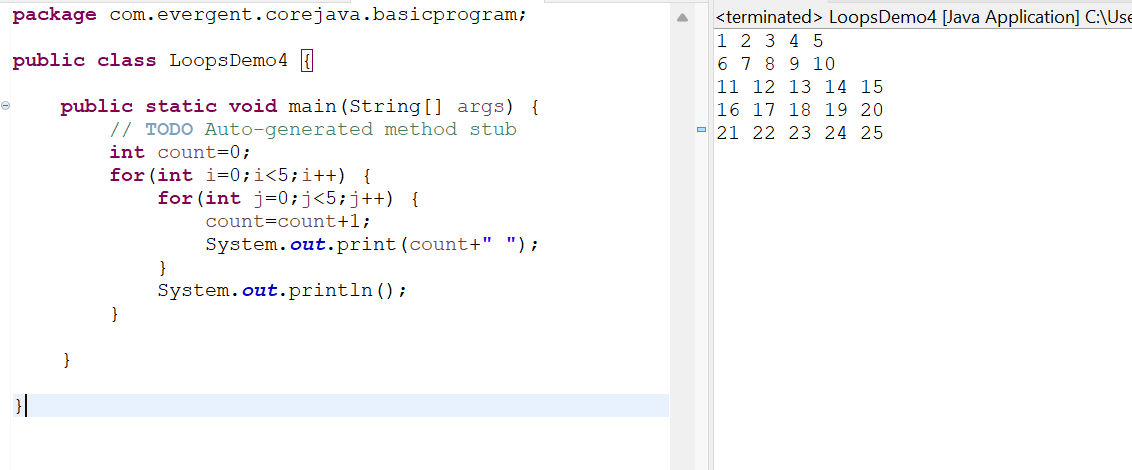
**Program-2:**

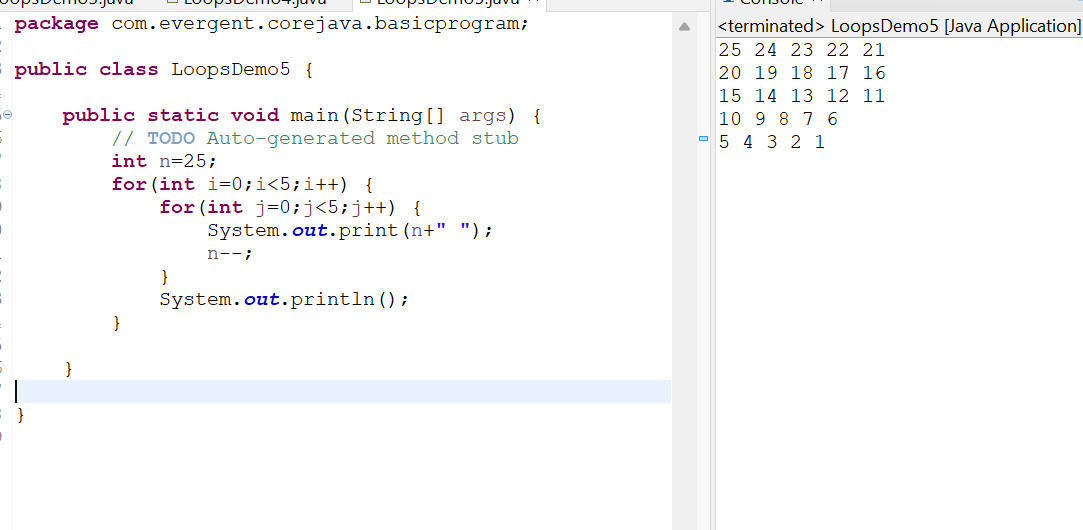


**Program-3:**



**Program-4:**

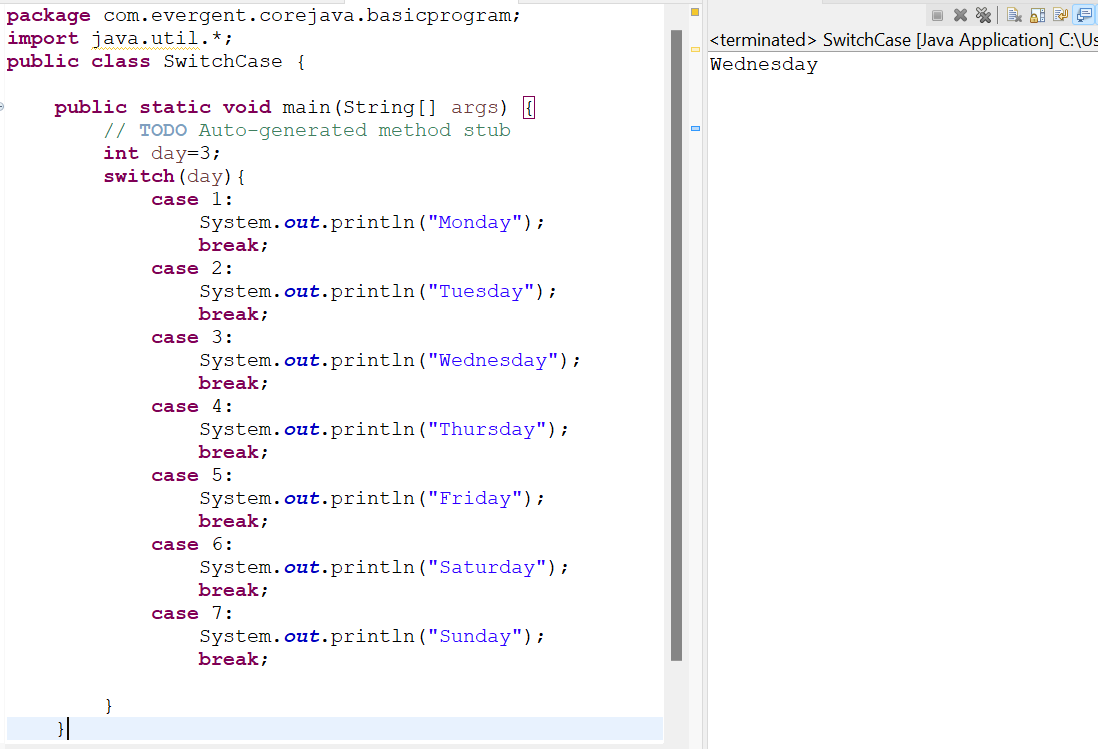


**Program- 5:**

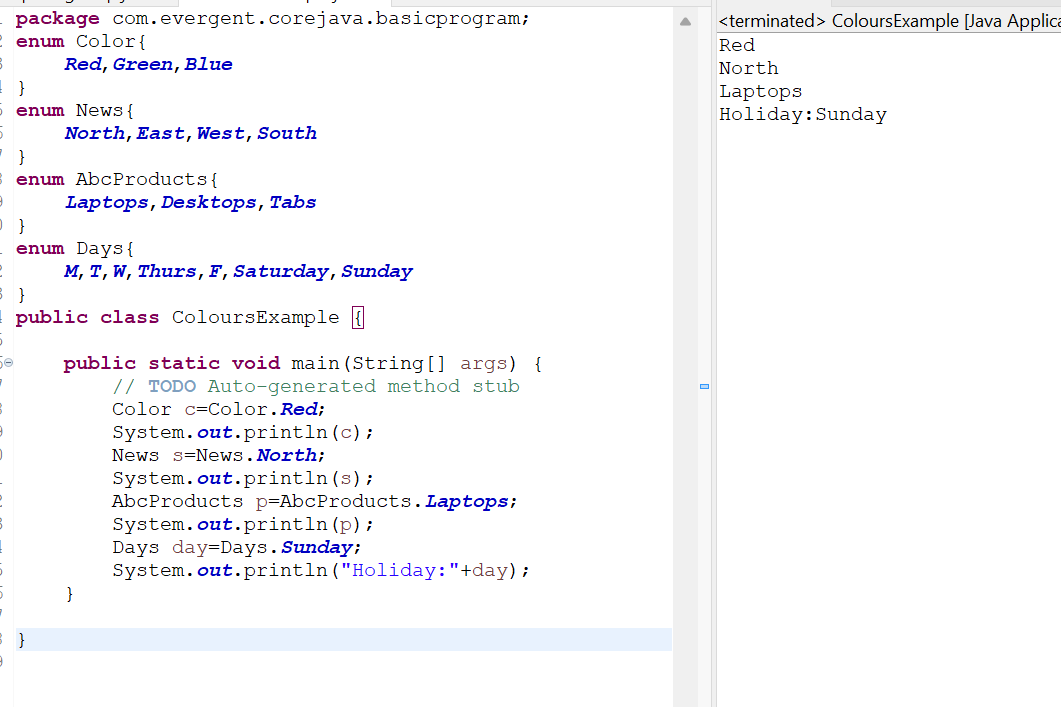
**Program-6:**



**Program-7: Using SwitchCase**



**Program-8: Using enum**



**07-08-2024 (Day-3)**

**OOPS Concepts**

**1.**Encapsulation

**2.**Method Flows

a.No arguments No return type

b.Arguments with no return type

c.Arguments with return type

d.No arguments with return type

**3.**Inheritance

a.Multilevel Inheritance

**4.**Polymorphism

a.Overloading

b.Overriding

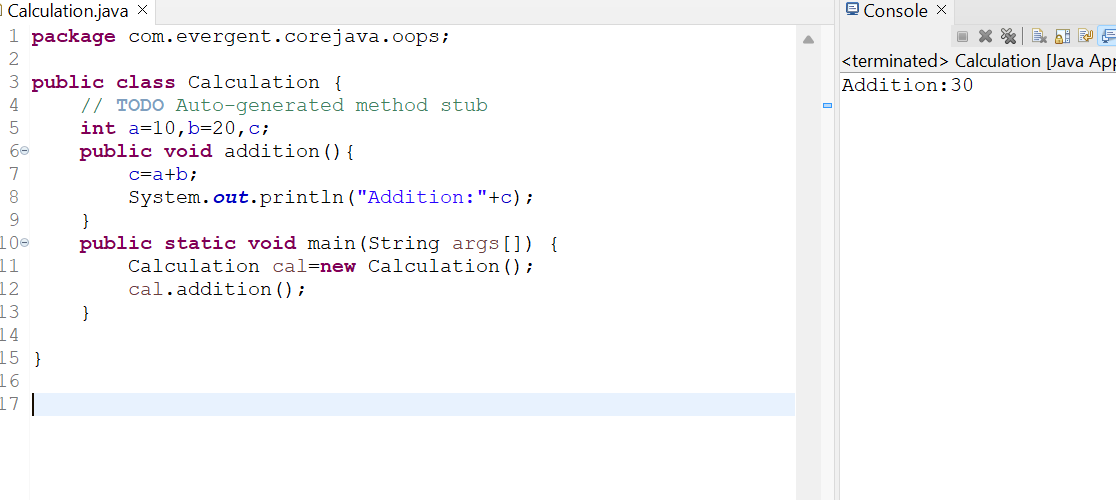
**5.**Abstraction

a.IS-A relationship

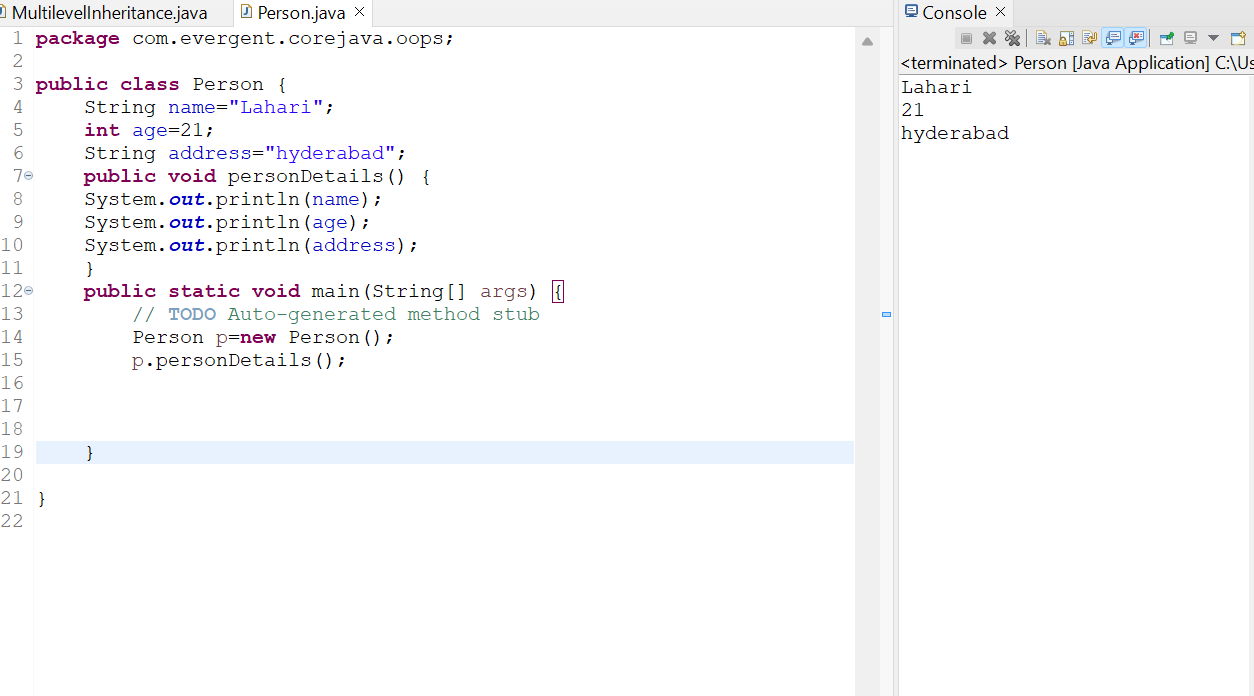
b.HAS-A relationship

**PROGRAMS:**

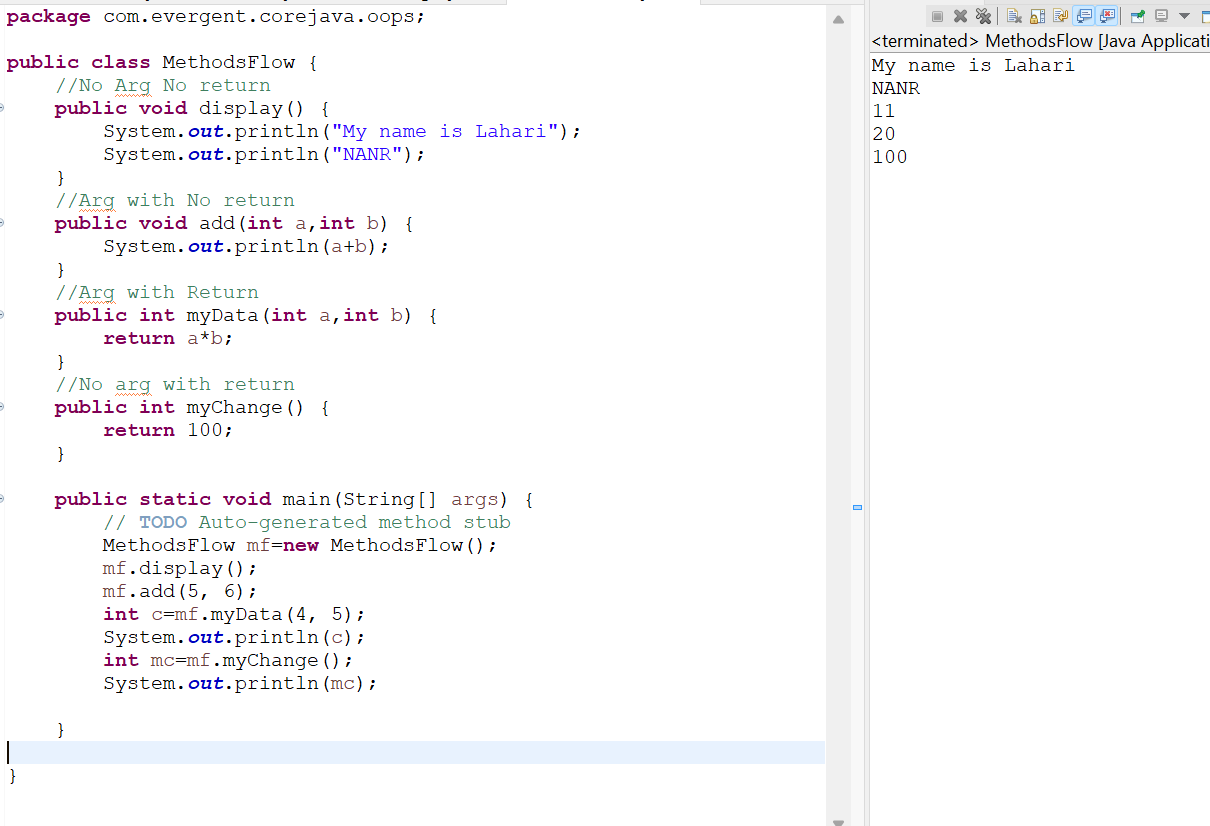
**Program-1:Encapsulation**



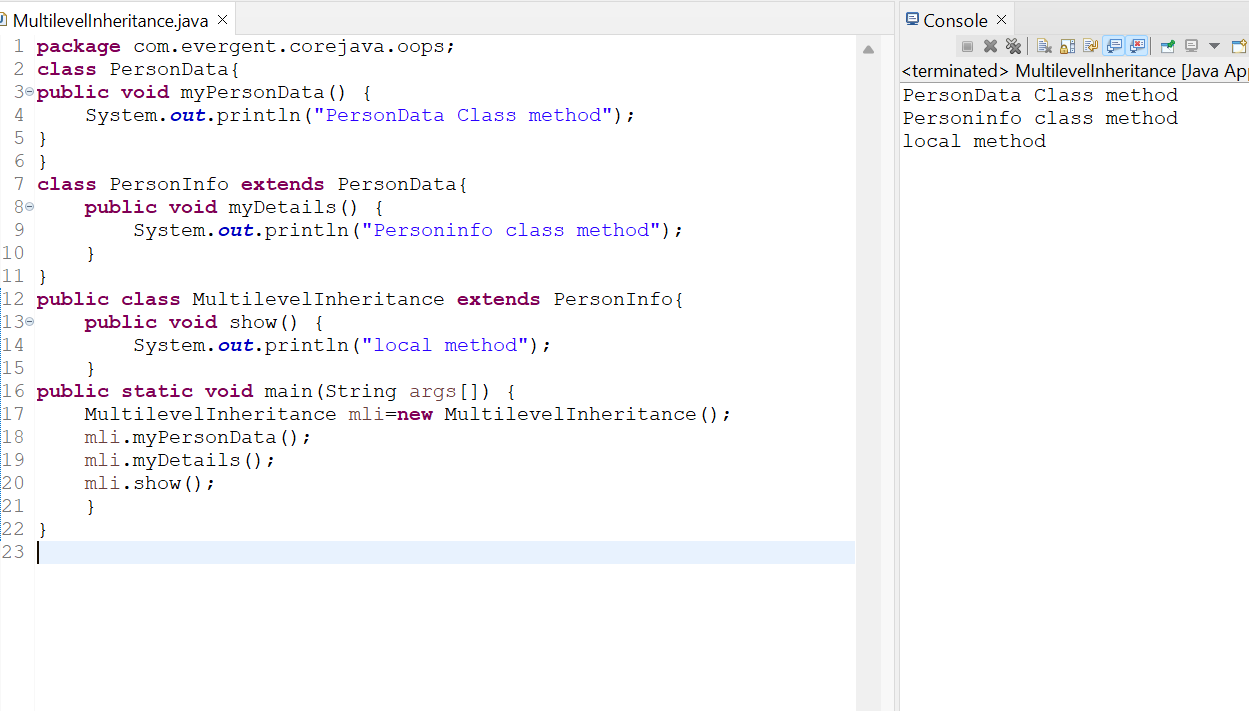
**Program-2:Encapsulation**

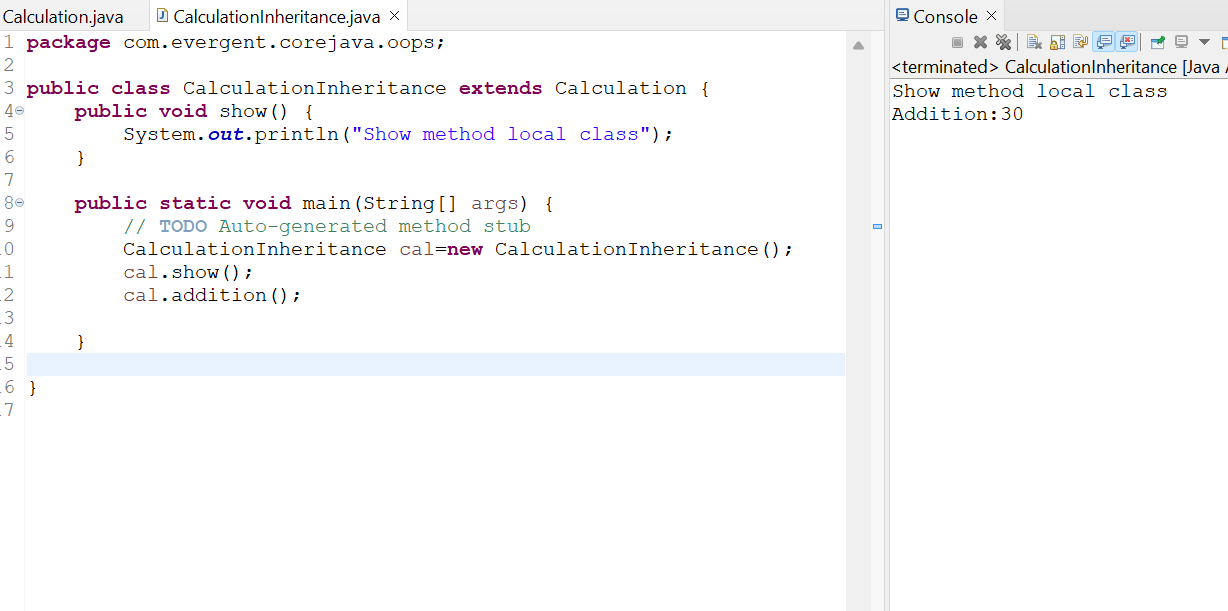


**Program-3:MethodsFlow**

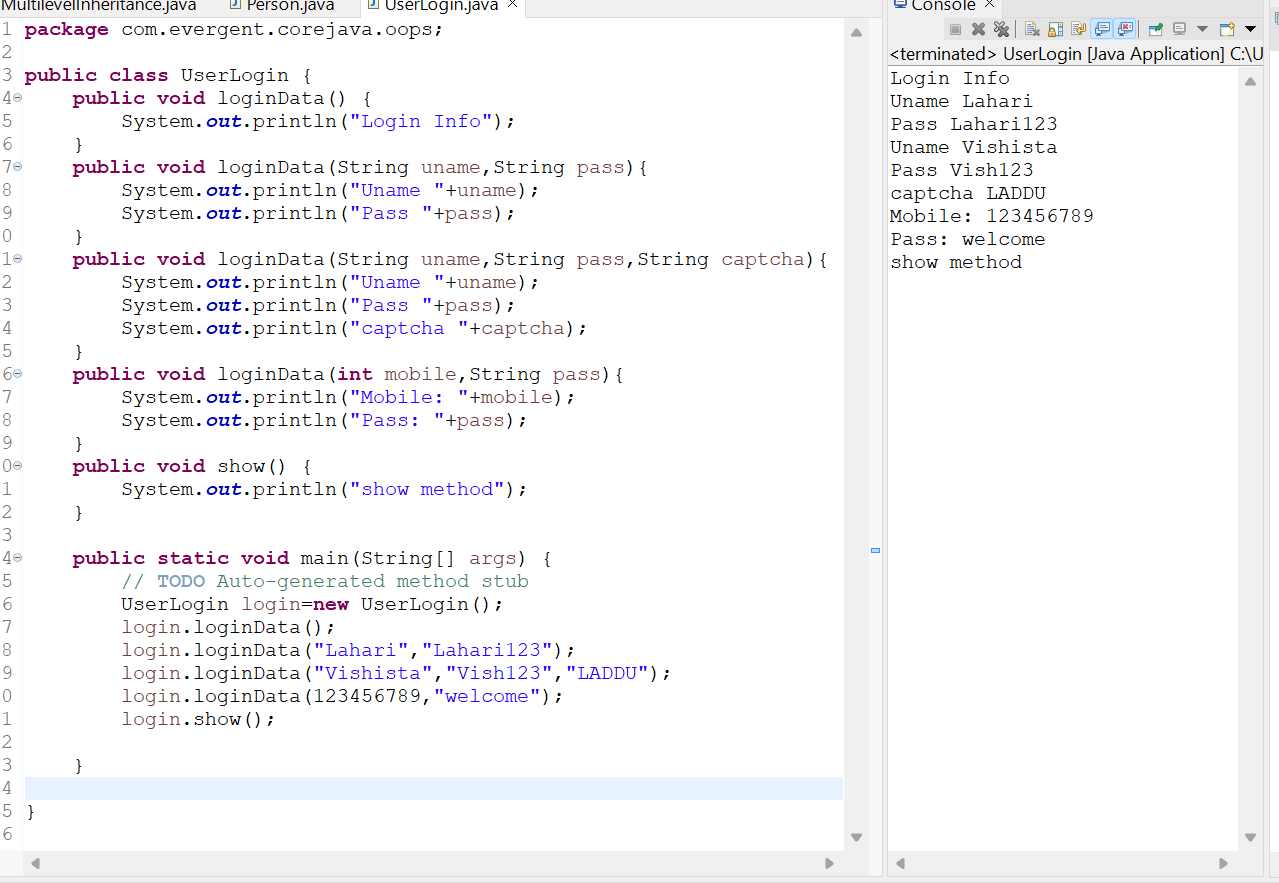


**Program-4:MultiLevelInheritance**

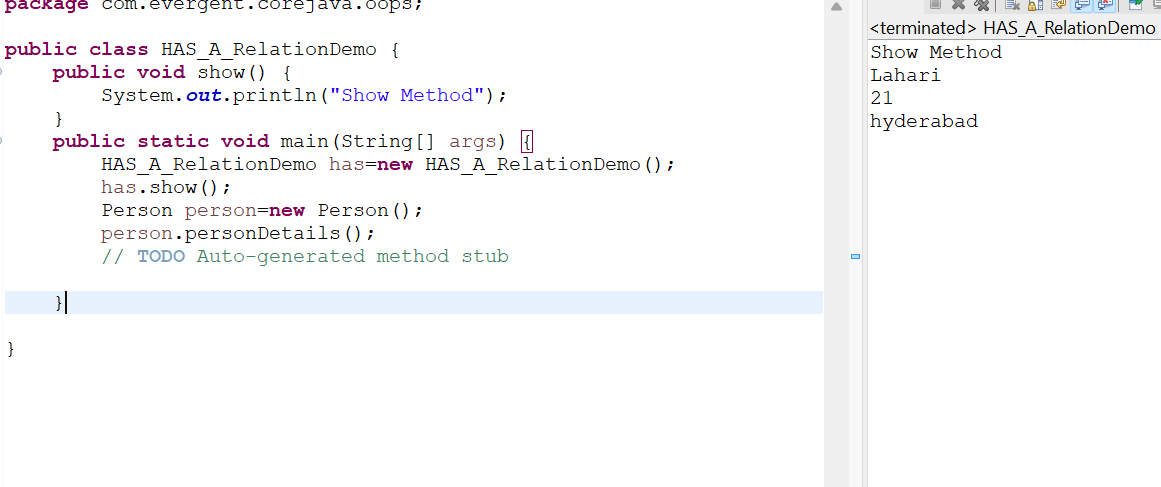


**Program-5: CalculationInheritance**

**Program-6:Method Overloading**



**Program-7: Has A Relationship**



**08-08-2024 (DAY-4)**

**CONSTRUCTORS**

1. Class name and Constructor name should be same.
2. There are 2 types of constructor.

* Default constructor
* Parameterised constructor

1. We can access constructor while creation of objects.
2. Constructors are mainly for initialization.
3. Constructor doesn’t have any return type,not even void also.If we declare as a void,the compiler will consider as method not as constructor.
4. EVery class needs atleast one default constructor.
5. Always constructors are overloaded.
6. This is a keyword always pointing to instance variables.
7. Super is a keyword always points to the super class variables and constructors.
8. We also have copy constructor,that will copy the constructor.

**PROGRAMS:**

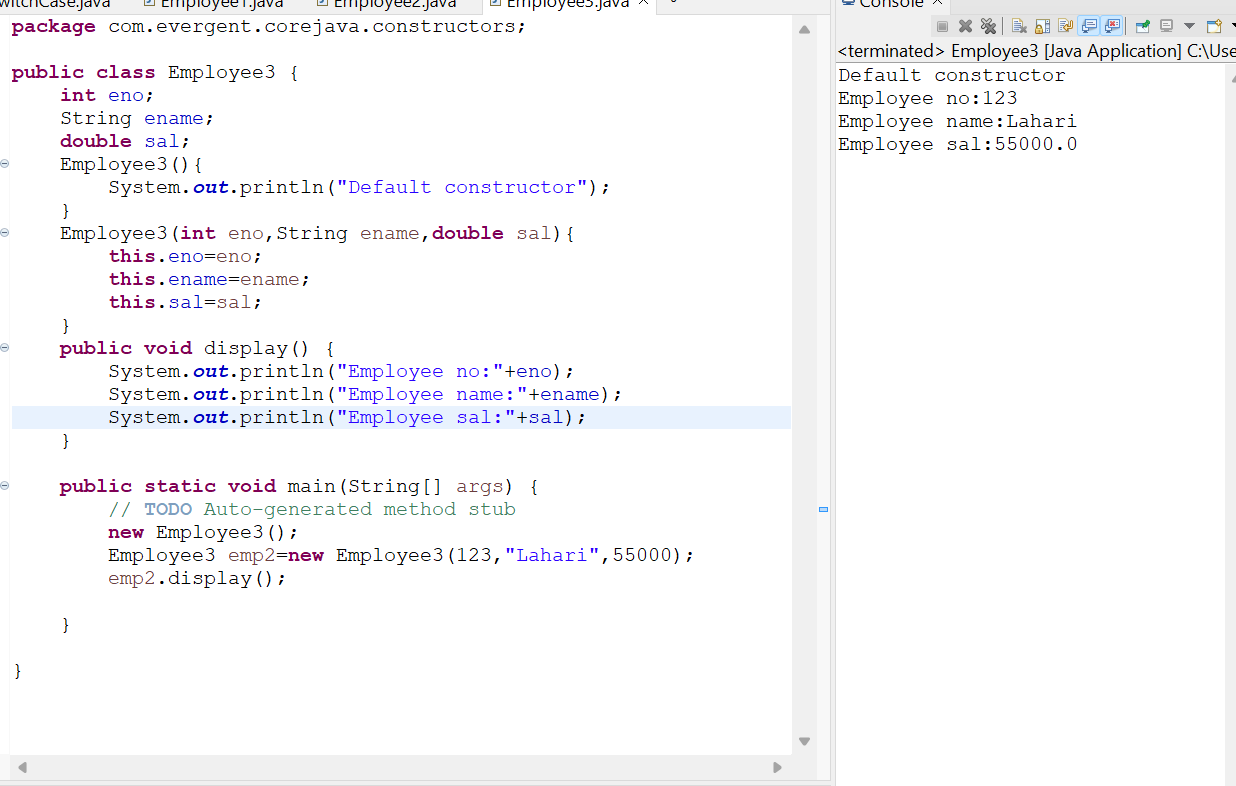
**Program 1: Constructor Example**



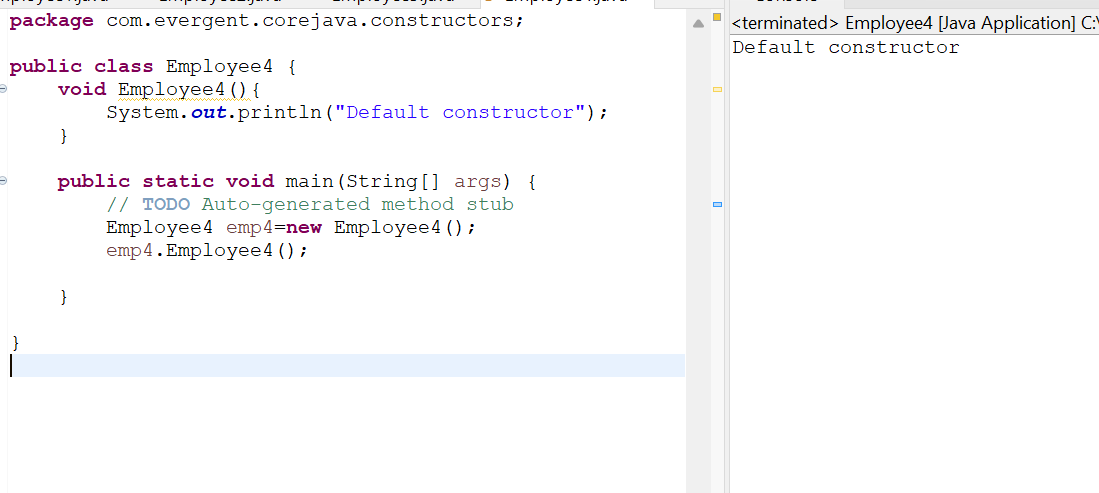
**Program-2: Constructor Overloading**

****

**Program-3:Using this keyword**

****

**Program-4:Constructor with void**

****

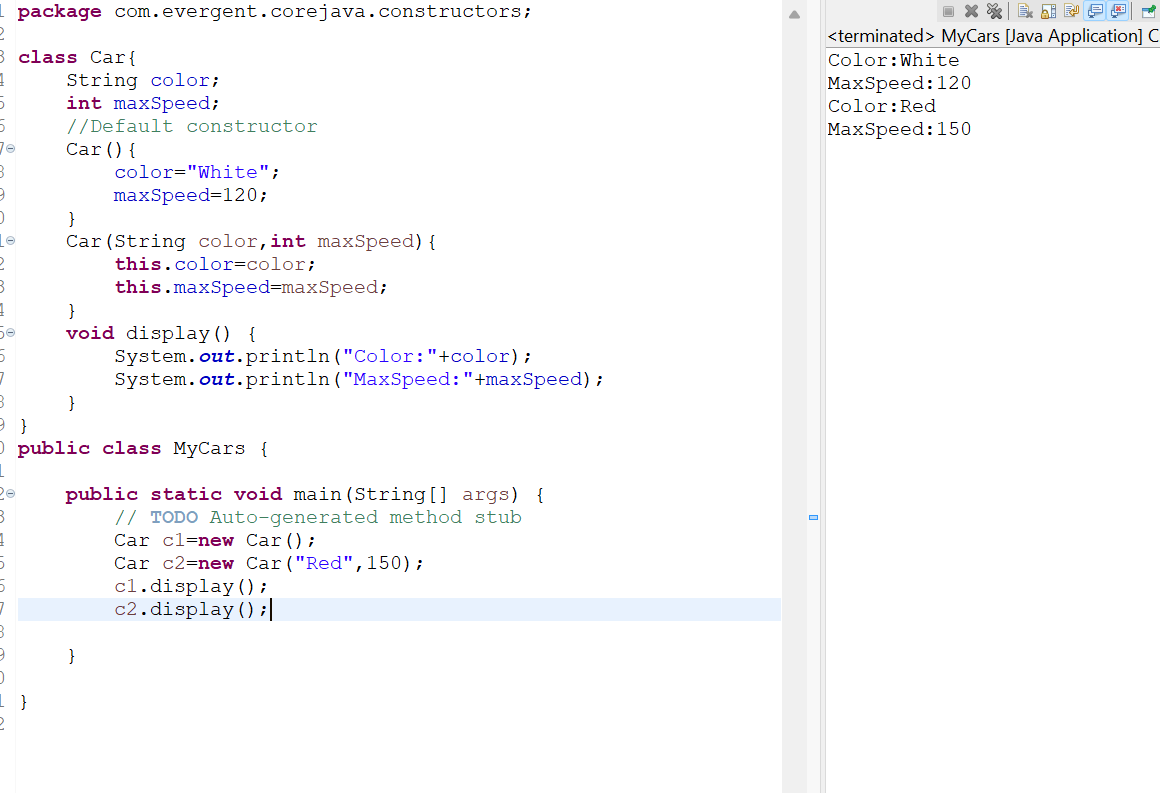
**Program-5: Constructor using this**

****

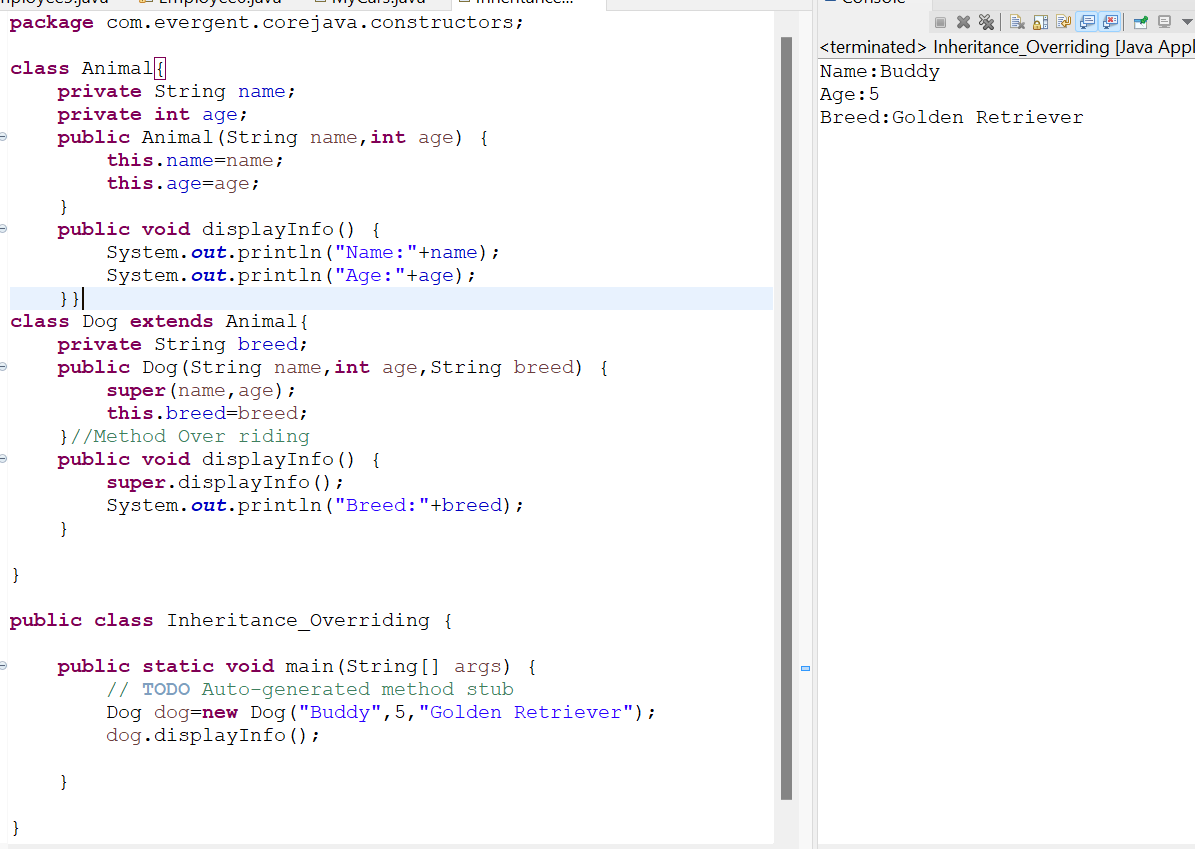
**Program-6: super keyword**

****

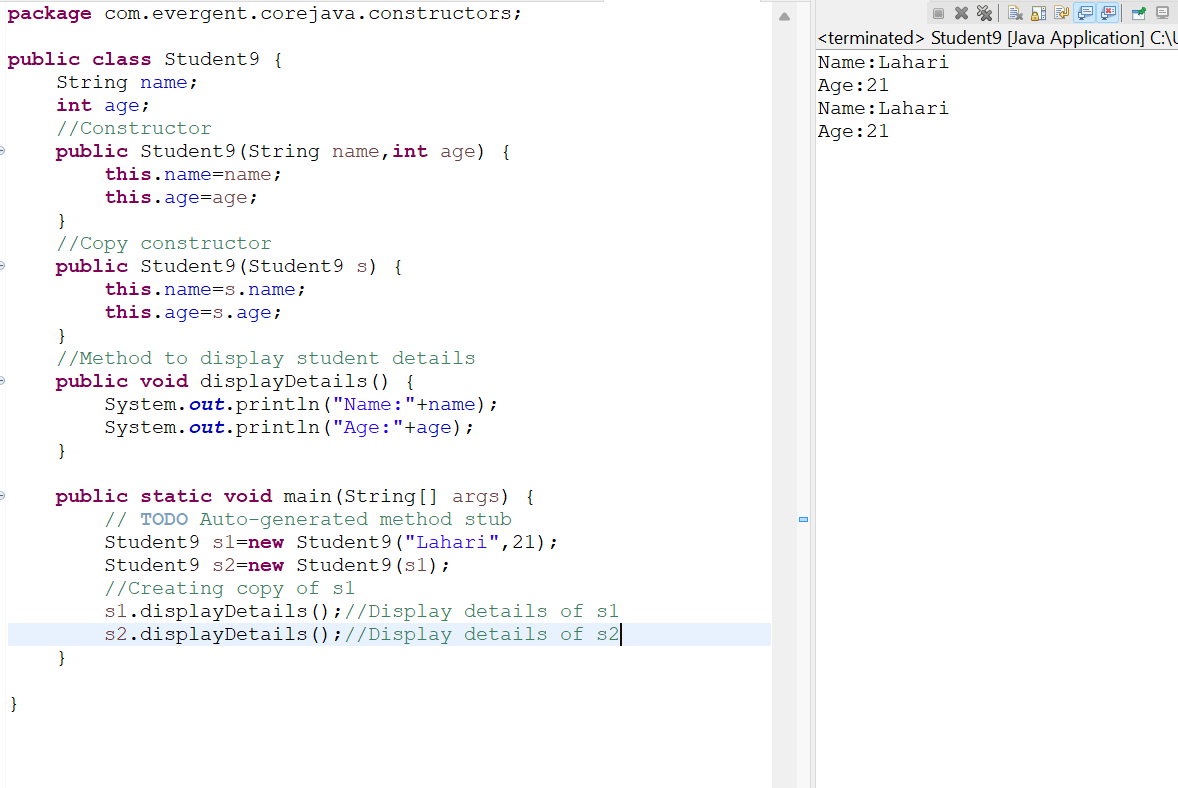
**Program-7: Constructor Overloading**

****

**Program-8: Inheritance Overloading**

****

**Program-9:Copy Constructor**

****

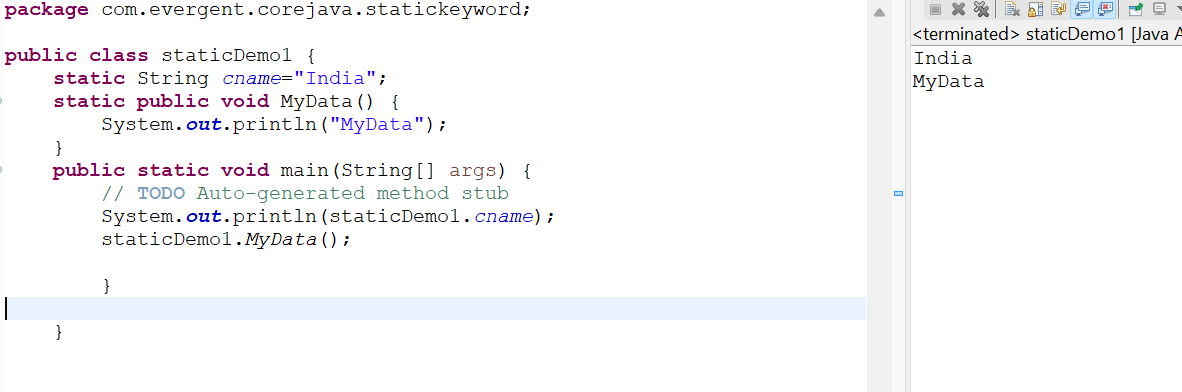
**09-08-2024 (DAY-5)**

**STATIC**

1. Static is a keyword.
2. We can declare static variables and methods direct through classname.methodname and classname.variablename.
3. Static methods can access static variables and static methods only.
4. Static methods cannot access non static methods and non static variables.
5. Non static methods can access static methods and static variables.
6. Whenever class is loaded into jvm,at that time static blocks are initialized first only.

**PROGRAMS:**

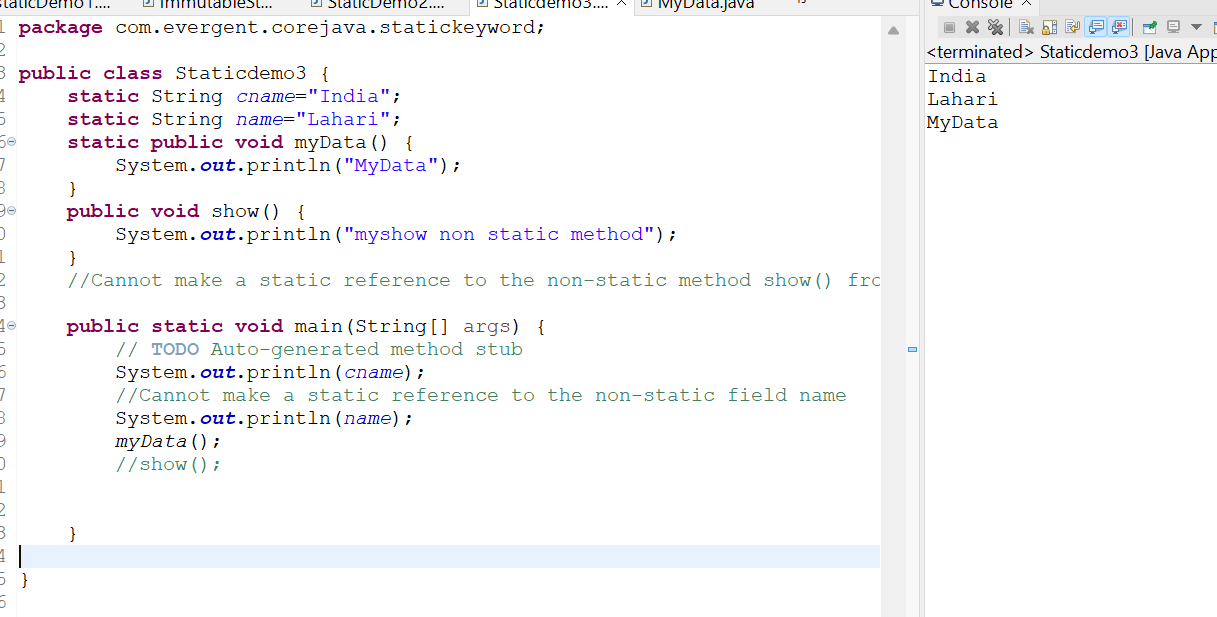
**Program-1: Static as Variable**

****

**Program-2:Static as method**

****

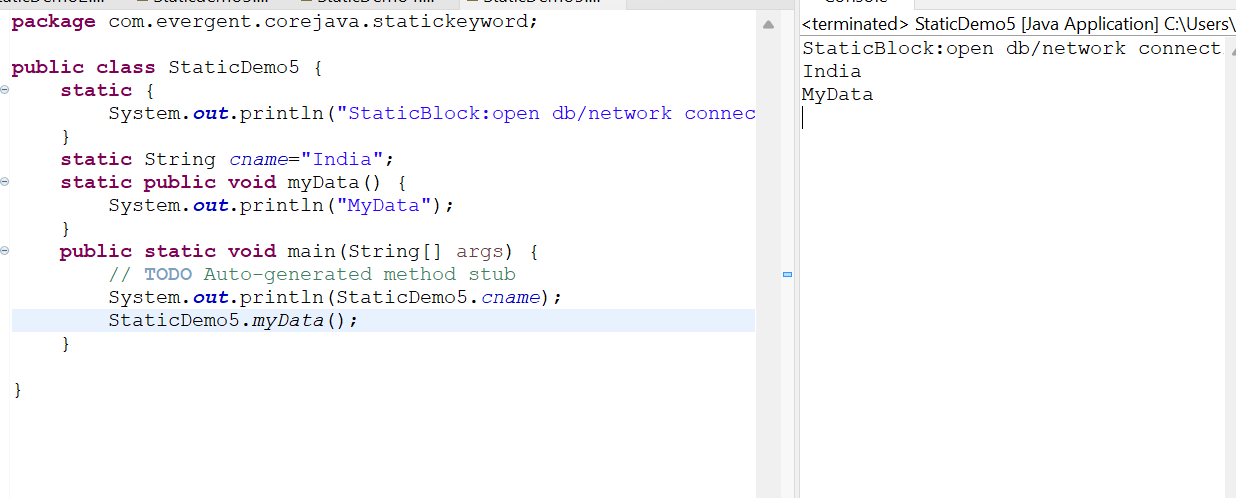
**Program-3: Static can not access Non Static variables**

****

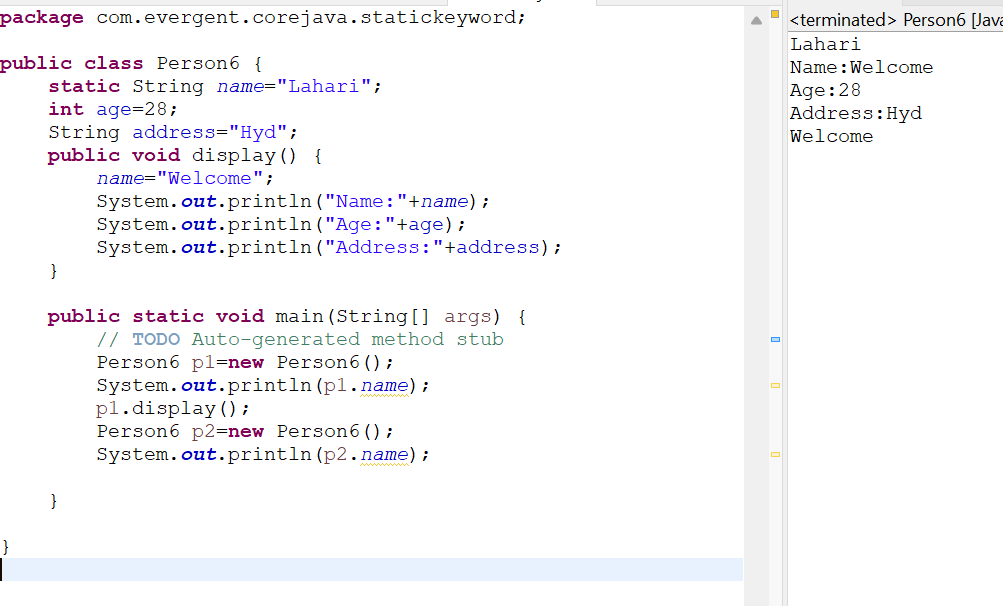
**Program-4: Non static can access Static methods**

****

**Program - 5: Static Block**

****

**Program-6:Change in Static variable in local method**

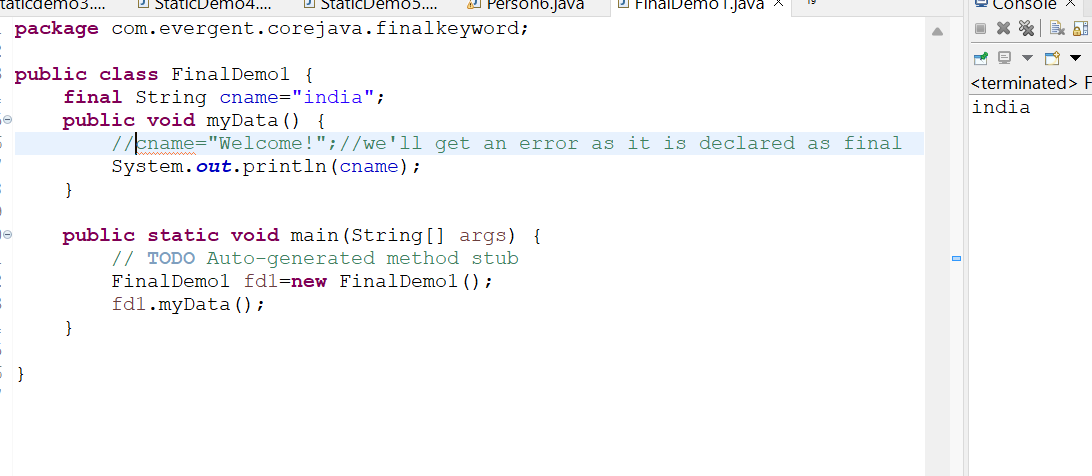
****

**FINAL**

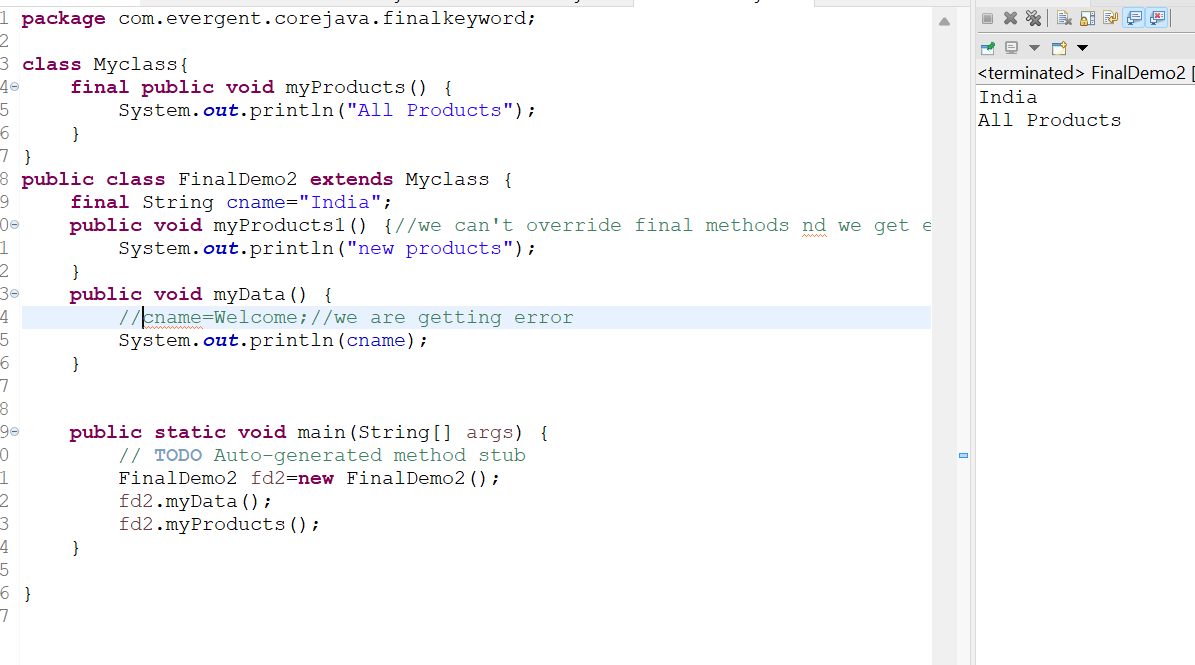
1. Final Keyword
2. We can declare final as variables,methods and class
3. Final variables we can’t modify.
4. Final method we can’t override.
5. Final class we can’t be inherited (we can’t use extends).
6. Final variables can also initialize while constructor calling.

**PROGRAMS:**

**Program-1: Final variable**

****

**Program-2:Final method**

****

**12-08-2024 (DAY-6)**

**STRINGS**

**1.**String class

**2.**String Buffer

**3.**String Builder

**Why Strings are immutable?**

Once we declare any string object it is constant,if we are trying to modify existing string,it will create another memory allocation and existing object is eligible for garbage collection.

**Heap pool:**

String str1=new String(“JAVA”);

String str2=new String(“JAVA”);

**Constant pool:**

String str1=”JAVA”;

String str2=”JAVA”;

== → Check Memory Location

Equals → Check content

String methods:

1.length()

2.toLowerCase();

3.toUpperCase();

4.trim();

**String Buffer**

1.String Buffer is final class

2.It is Mutable

3.String Buffer having methods.

4.All String class methods are synchronized.

**String Builder**

1.String Builder is final class.

2.It is Mutable.

3.String Builder having methods.

4.All String class methods are non-synchronized.

* String buffer is not recommended to use for development.

**String Buffer methods:**

1.append()

2.insert()

3.replace()

4.delete()

5.reverse()

6.capacity()

7.length()

**Strings Imp points**

1. In Java,string is a sequence of characters,often used to represent text.
2. Strings are objects in Java and are instances of the string class,which is part of the java.lang package
3. Key features of Strings in Java:

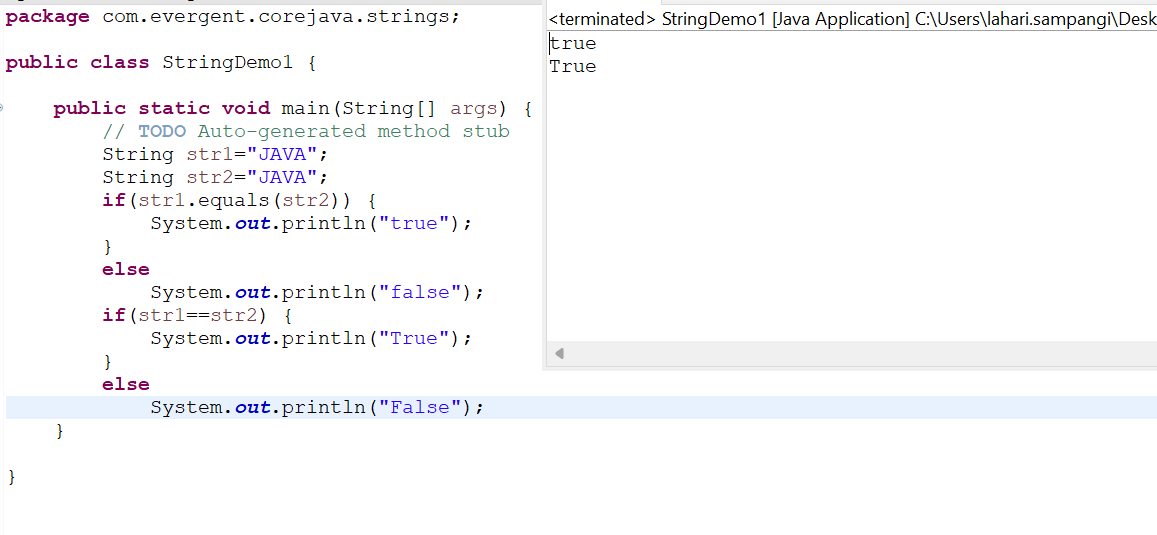
* Immutable: Once a string object is created,it cannot be changed.
* Any modification to a string creates a new string object.

1. Java optimizes memory usage by storing strings in a special area of memory known as “String pool”.
2. If two strings have the same value and are created without using the new keyword,they will reference the same object in the string pool.
3. We can create a string in Java in multiple ways:

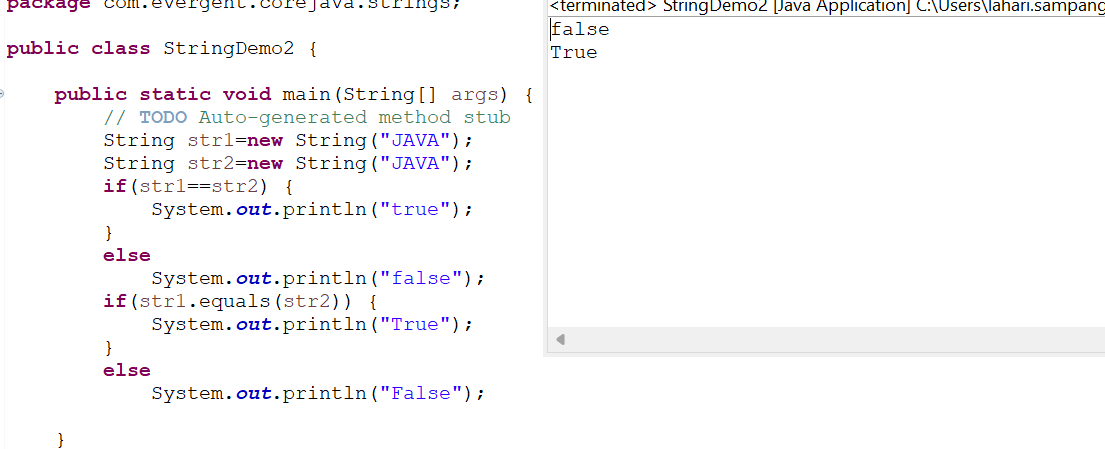
* Using the string literals:
  + - String str=(”Hello,World!”);
* Using the new keyword:
  + - String str=new String(“Hello,World!”);

**PROGRAMS:**

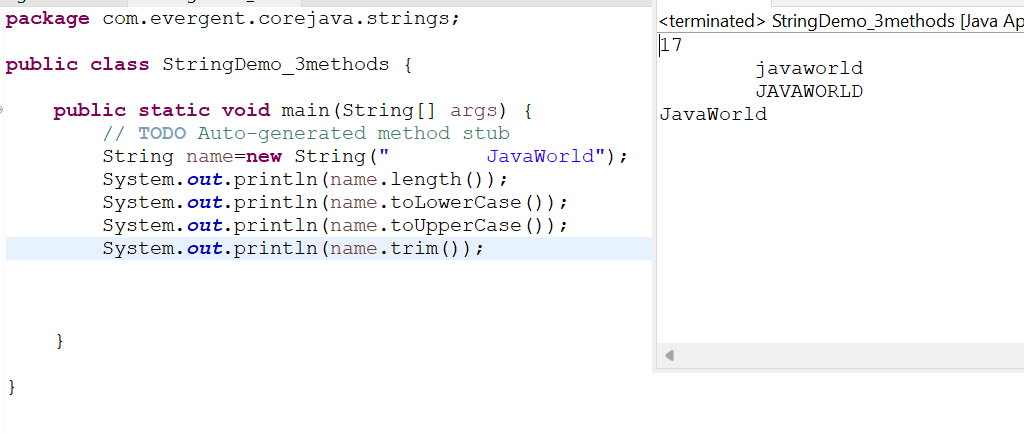
**Program 1:Checking equality in Constant pool**



**Program 2: Checking Equality in Heap Pool**

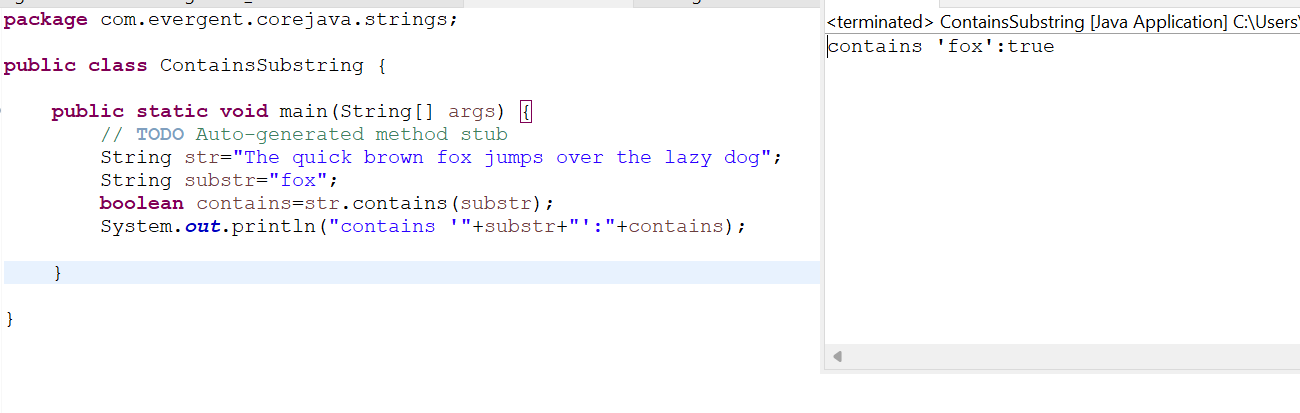


**Program 3: String methods**

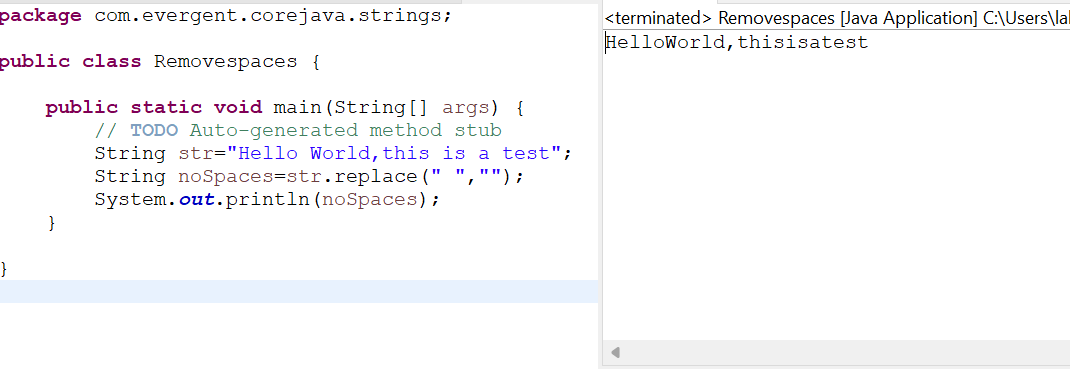


**Program 4:**

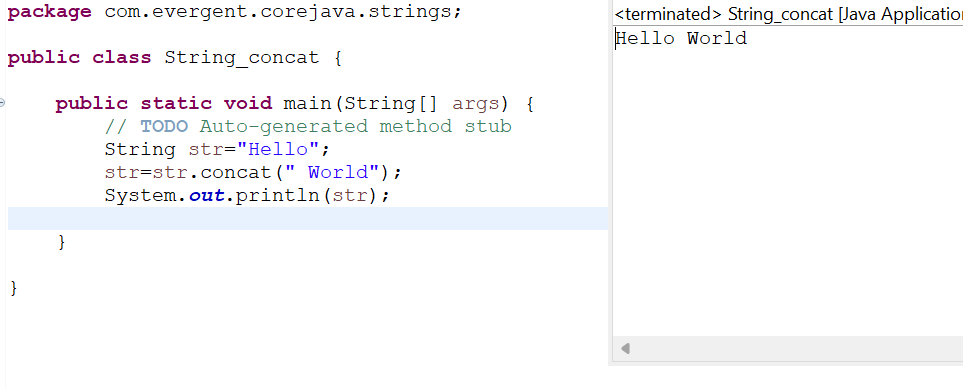
**Create a Java program that creates a String,checks if it contains a specific substring and then prints out the result.**

**Program 5:**

**Write a Java program to create a string,remove all spaces from the string, and then print out the result.**

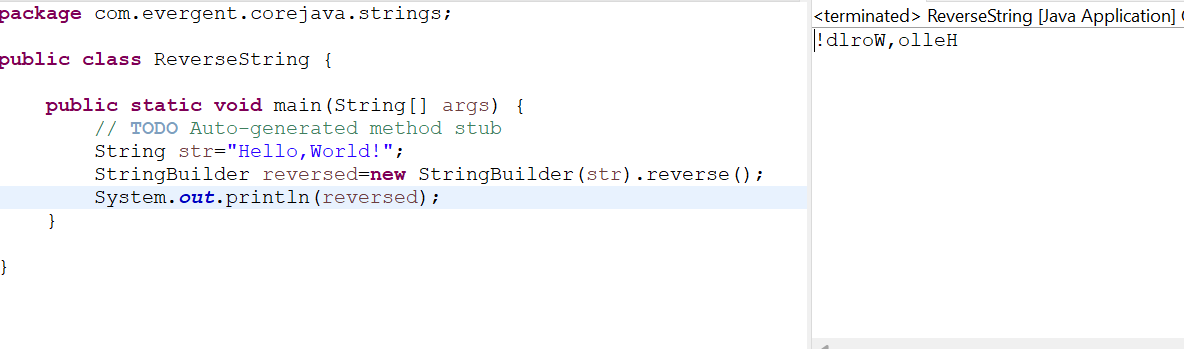


**Program 6: String Concatenation**

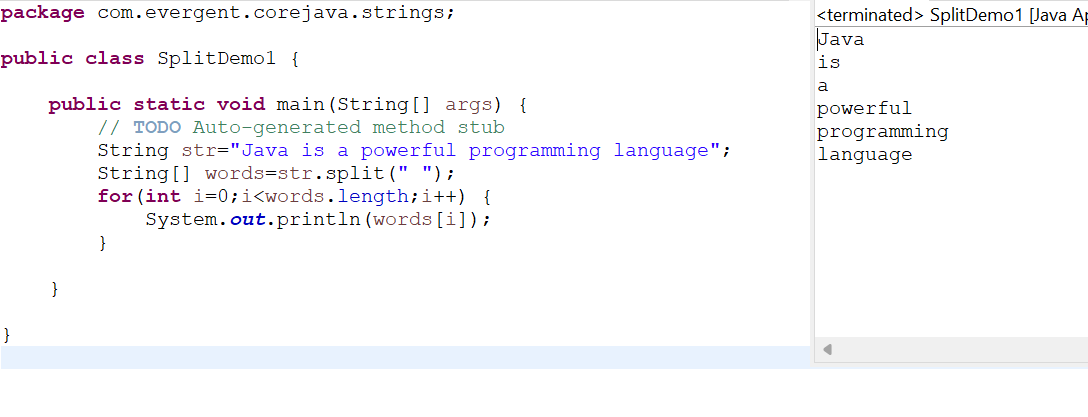


**Program 7:**

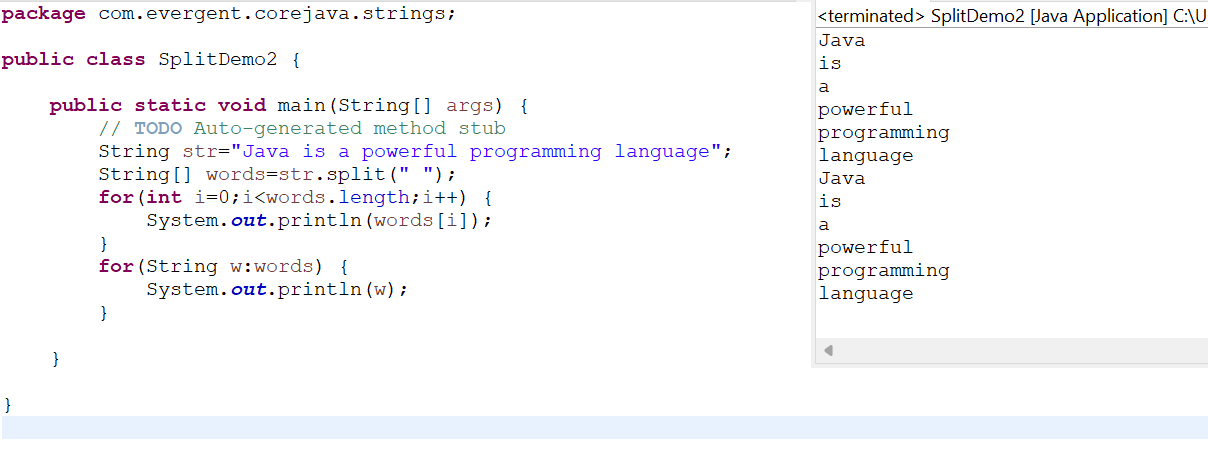
**Write a program to create a String,reverse the order of its characters,and then print out the result.**

**Program-8:**

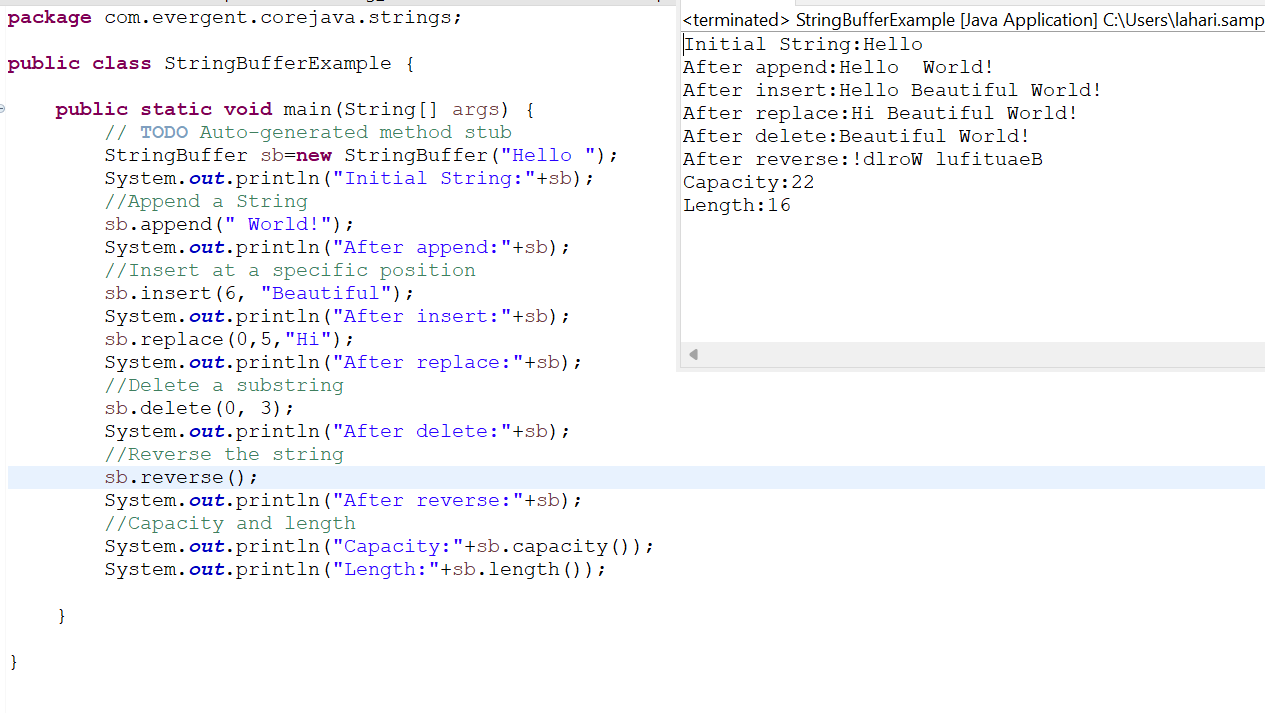
**a.Write a program to split the text on spaces.**



**8(b) Split using for-each loop**



**Program-9: String Buffer methods**

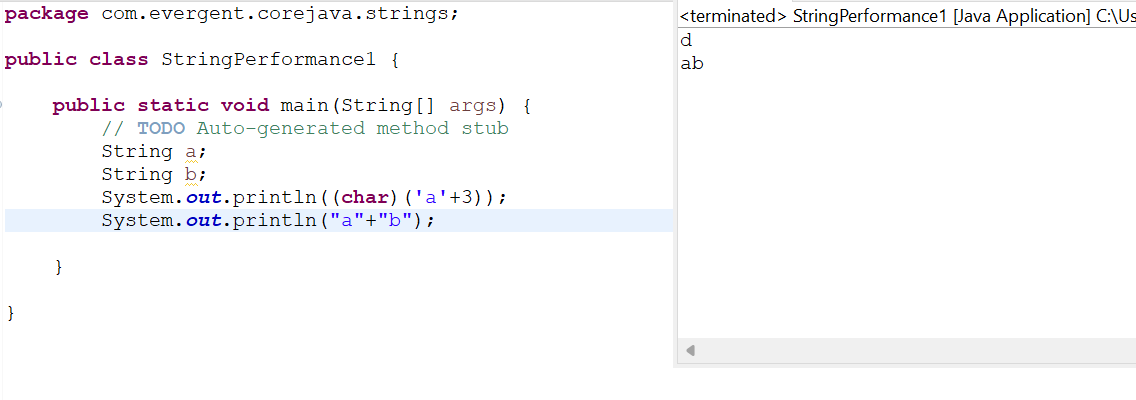


**Program-10:StringBuilder Methods**

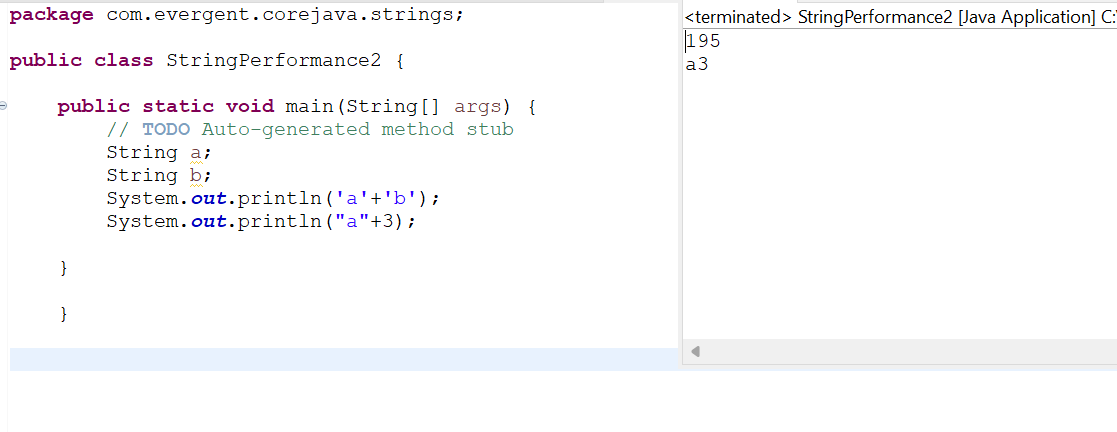


**String Performance**

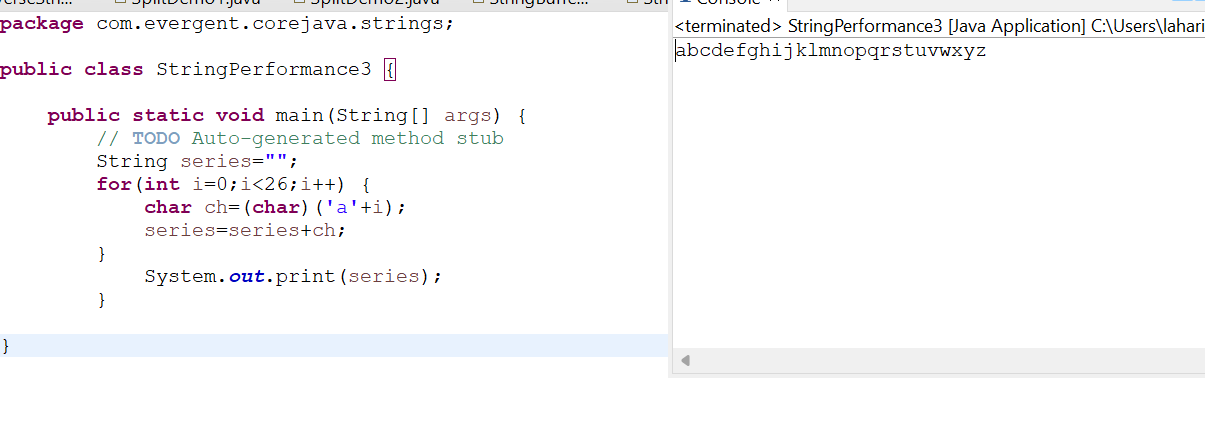
**Program - 11:**



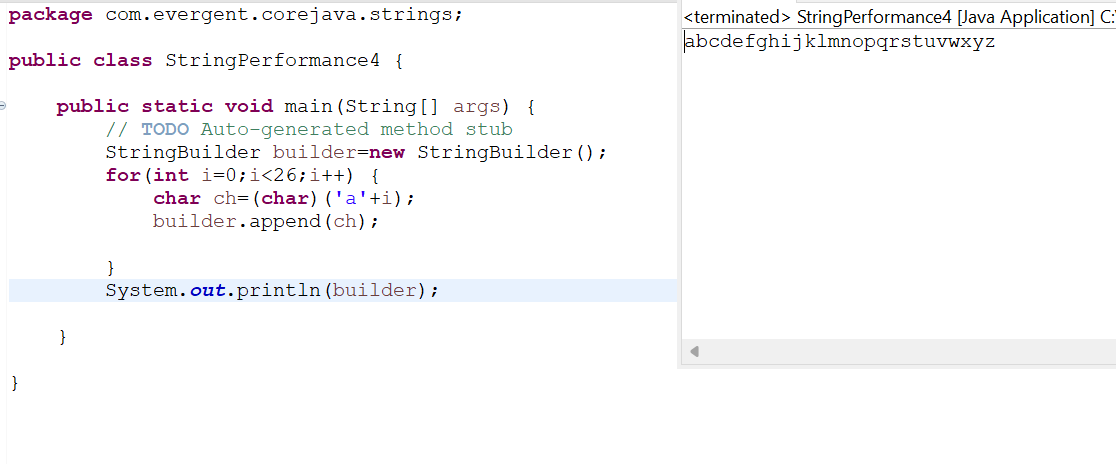
**Program - 12:**



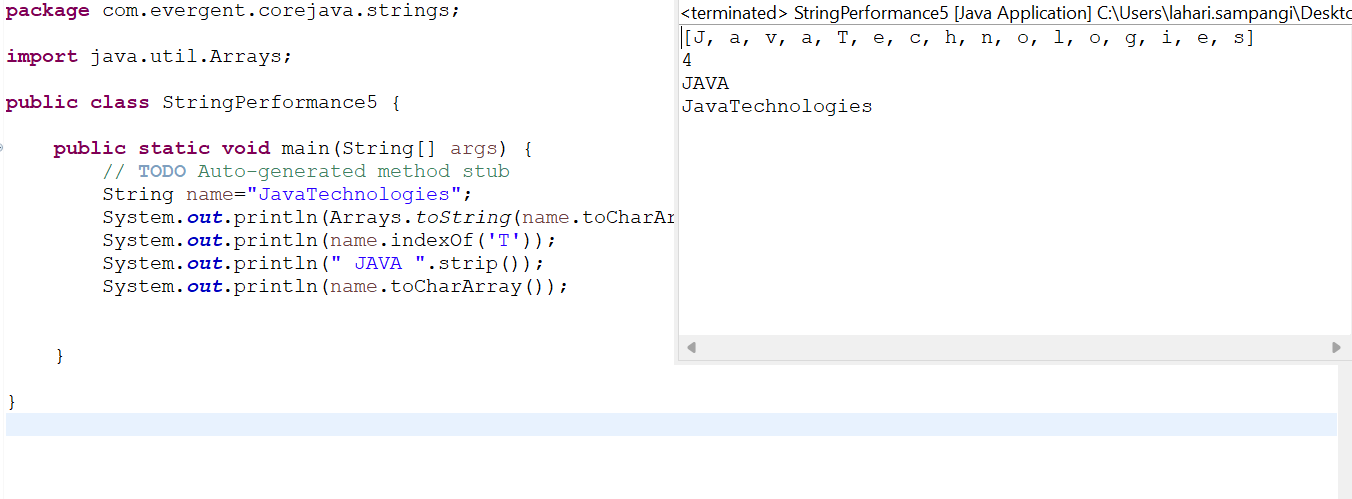
**Program-13:**



**Program-14:**



**Program-15:**



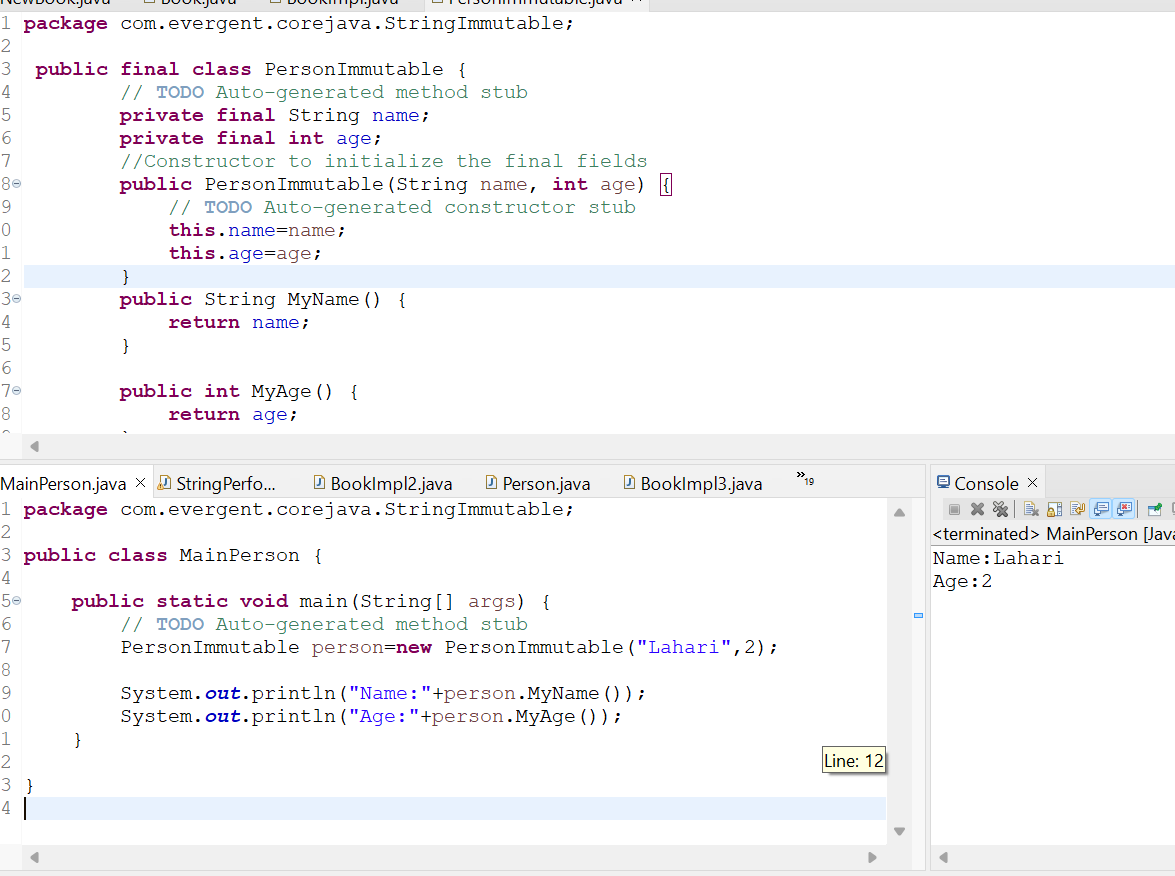
**Day-7 (13-08-2024)**

**Immutable class**

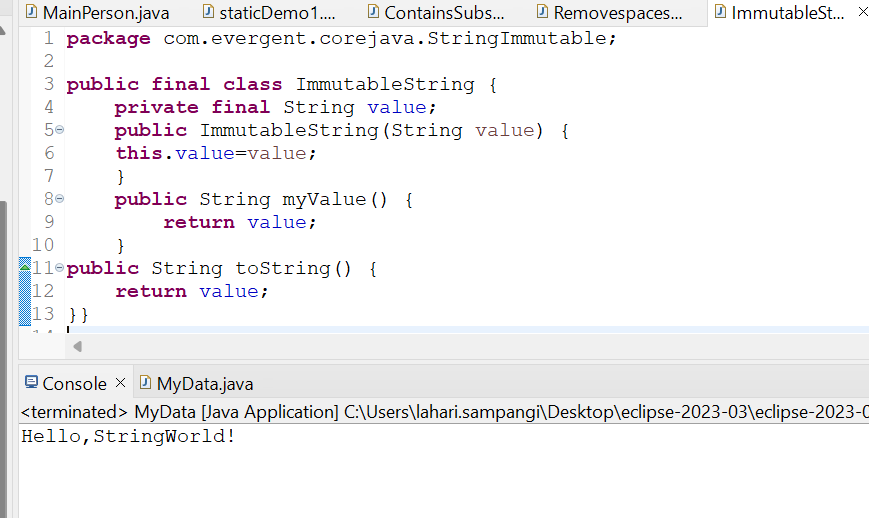
1. We can create our own immutable class.
2. We can declare class as final
3. The class is declared as final so that it cannot be subclassed.

**PROGRAMS:**

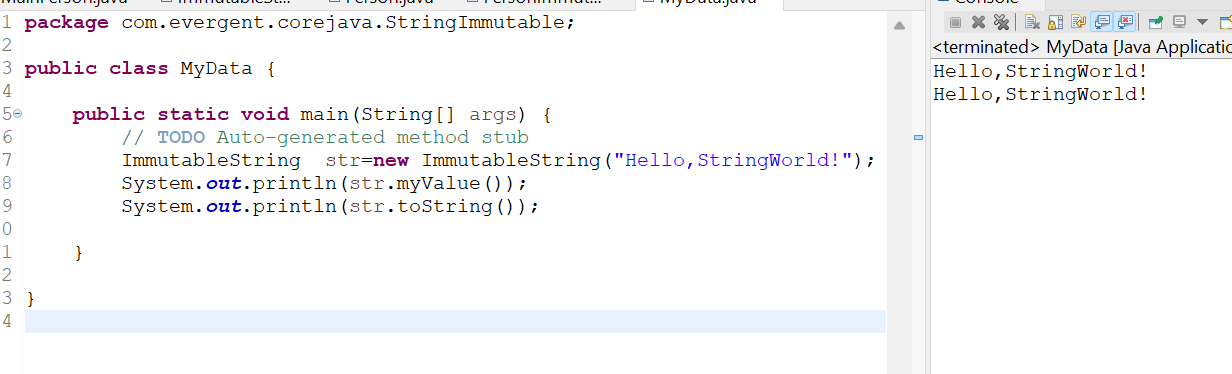
**Program-1: Immutable**



**Program-2: final class Immutable**



**Program-3:**

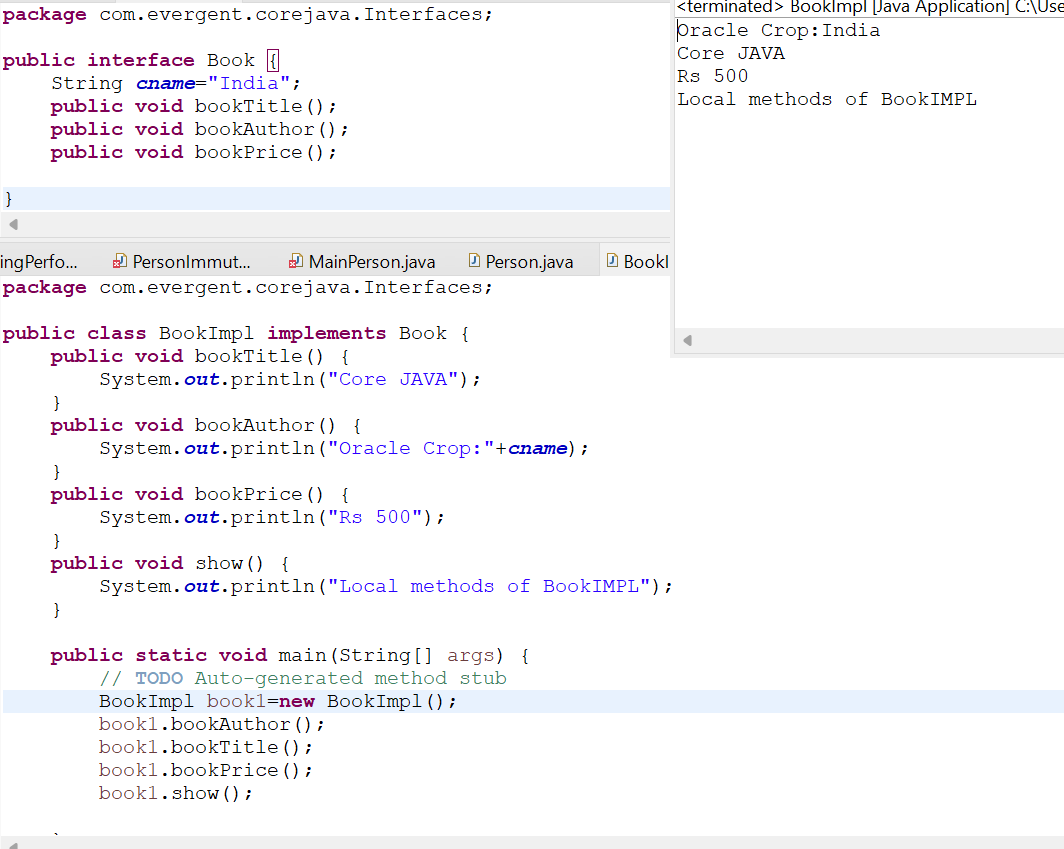


**INTERFACES**

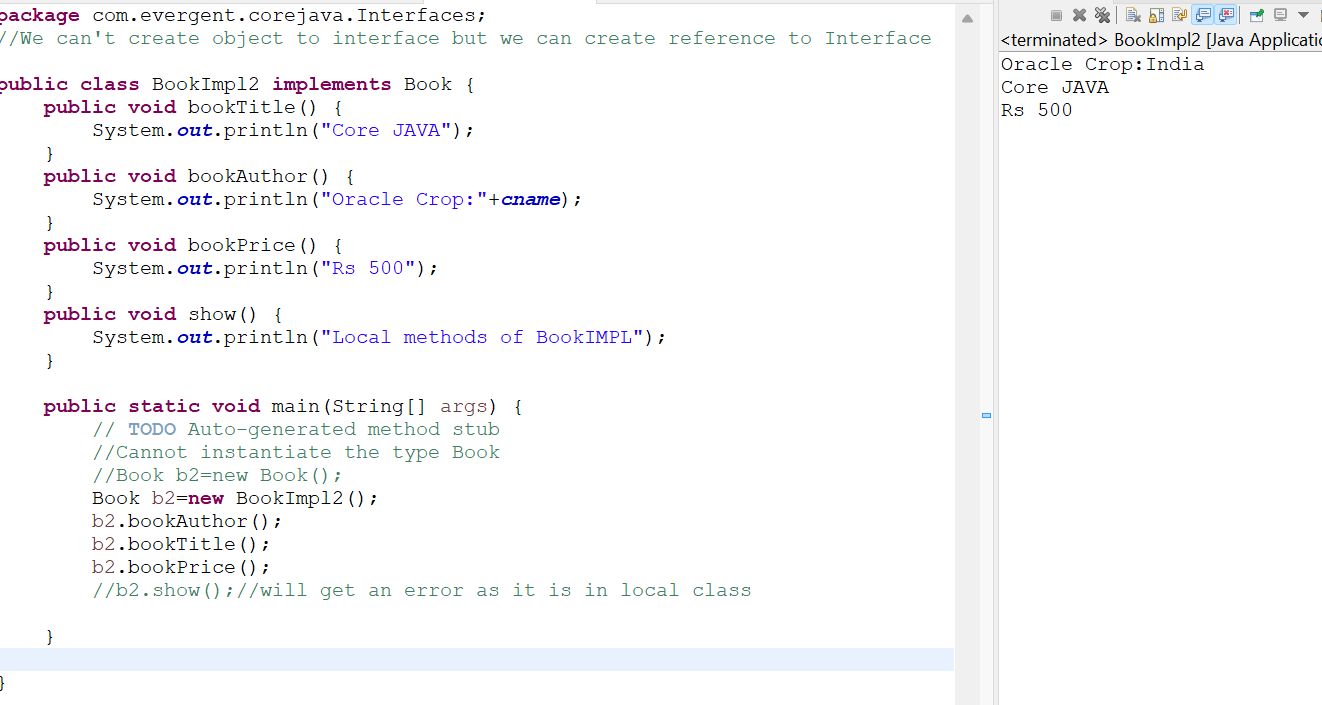
1. Interface is a keyword.
2. We can declare methods signatures only but not implementation.
3. By default,all interface methods are abstract.
4. If any class implements interface that class should be override all interface methods otherwise the class will be showing compile time error.
5. We cannot create object to interface but we can create reference to interface.
6. We can declare variables inside interface and all are public static final.
7. Java will support multiple inheritance through interface.
8. One class can implements more than one interface.
9. One interface can extend other interface.

**PROGRAMS:**

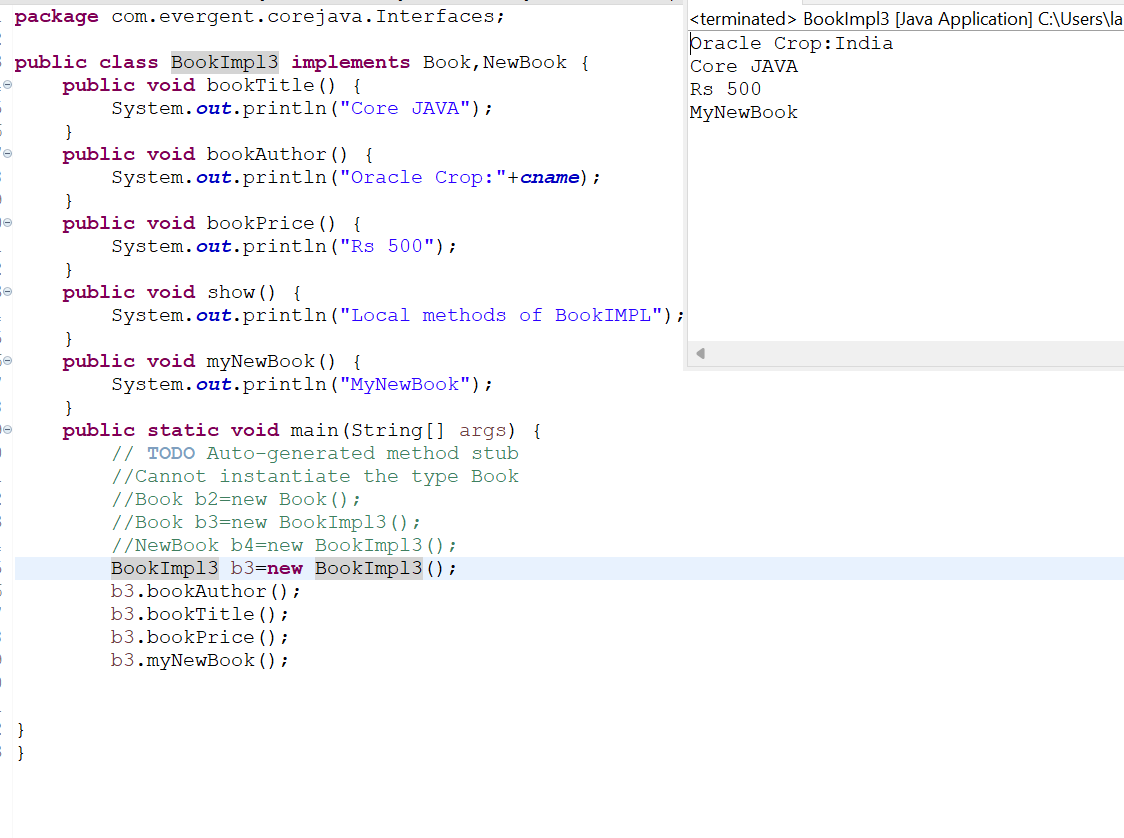
Program1: Example of Interface



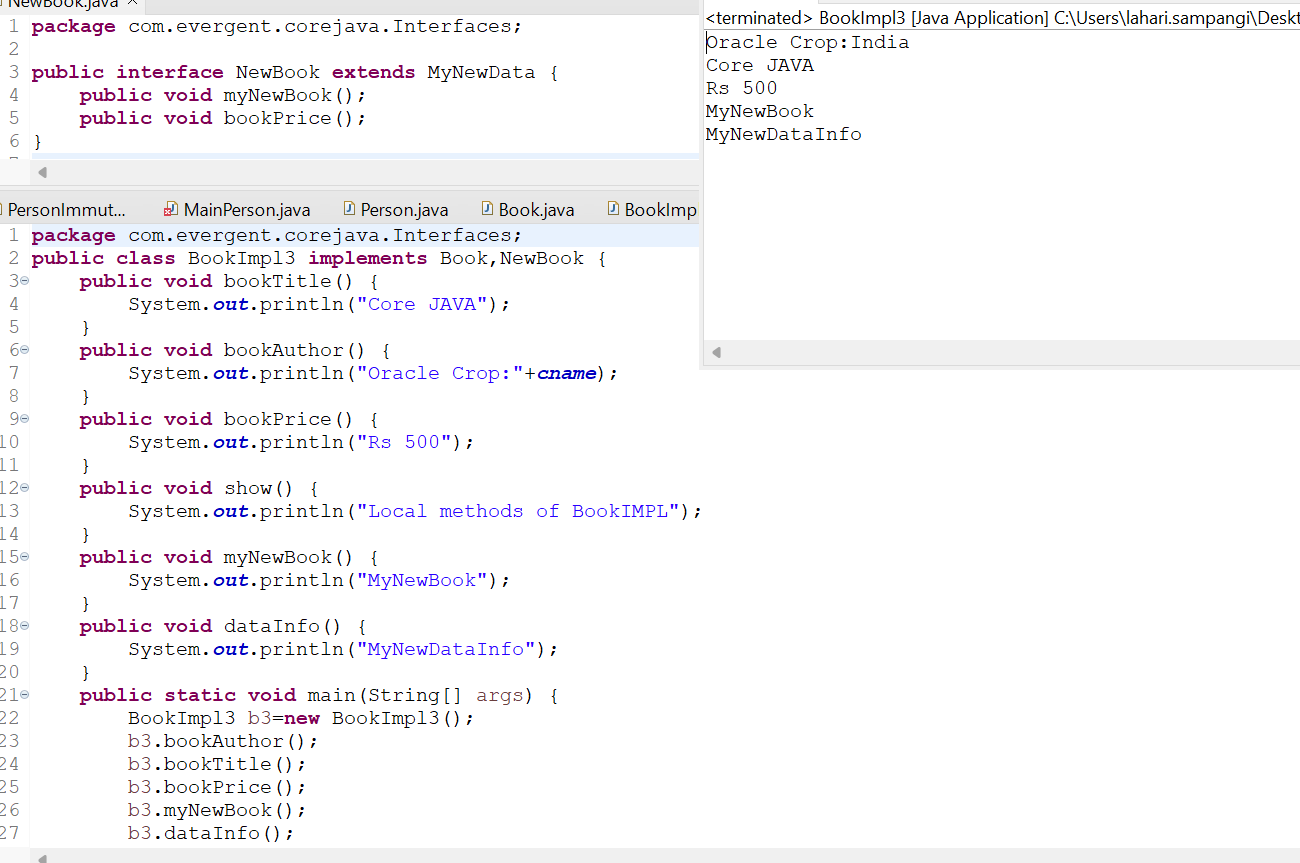
**Program-2:**



**Program-3:Multiple inheritance**



**Program-4: Interface Inheritance**

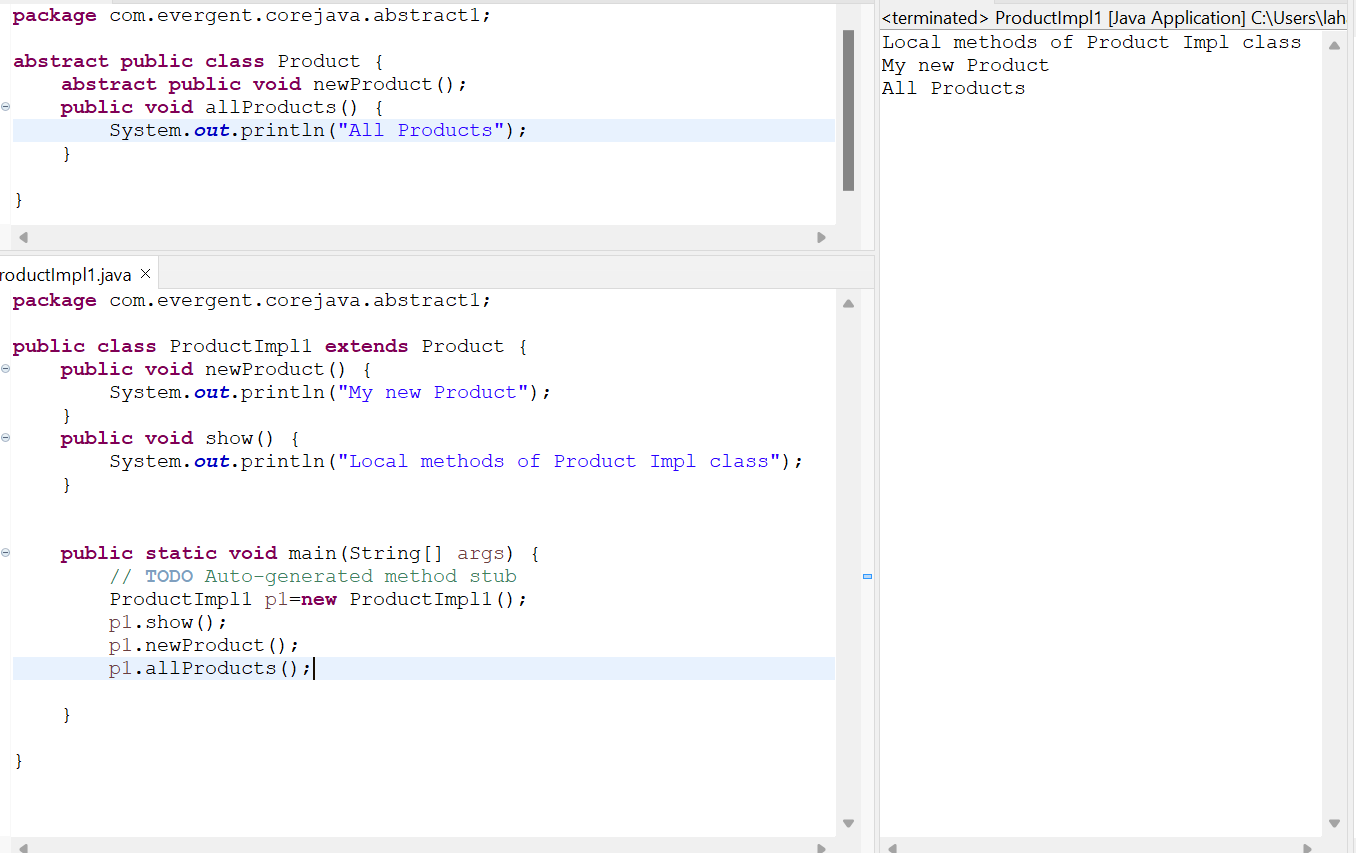


**ABSTRACTION:**

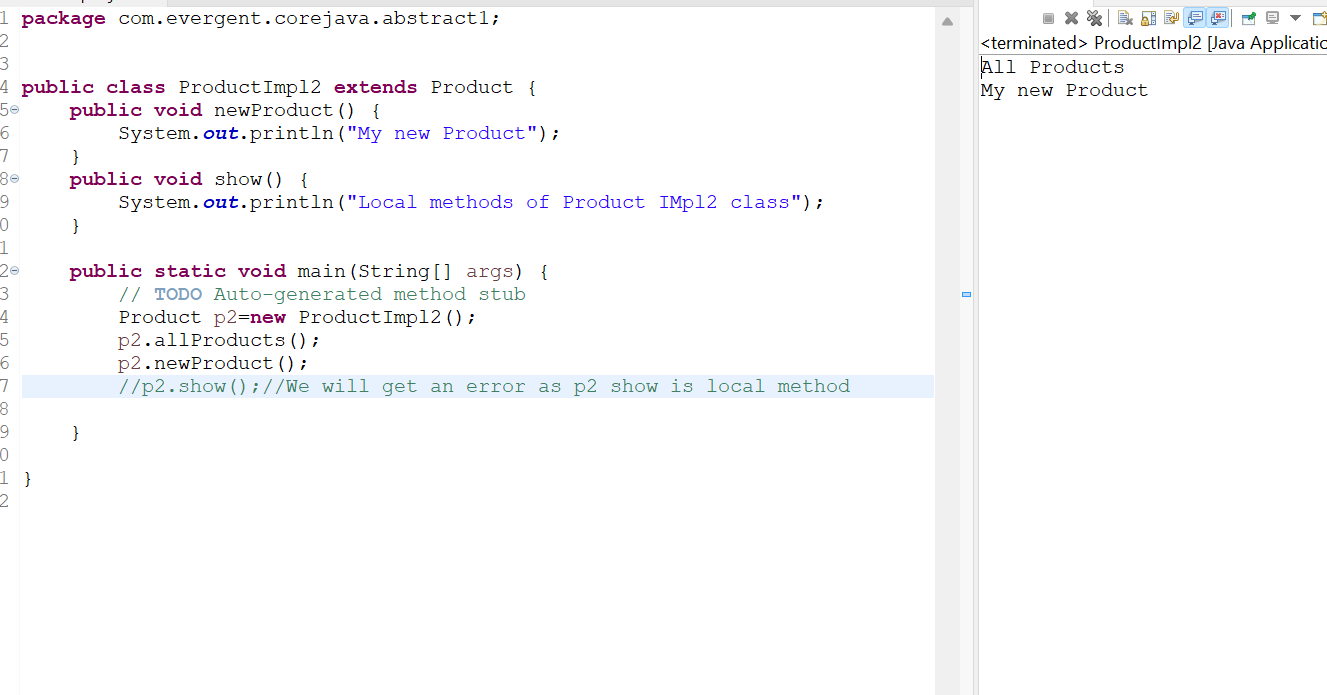
1. Abstract is a keyword
2. Abstract class having abstract methods and concrete(implemented methods)
3. If any class having one abstract method that class should be declared as a abstract keyword,otherwise the class will be showing compile time error.
4. If any class extends abstract class that class should be override all abstract methods otherwise the class will be showing compile time error.
5. We can’t create object to abstract class but we can create reference to abstract class.

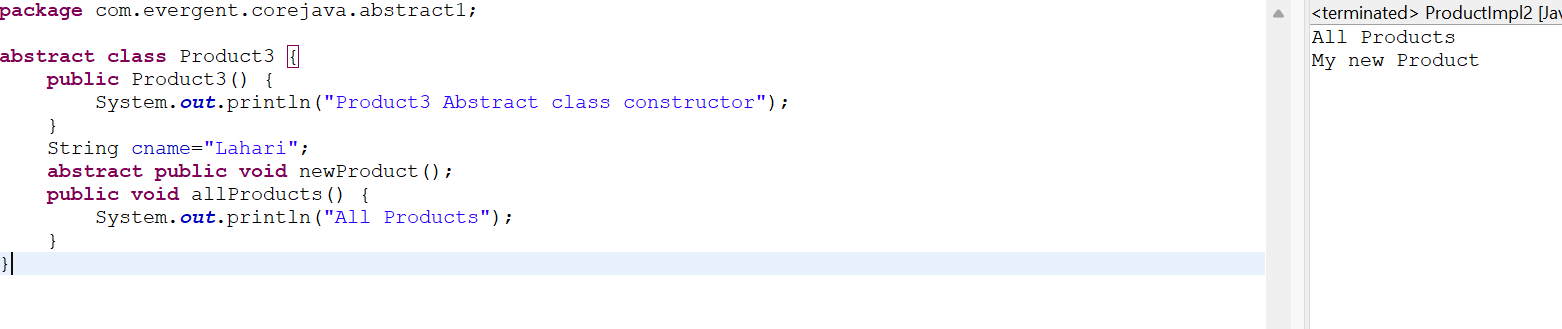
**PROGRAMS:**

**Program-1:Abstract class Example**

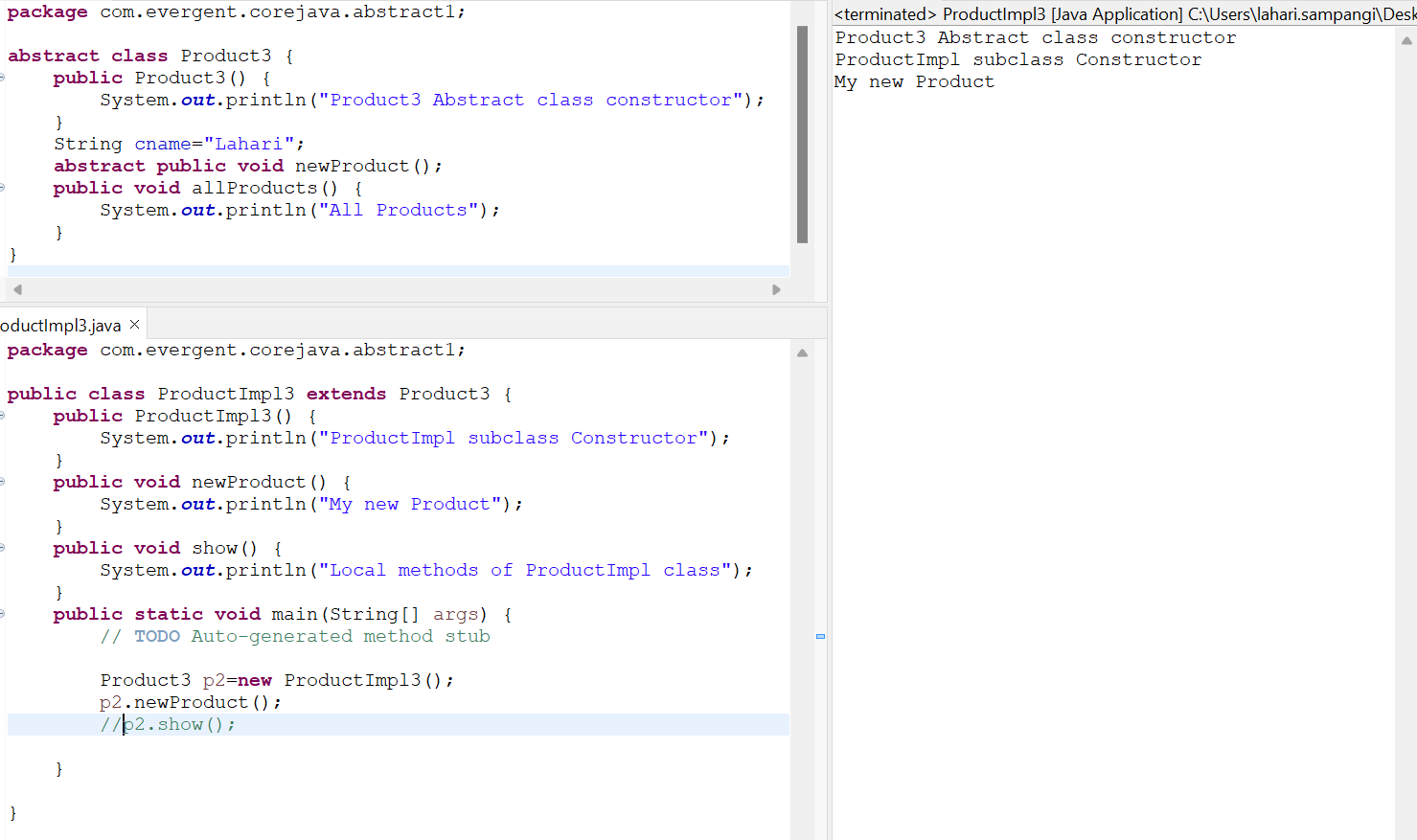


Program-2:



Program-3:  


Program-4



**EXCEPTION HANDLING:**

1. Exception Handling is a mechanism.
2. Exceptions are in built mechanism of Java.
3. All exceptions are executed while abnormal conditions only.
4. Normal flow it won’t execute any exceptions.
5. Once any exceptions are occurring in Java then remaining lines of code is unreachable.
6. Java.lang.Throwable is super class for Exceptions and Errors.
7. There are two types of Exceptions in Java.

* Checked exceptions
* Unchecked exceptions

1. All checked exceptions are compile time exceptions.
2. All unchecked exceptions are Runtime exceptions.
3. There are 5 keywords in Exception Handling.

* try()
* catch()
* finally()
* throws()
* throw()

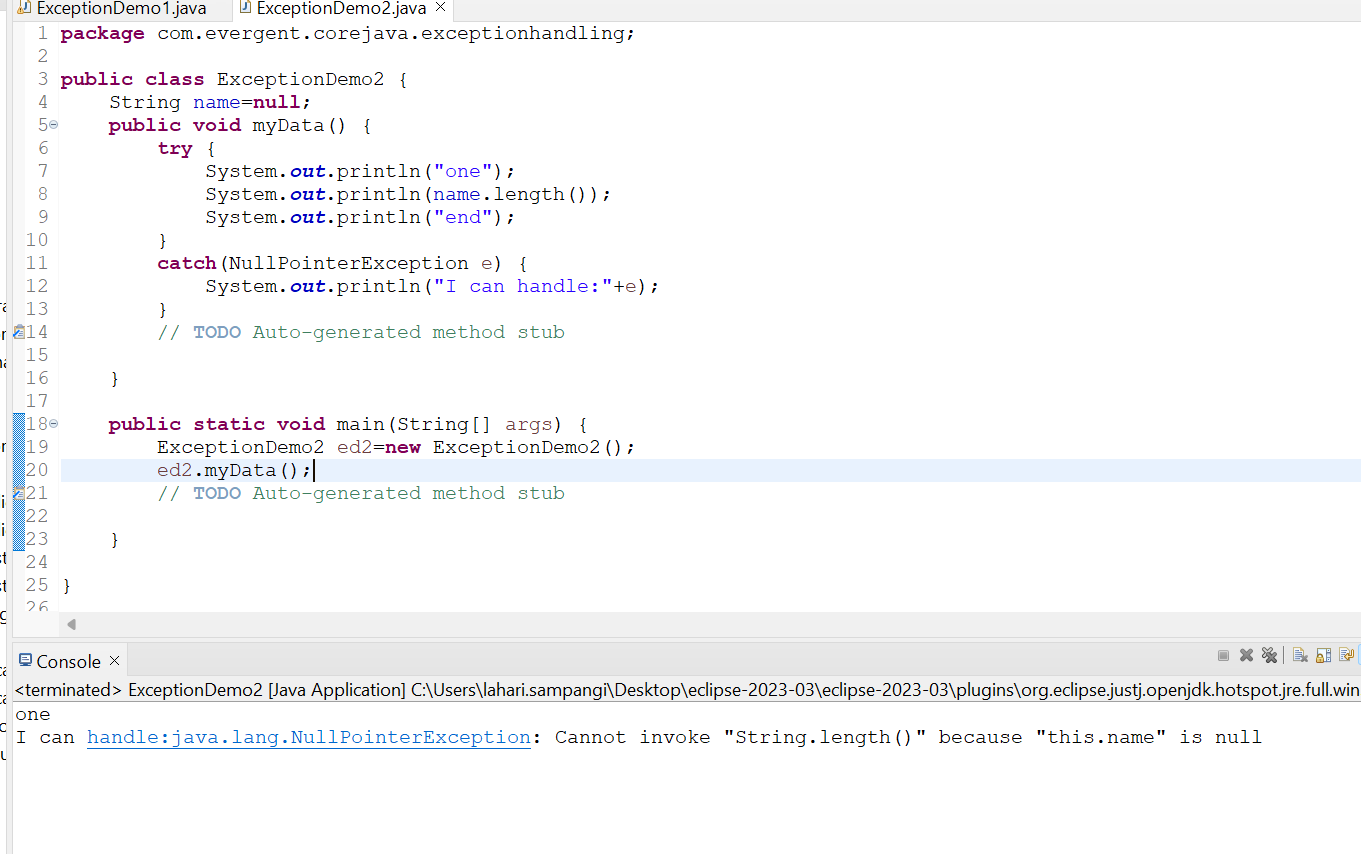
1. Try is for business logic.
2. Catch is for handling exceptions.
3. Finally is block,if exceptions is occurs or not finally block will be executed.
4. Throws an exception will be executed method by method.
5. Throw is for runtime exceptions and will call predefined exceptions or user defined exceptions.
6. Try followed by either catch block or finally block.
7. We should follow exceptions hierarchical.
8. We can create our own(user defined) exceptions.
9. Our own exceptions extends Exception or Runtime Exception.
10. All Exceptions classes are in to java.lang package.
11. There is two exceptions in class,Developer should be handled one after one.
12. Developer can’t handled errors.

**Programs:**

**Program-1:Example demo**



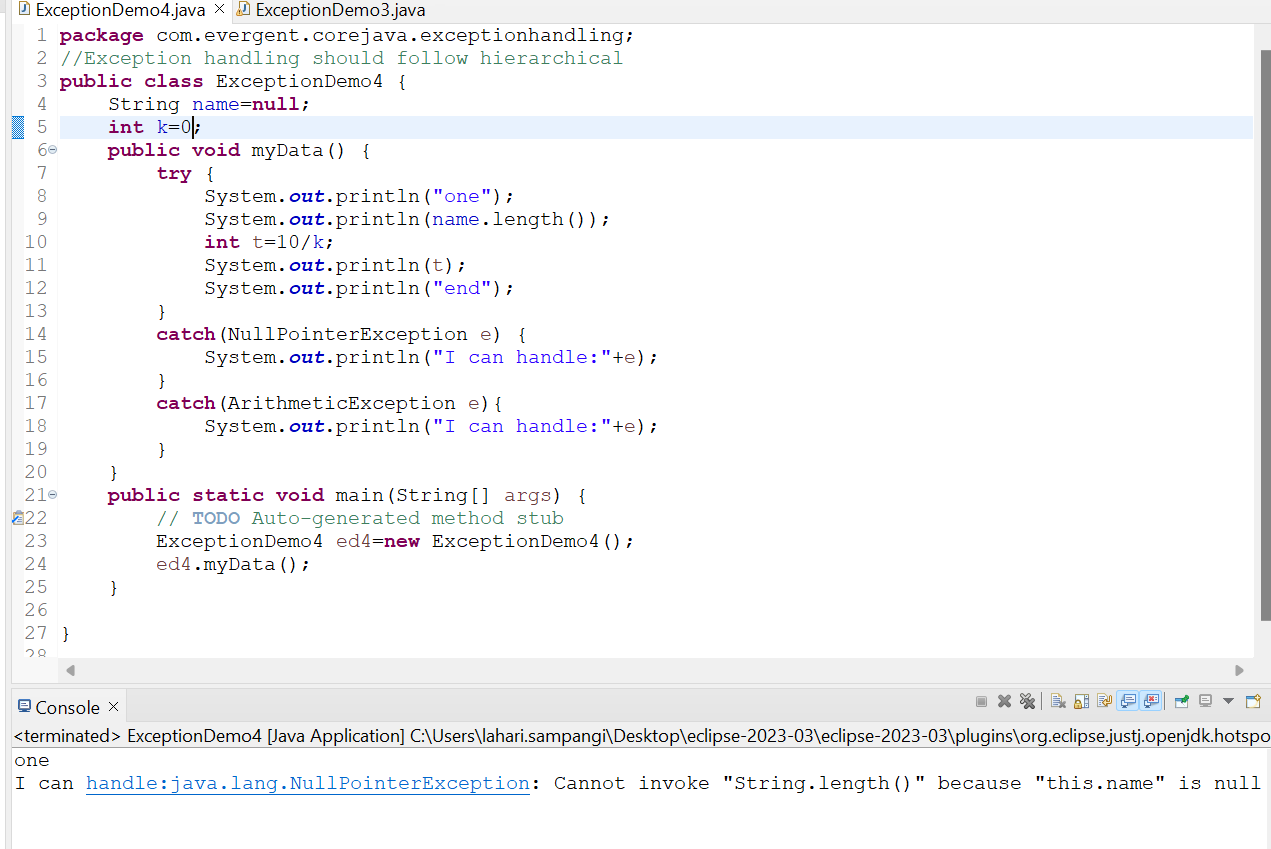
**Program-2: try and catch block**



**Program-3:Multiple Catch(Handling when we get multiple exceptions)**



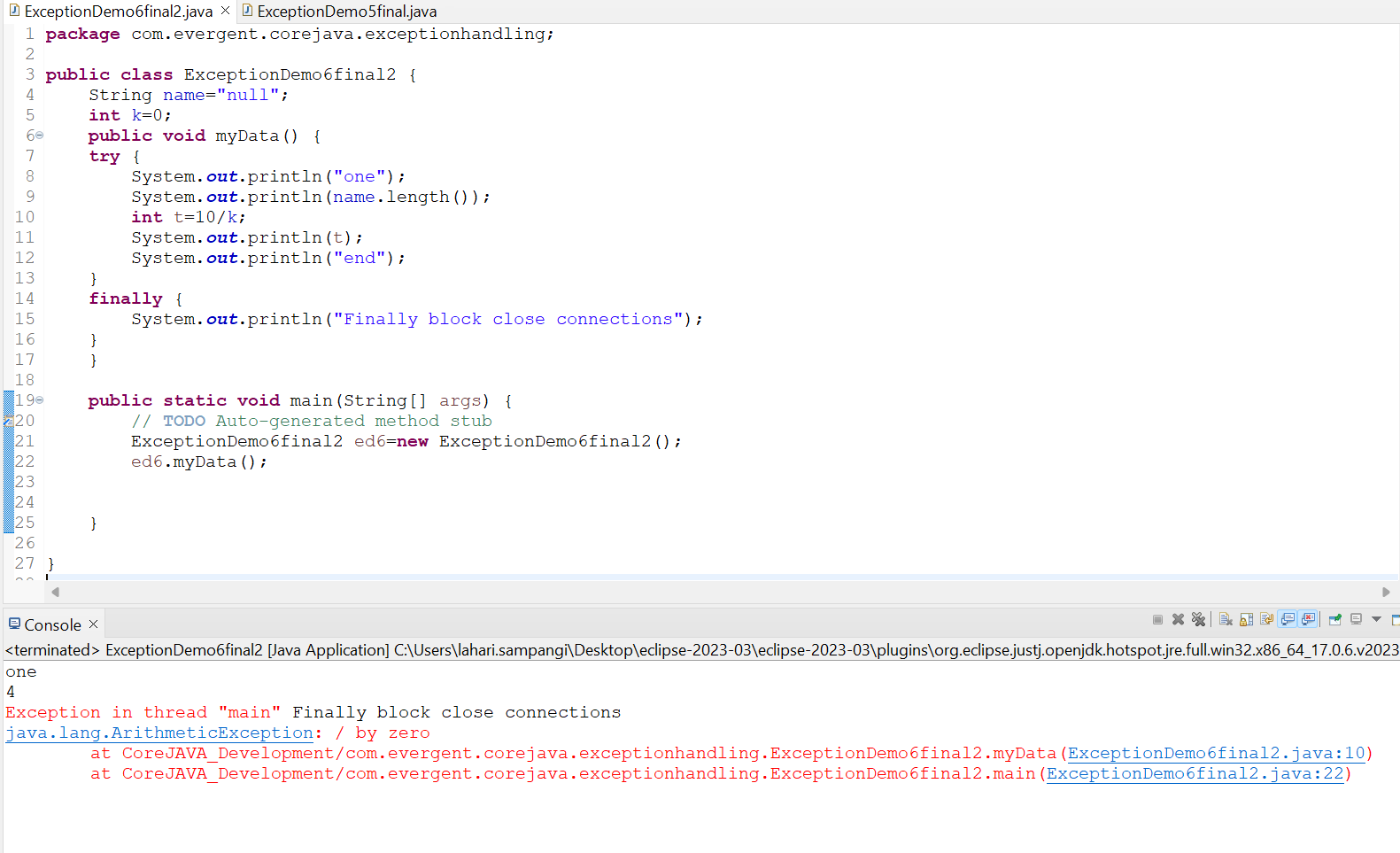
**Program-4: Hierarchical Exception**



**Program-5:Finally Block**



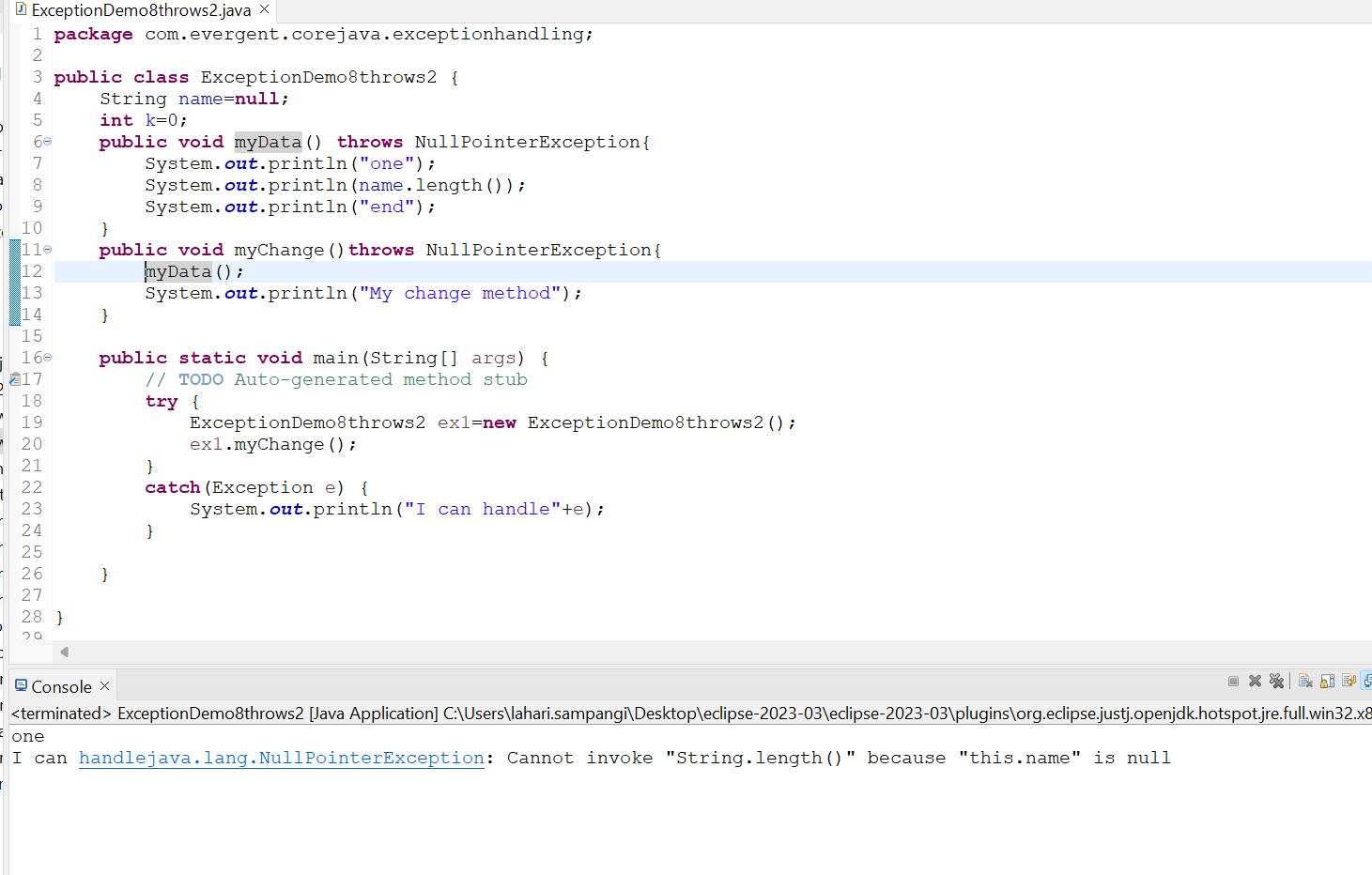
Program-6:



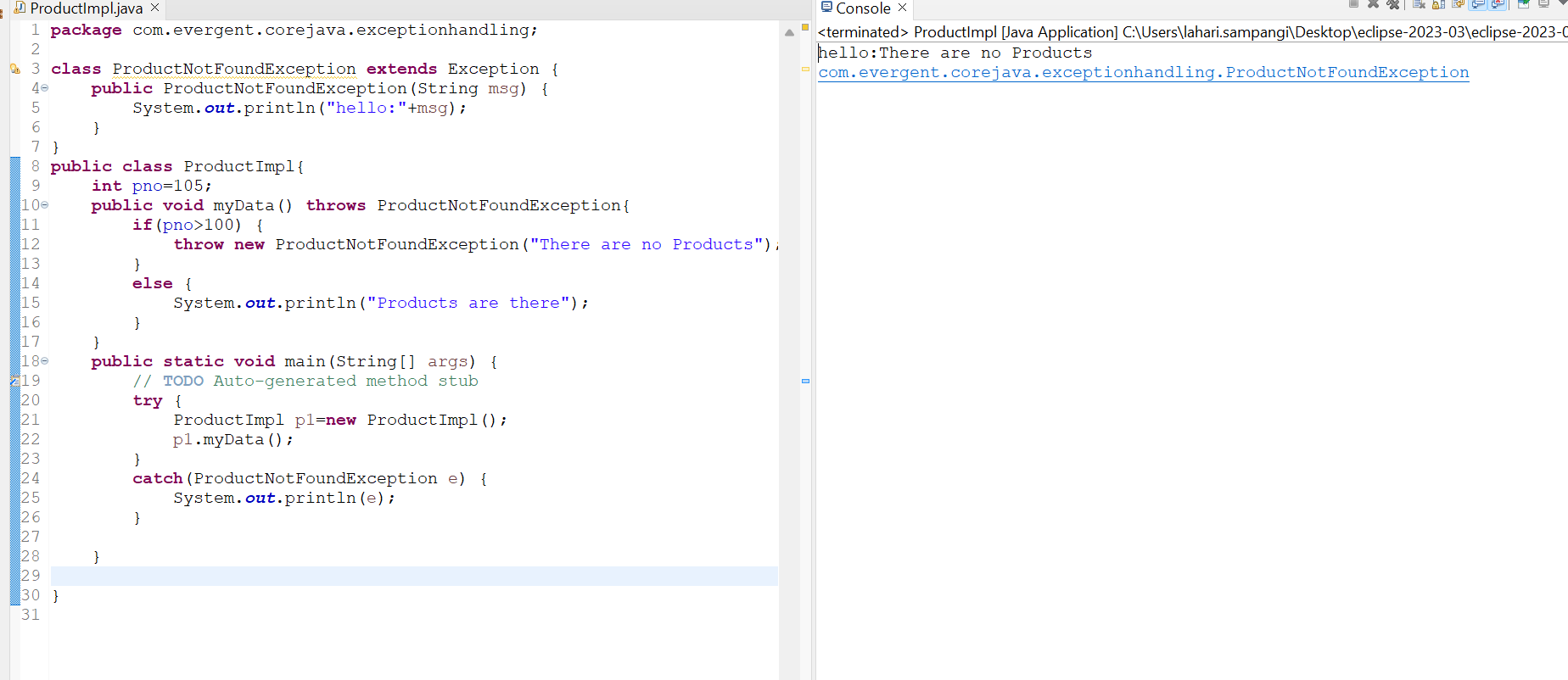
**Program-7:Throws**



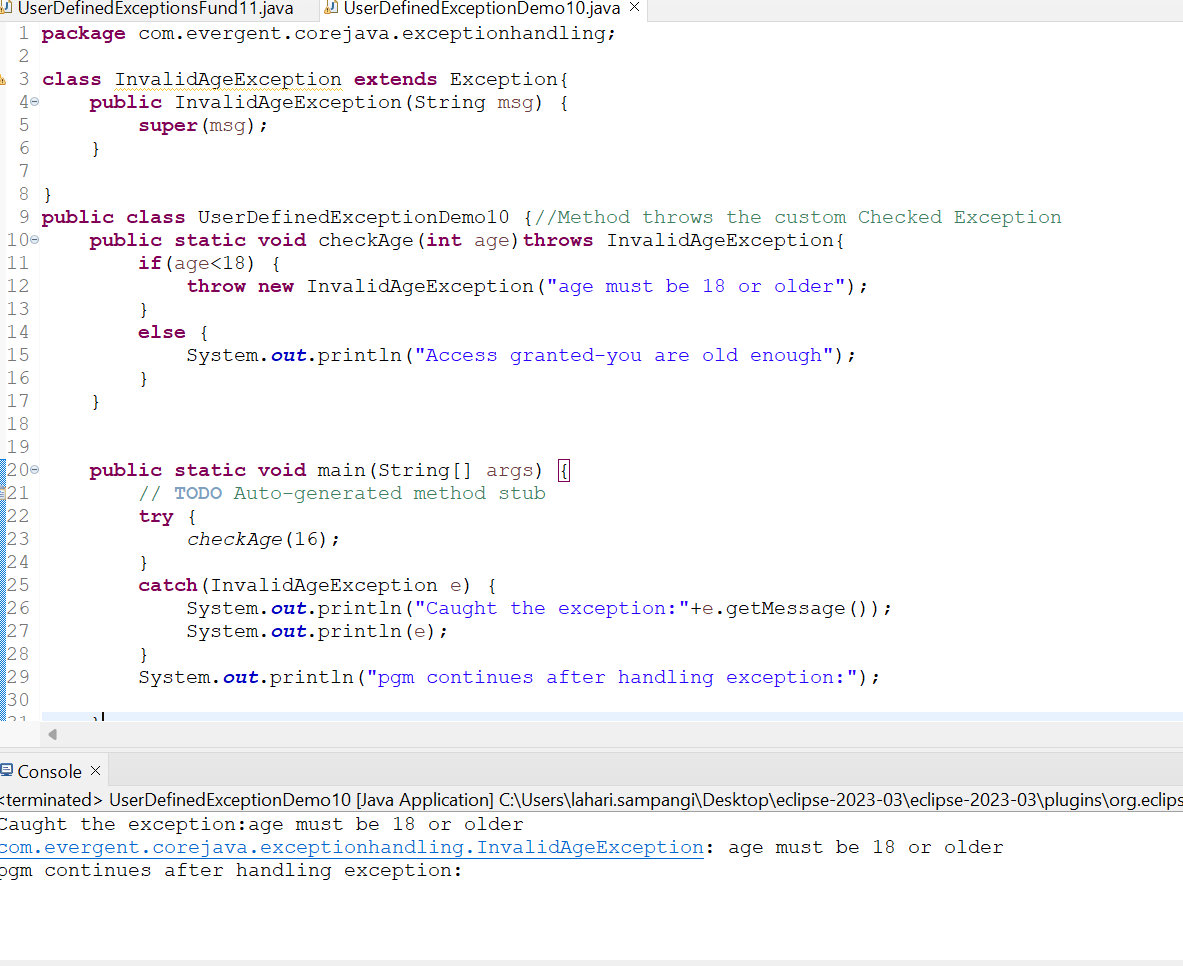
**Program-8: Handling exceptions in different methods**



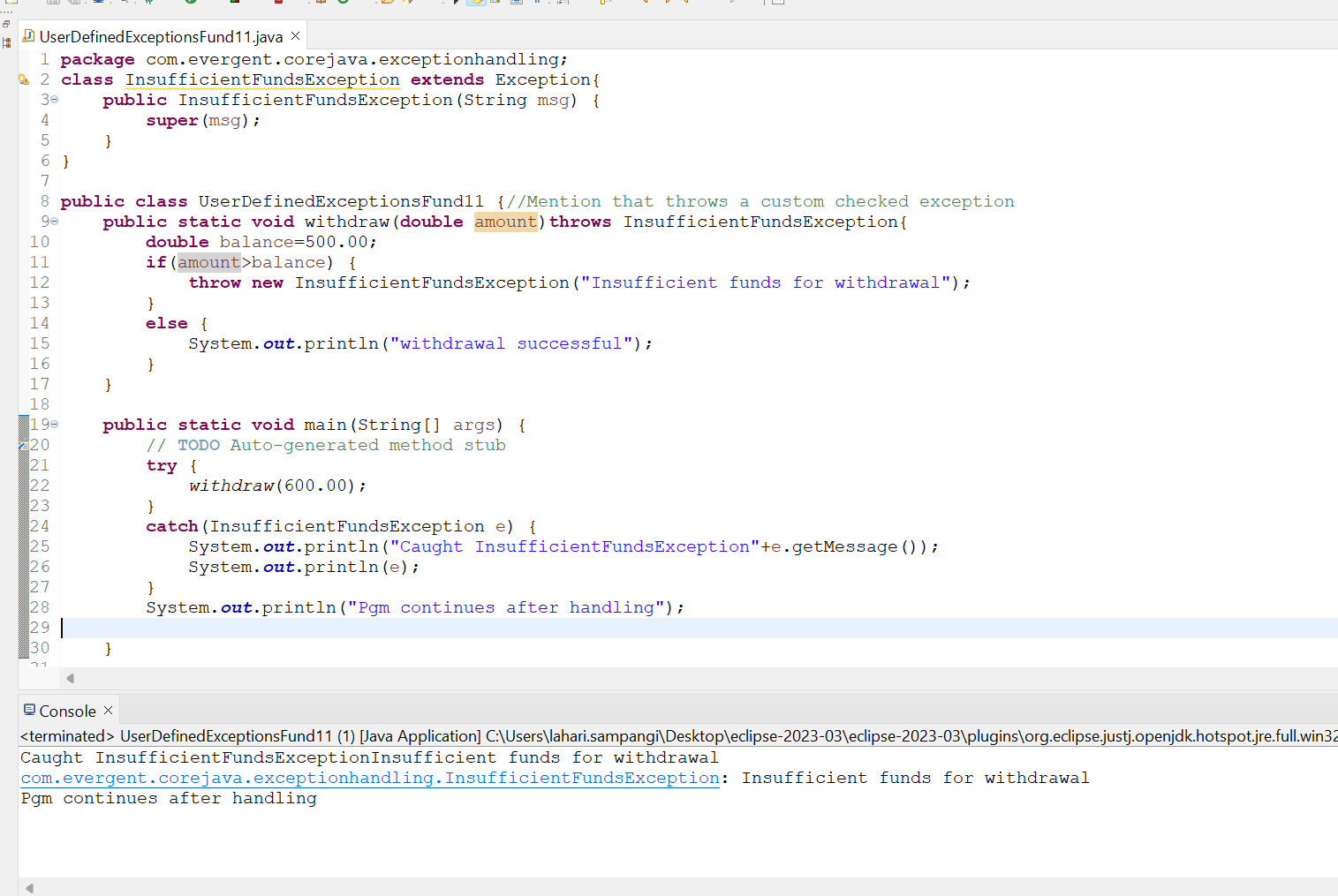
**Program-9:User defined Exceptions**



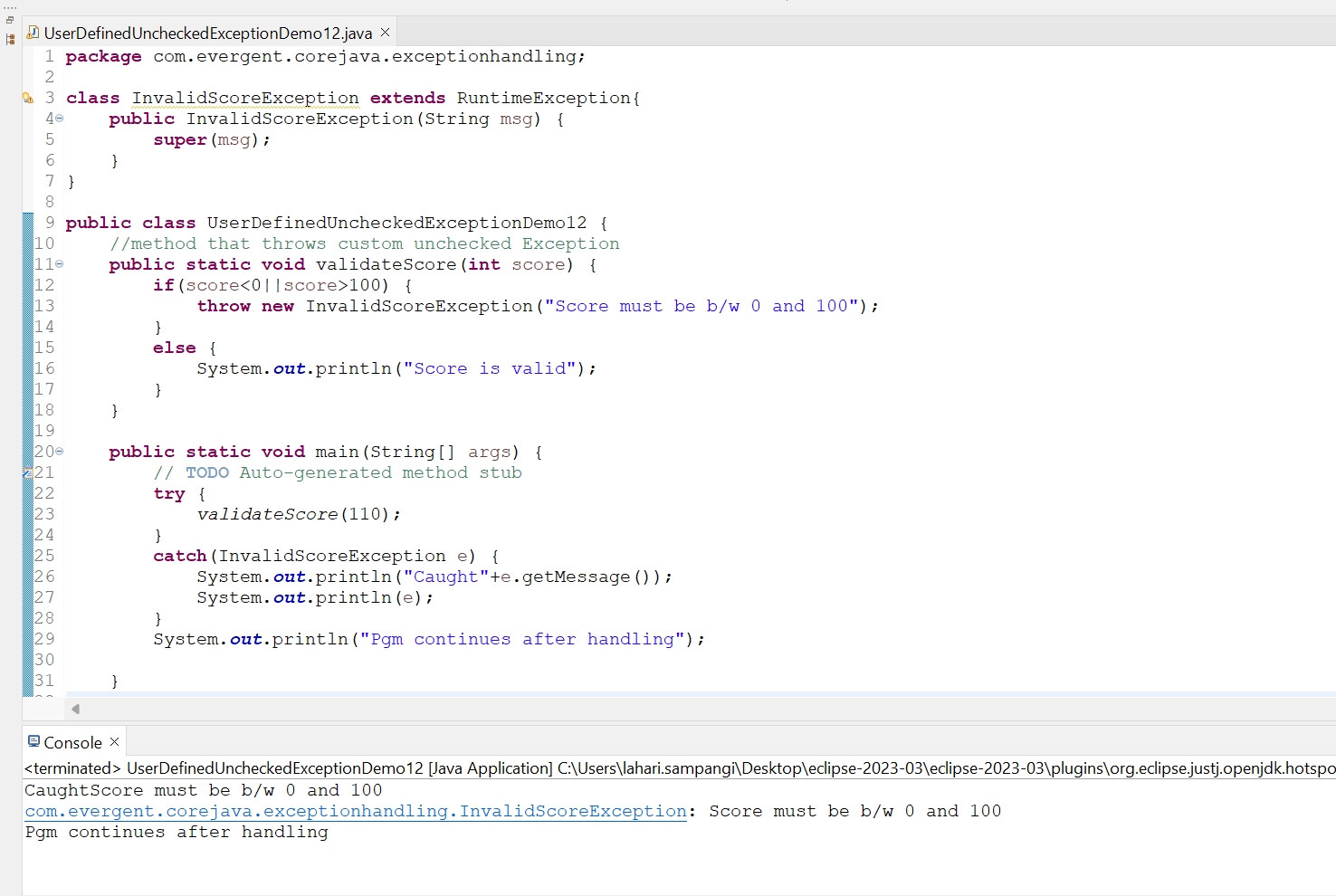
**Program-10:User defined Exceptions**



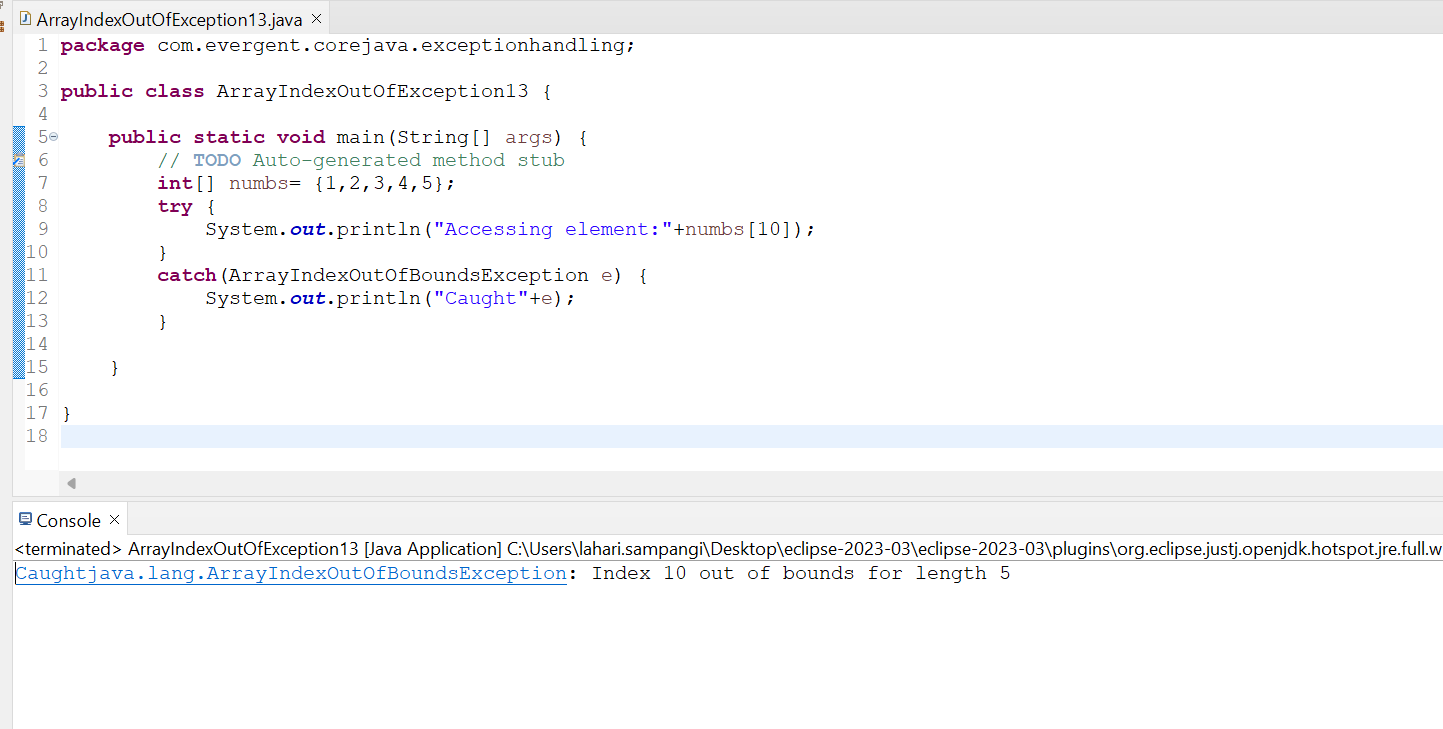
**Program-11: InsufficientFundsException**



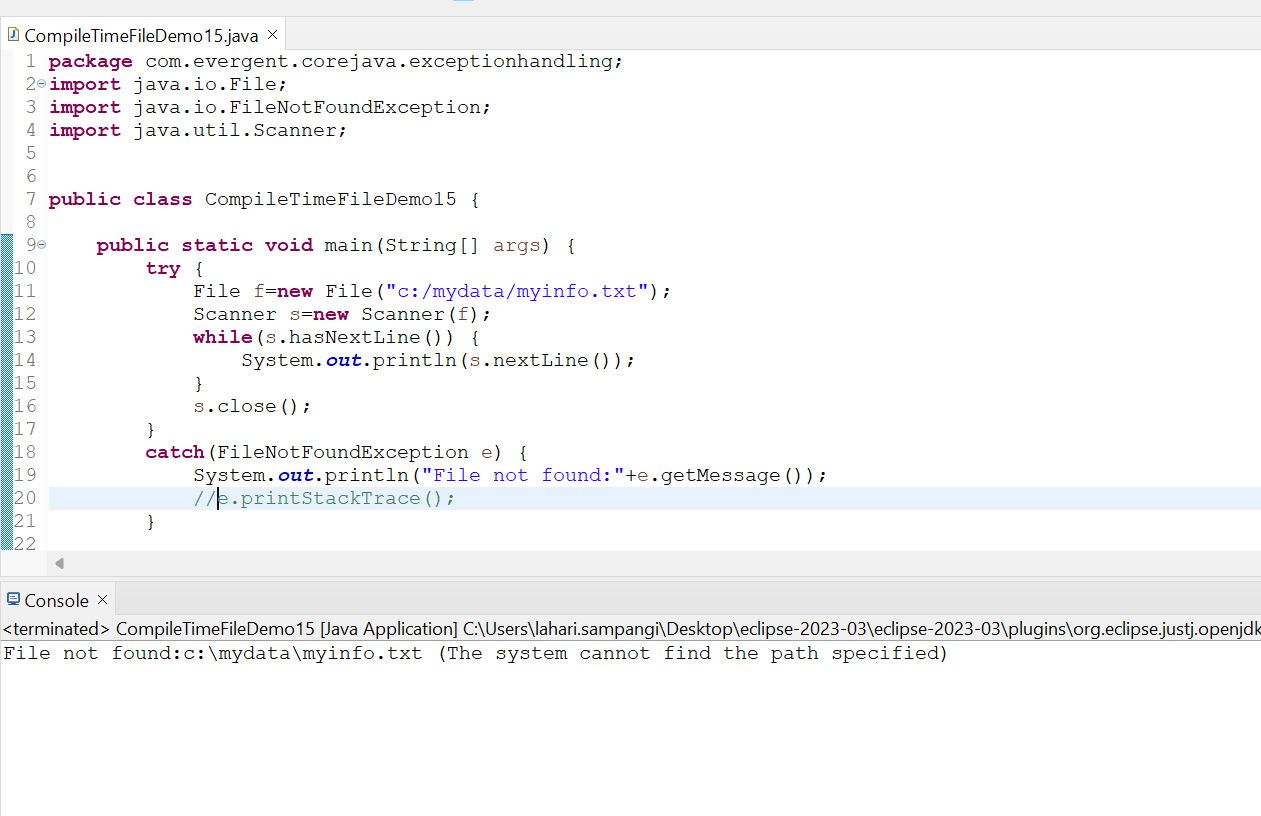
**Program-12: InvalidScoreException**



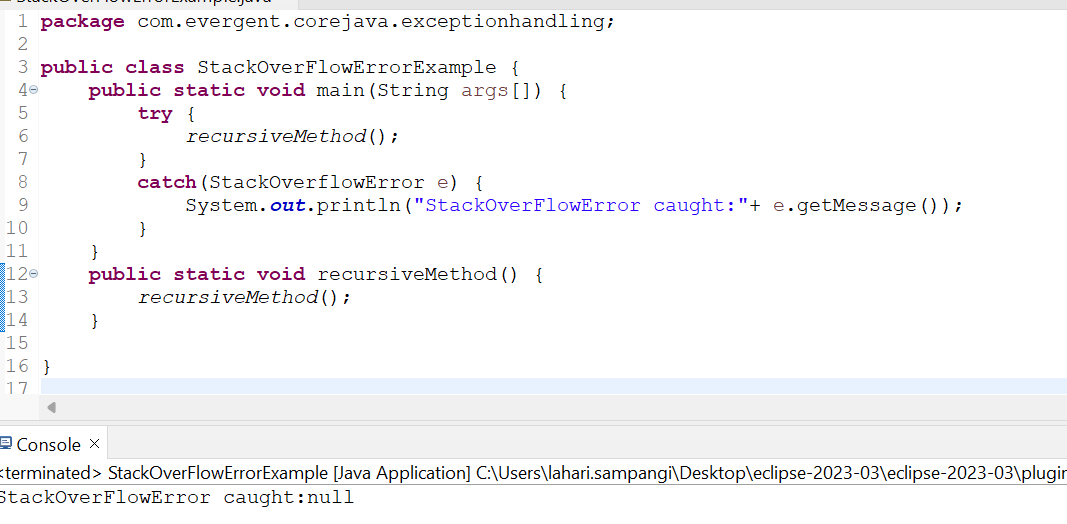
**Program-13: ArrayIndexOutOfBoundsException**



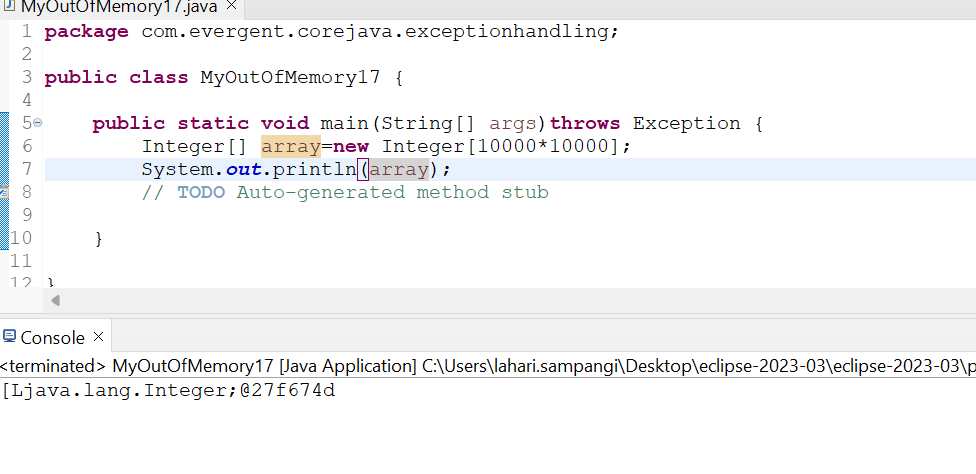
**Program-14: FileNotFoundException**



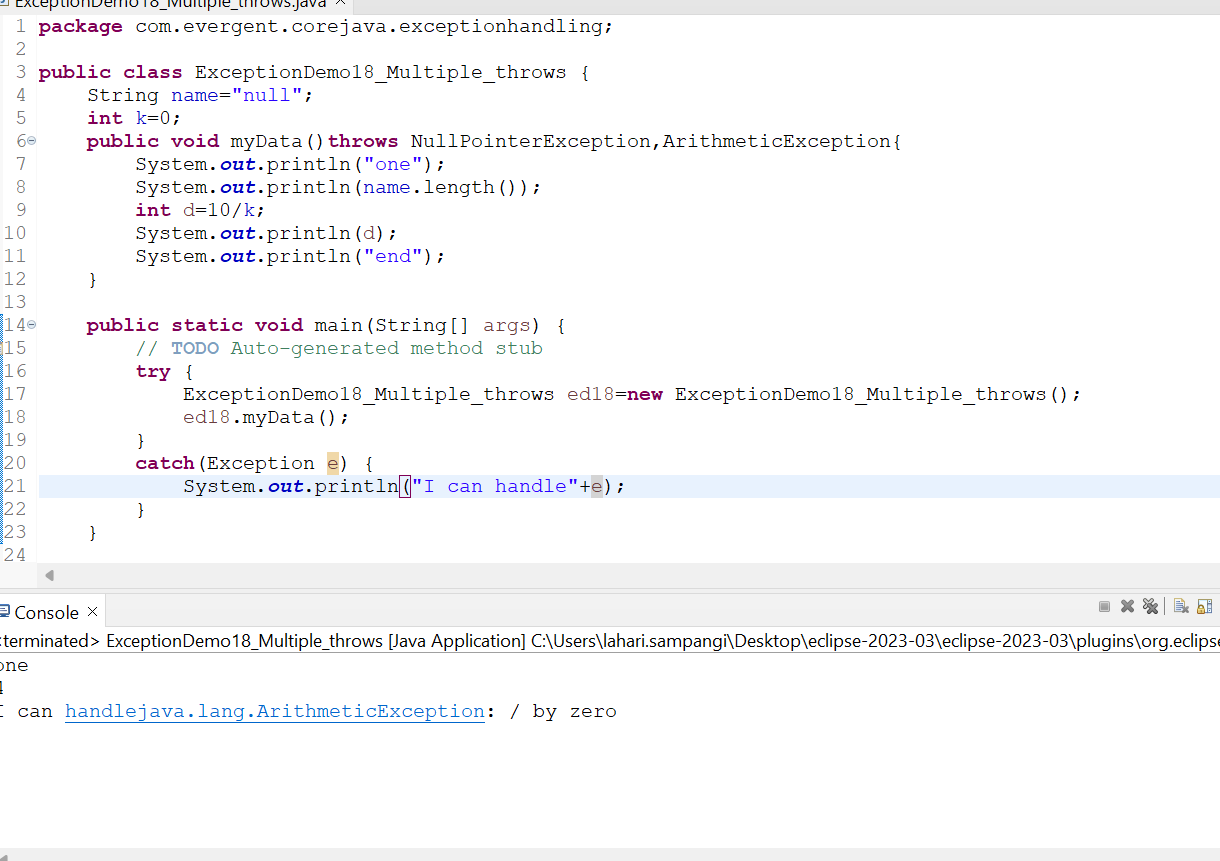
**Program 16:StackOverFlowErrorExample**



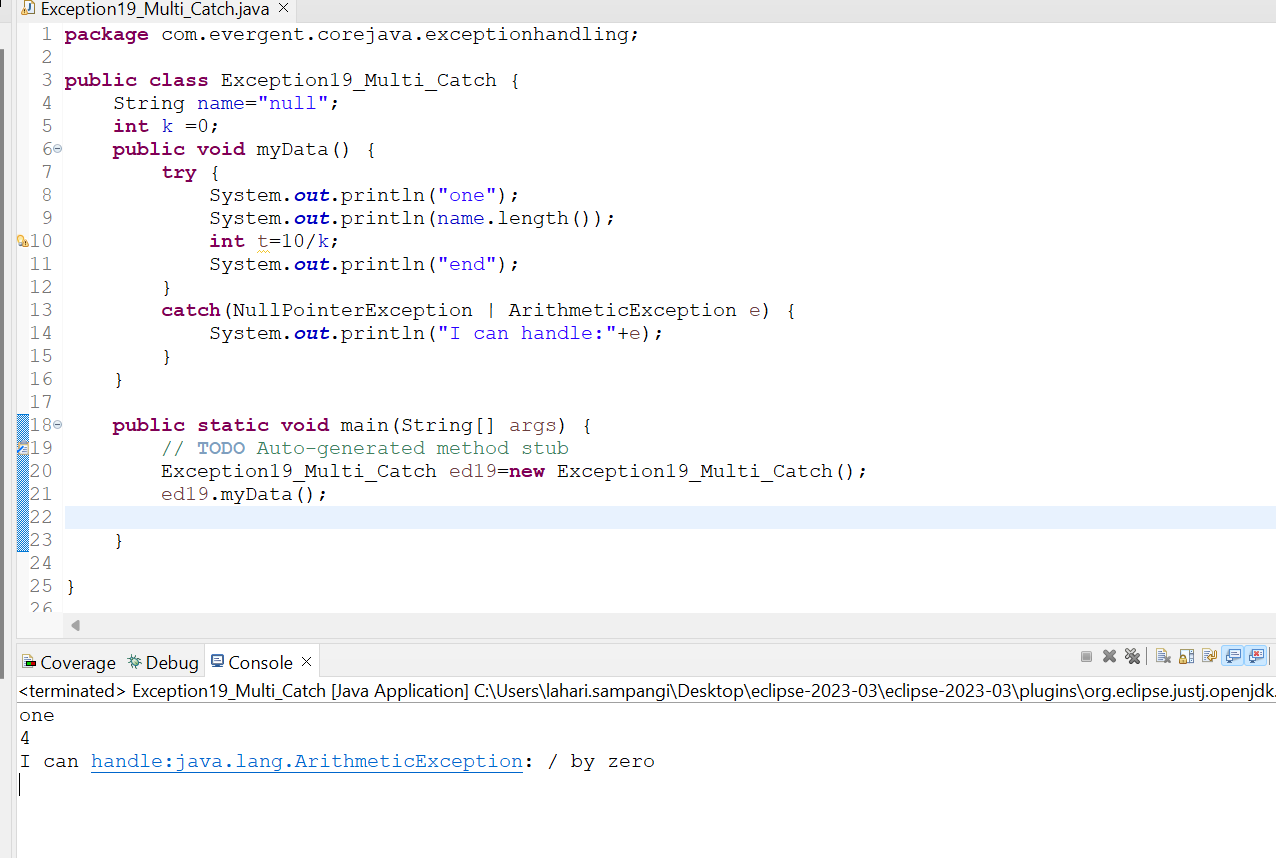
**Program-17: Outofmemory**

****

**Program-18:Multiple Throws**



**Program-19 : Multi-Catch**



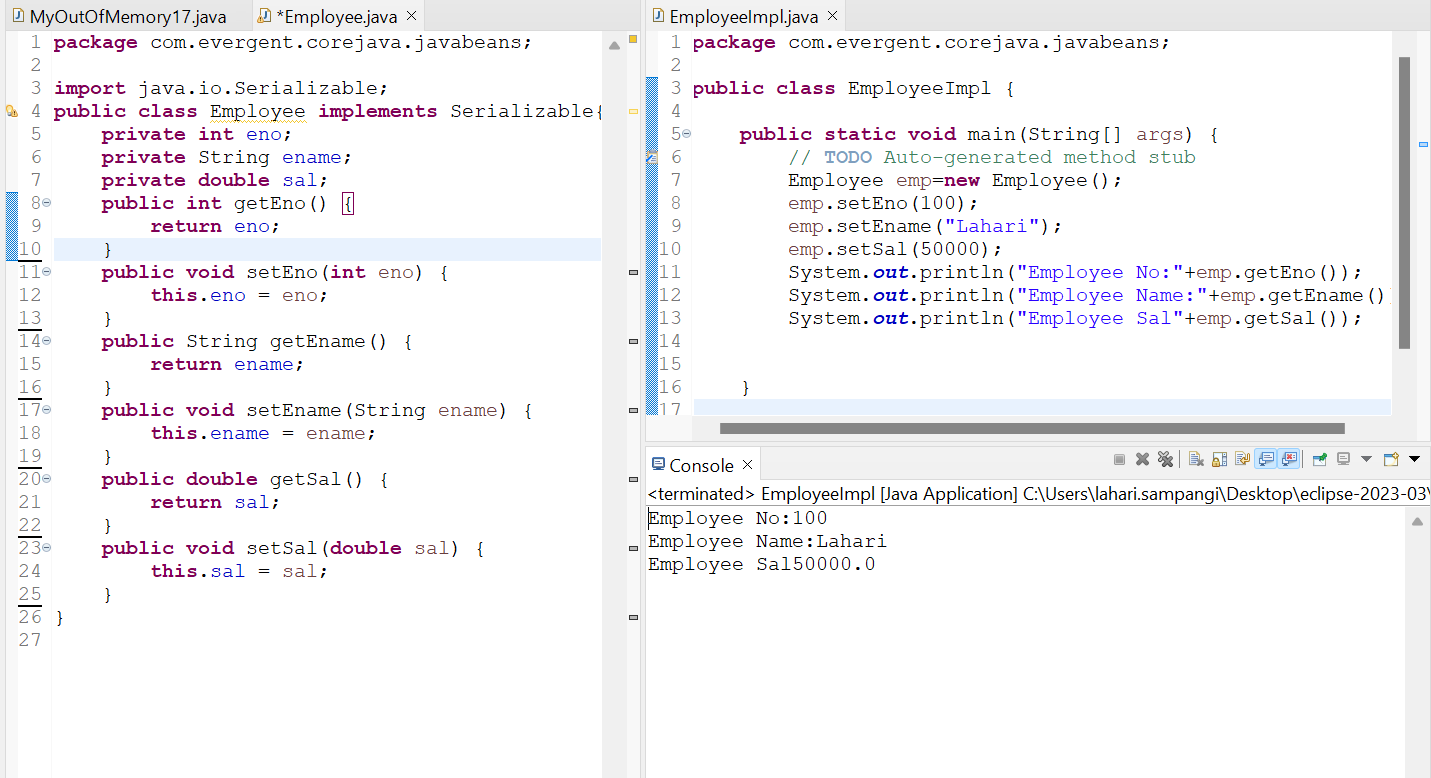
**Program-20: Nested Try-Catch**



**JAVA BEANS:**

1. Java bean is a mechanism.
2. Javabean is a light weight,all attributes are private and get/set(getter and setter methods) are public implements java.io.Serializable.

**Program-1:**



**Program-2:**



**Program-3:**

