**CORE JAVA DEVELOPMENT**

**Date:05/08/2024**

**Day 1**

1.Language and Applications

2.Java Features

* Why java is platform independent?
* OOPS
* Exceptional Handling
* Multi threading
* Security
* Open source
* Networking
* Memory Management
* Web Application

3.JDK,JRE,JVM

4.Basic Java Programming

* Variable
* Methods
* Conditional Statements
* Loops

5.Packages

**Date:06-08-2024**

**Day 2**

1.Loops

2.Single Dimensional Arrays

3.Two Dimensional Arrays

4.Logical Programs

* Fibnocci
* Factorial

5.Switch Case

6.Enum

7.Scanner Case

8.Object class and methods

9.Packages

10.Event Management System(EMS) Application overview

**Date:07-08-2024**

**Day 3**

* **OOPS CONCEPTS**

1. Encapsulation

- Calculation program

- Person class program

- MethodsFlow

2. About System.out.println();

3. Inheritance

4. Polymorphism

- Overloading

- Overriding

5.Abstract

6. IS-A Relationship(Inheritance)

7. HAS-A Relationship(object creation)

**Date:08-08-2024**

**Day 4**

**Constructor**

1.Class name and constructor name should be same.

2.There are two types of constructors

A.default constructor

B.parameterized constructor

3.we can access constructor while creation of objects.

4.constructors are mainly for initialising.

5.constructors does not have any return type ,not even void also.

6.If we Declare as a void the compiler will consider as a method not a constructor.

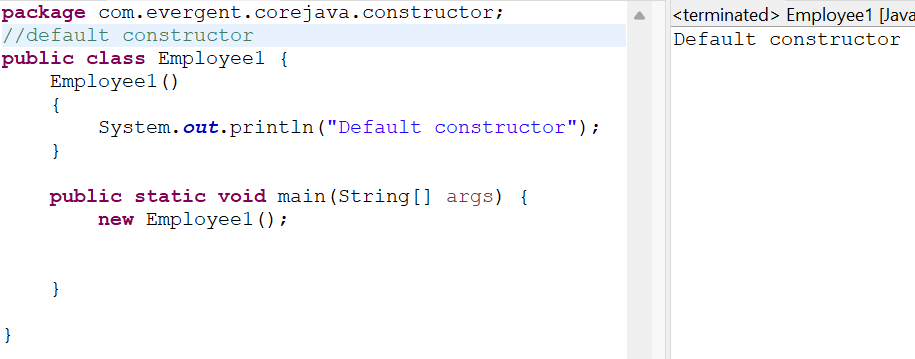
7.Every class needs atleast one default constructor.

-this

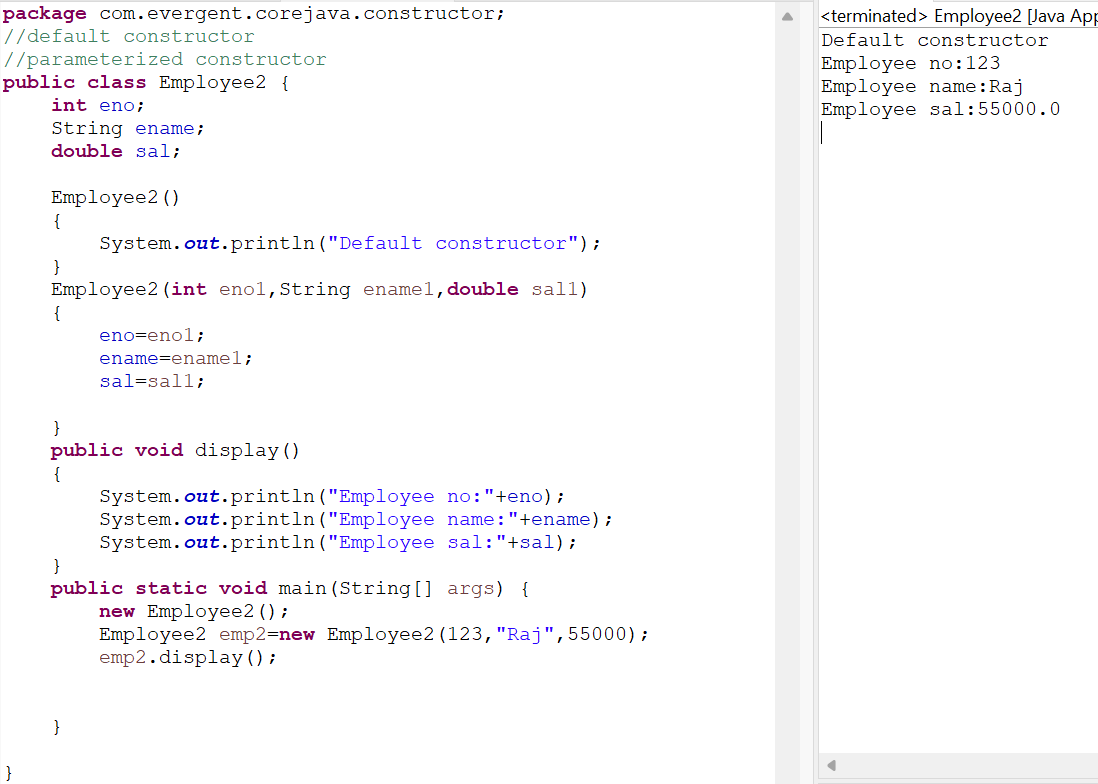
-super

8.constructors are overloaded.

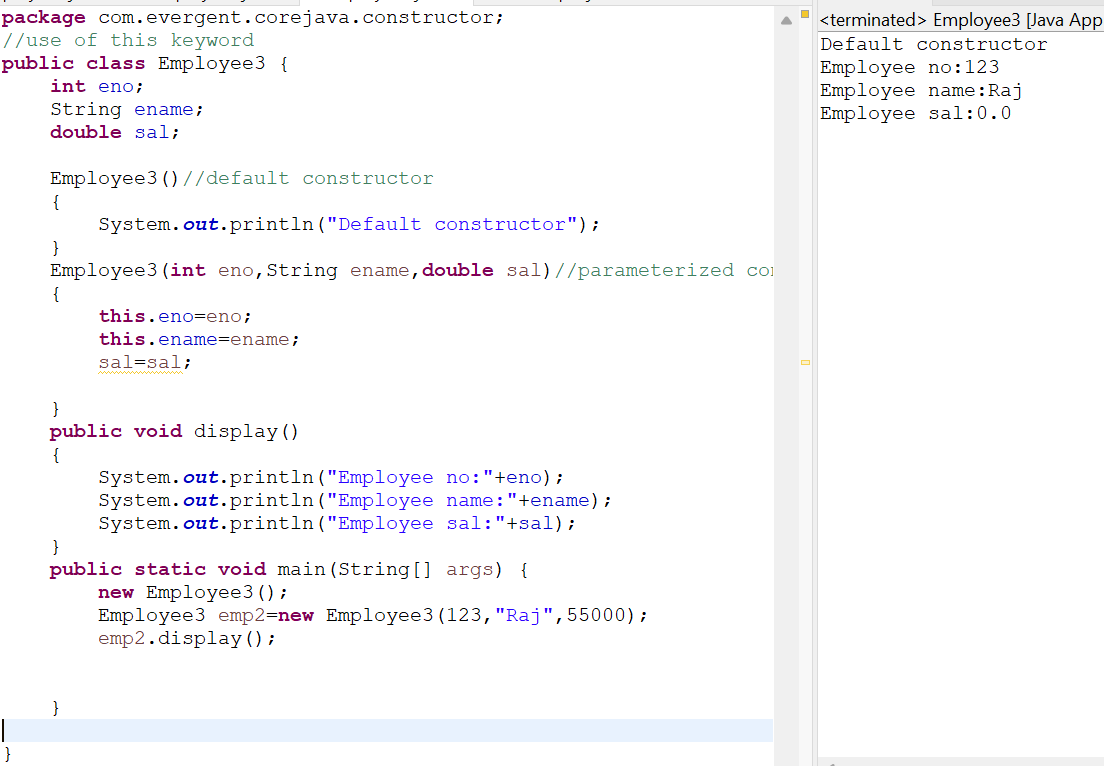
**Program1:**

****

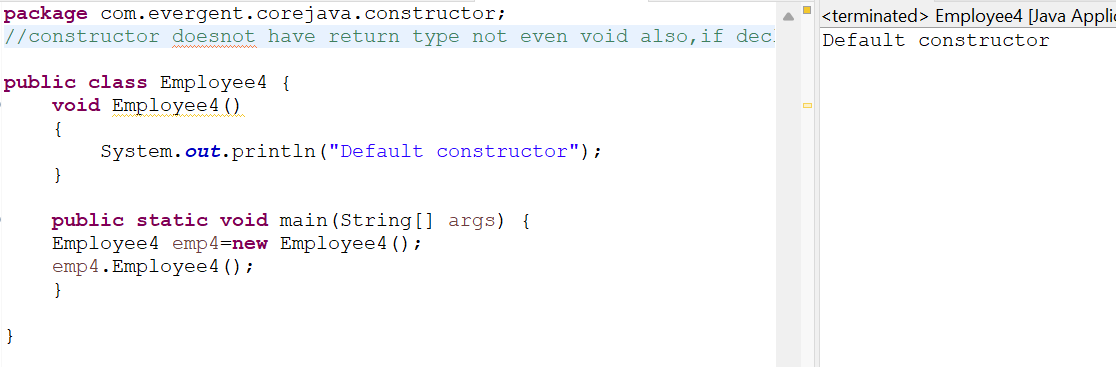
**Program2:**

****

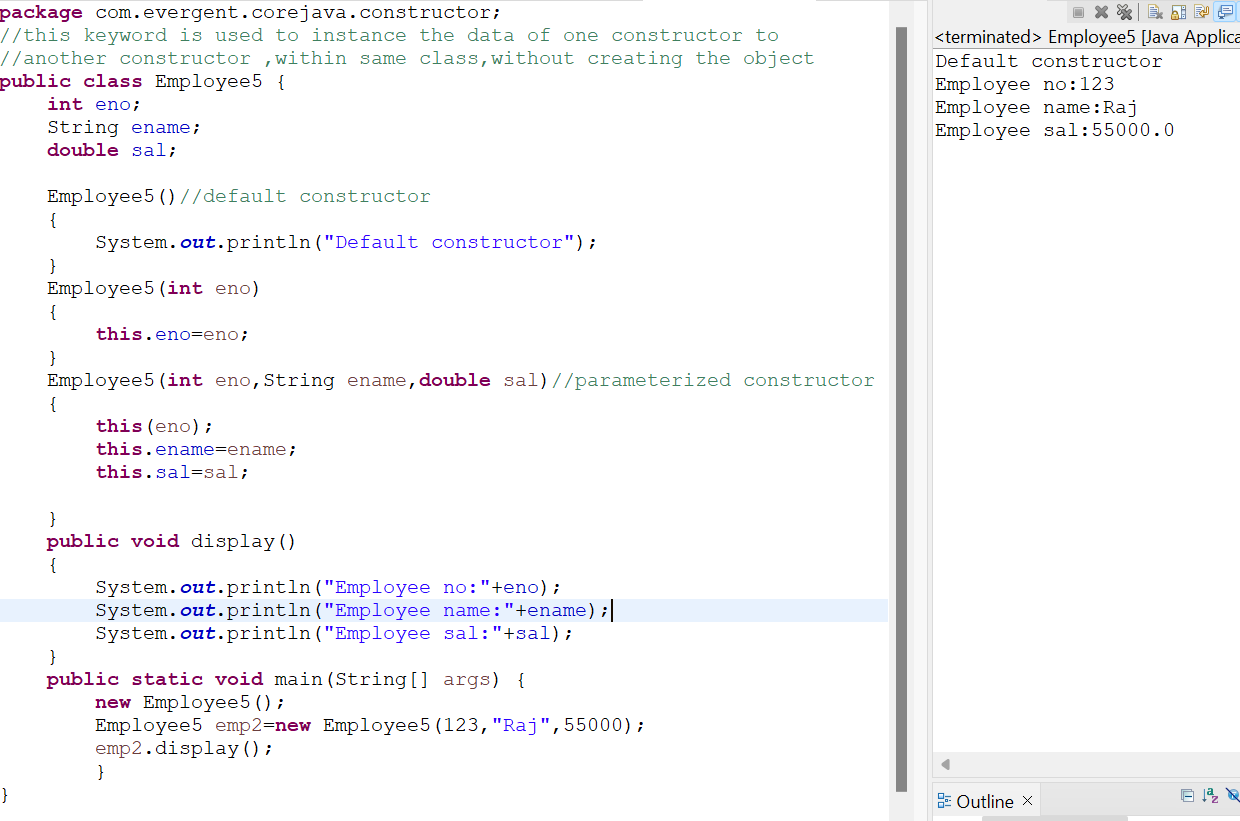
**Program 3:**

****

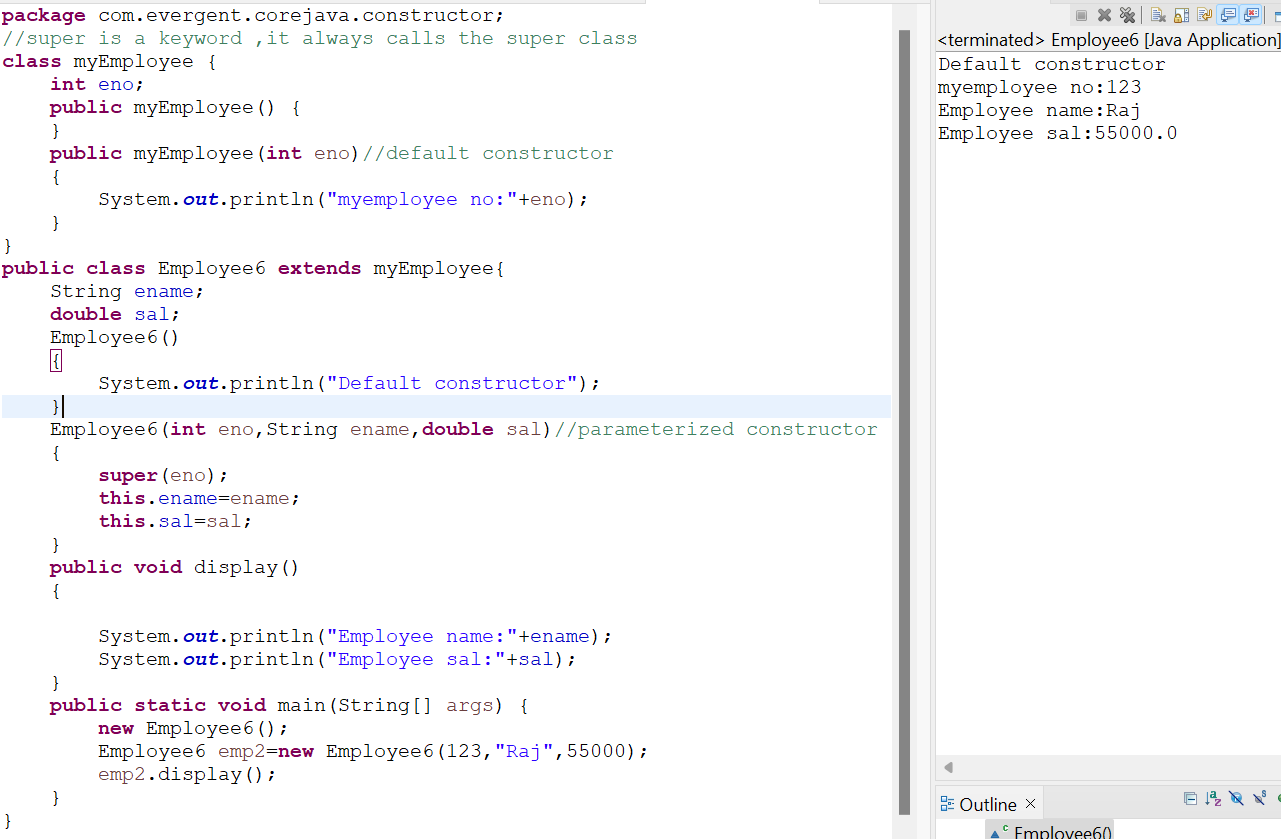
**Program 4:**

****

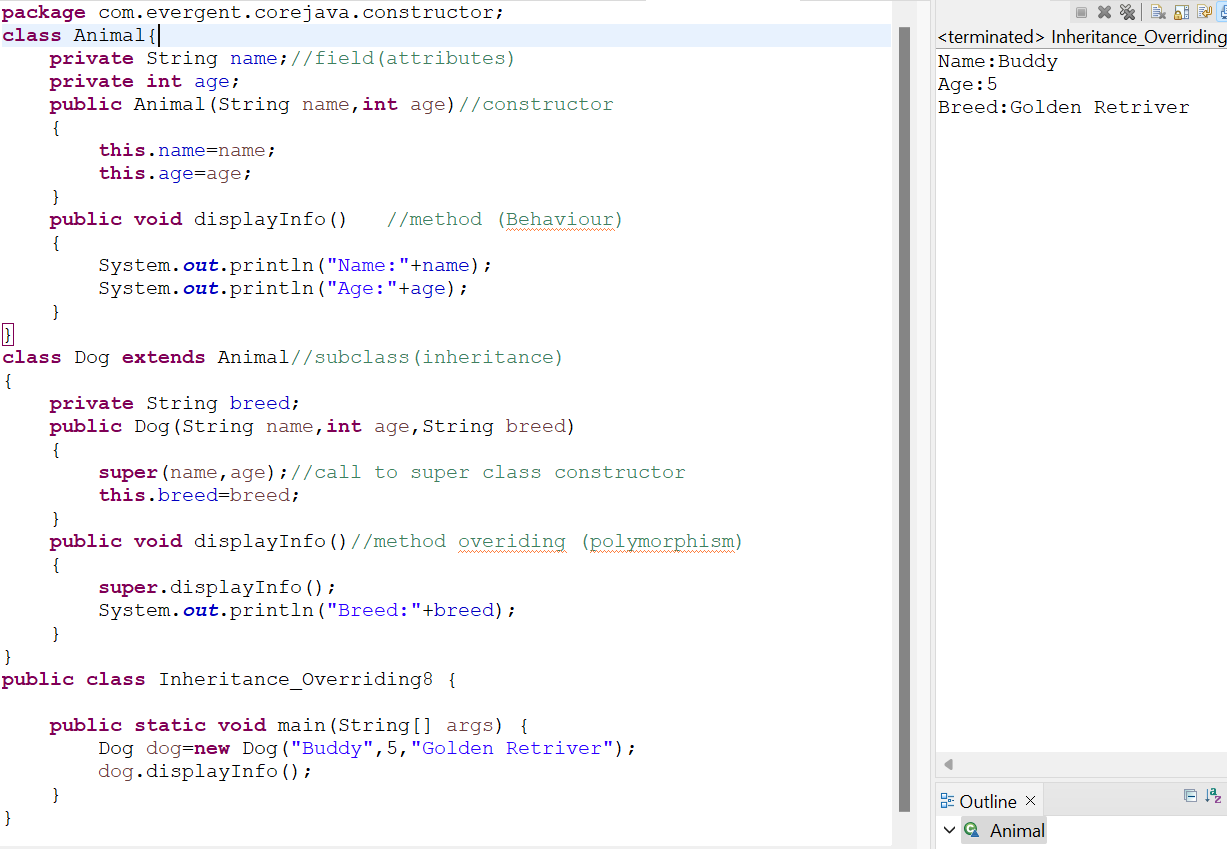
**Program 5:**

****

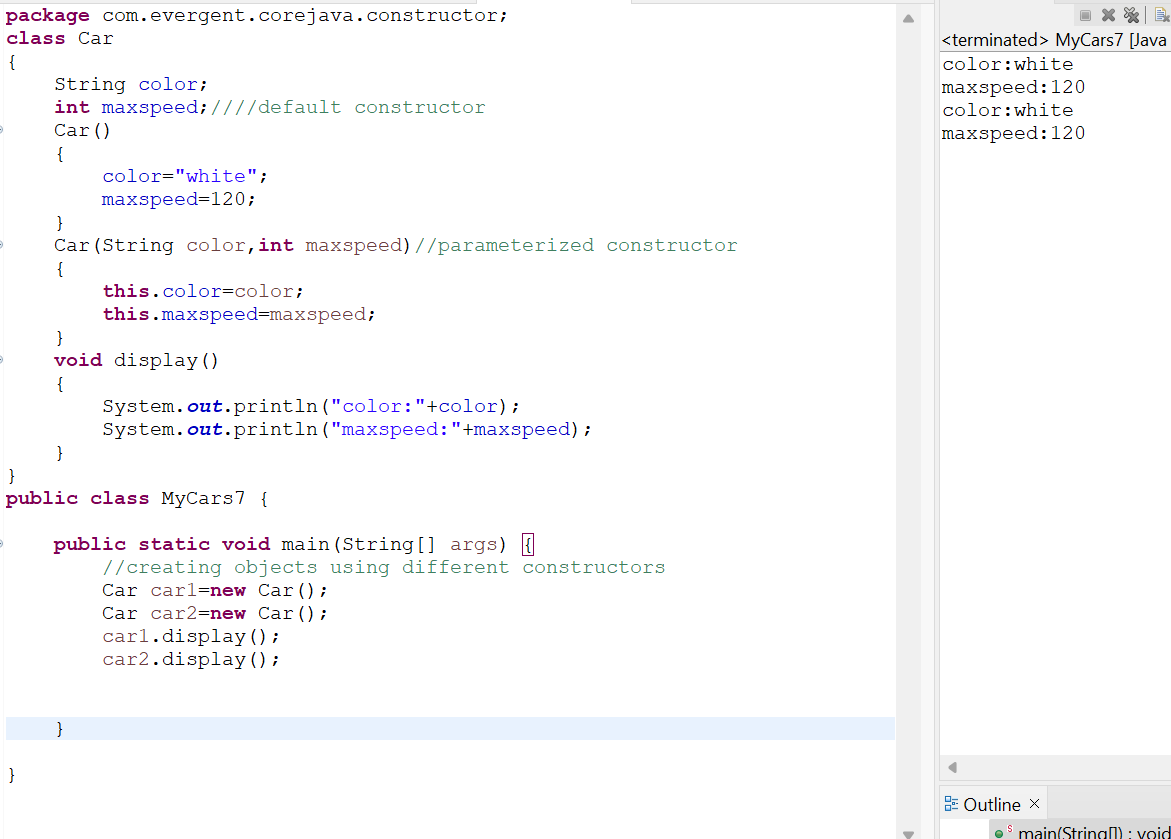
**Program 6:**

****

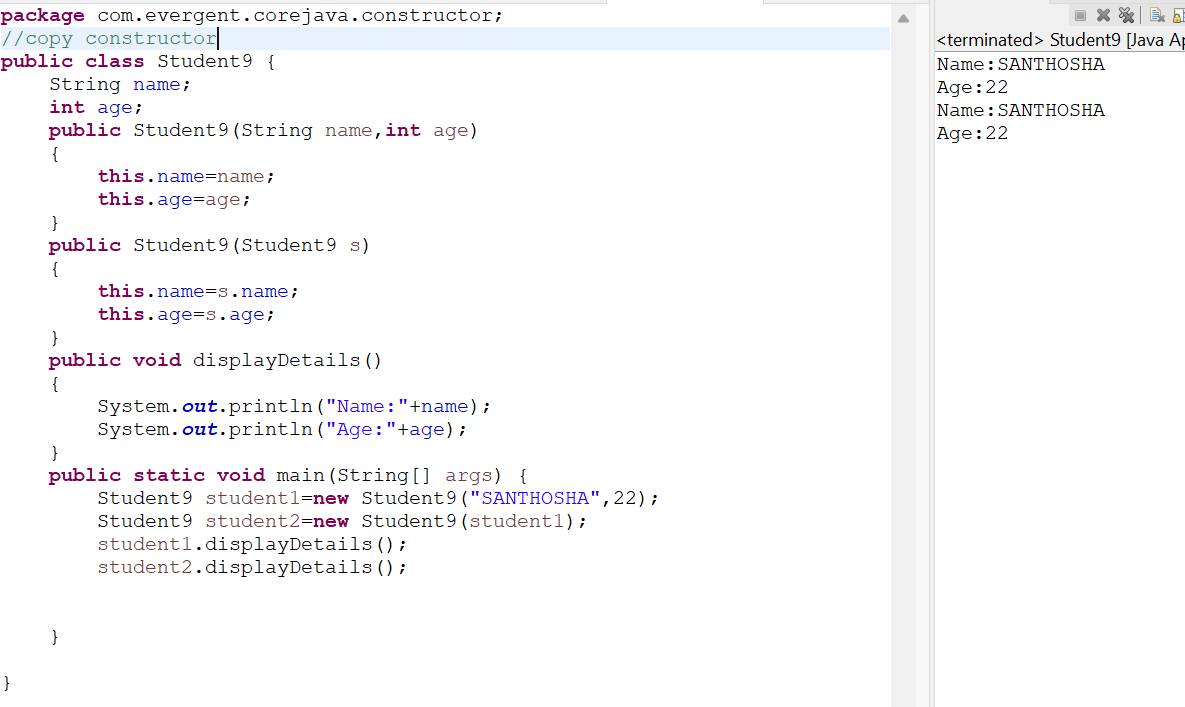
**Program 7:**

****

**Program 8:**

****

**Program 9:**

****

**Date:09-08-2024**

**Day 5**

**1.Static**

1.Static is keyword.

2.we can declare static as variables and methods.

3.we can access static variables and methods direct through classname.methodname and classname.variablename.

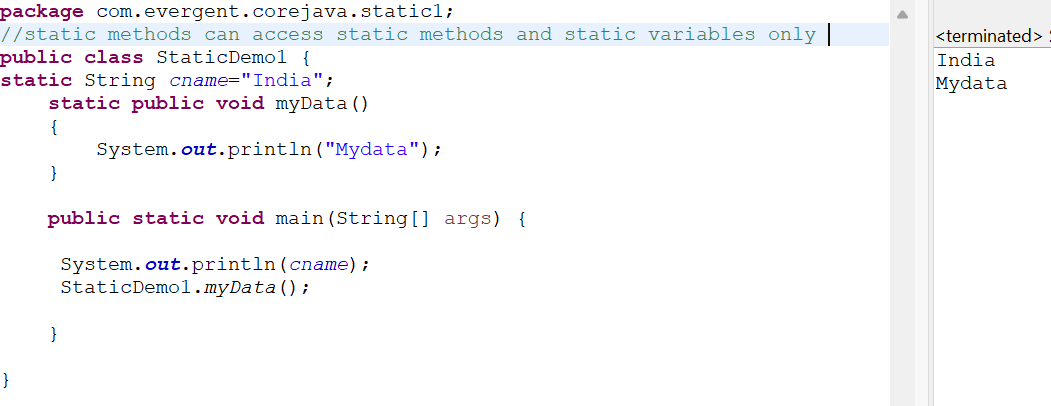
4.static methods can access static methods and static variables only.

5.static methods can not access non static methods and non static variables.

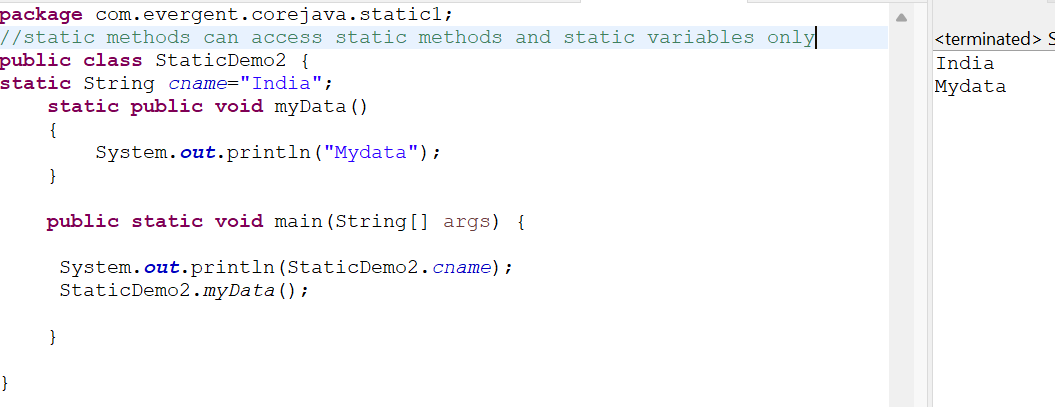
6.Non static methods can access static methods and static variables.

7.Static block-whenever class is loaded inside the JVM at that time static block is initiated.

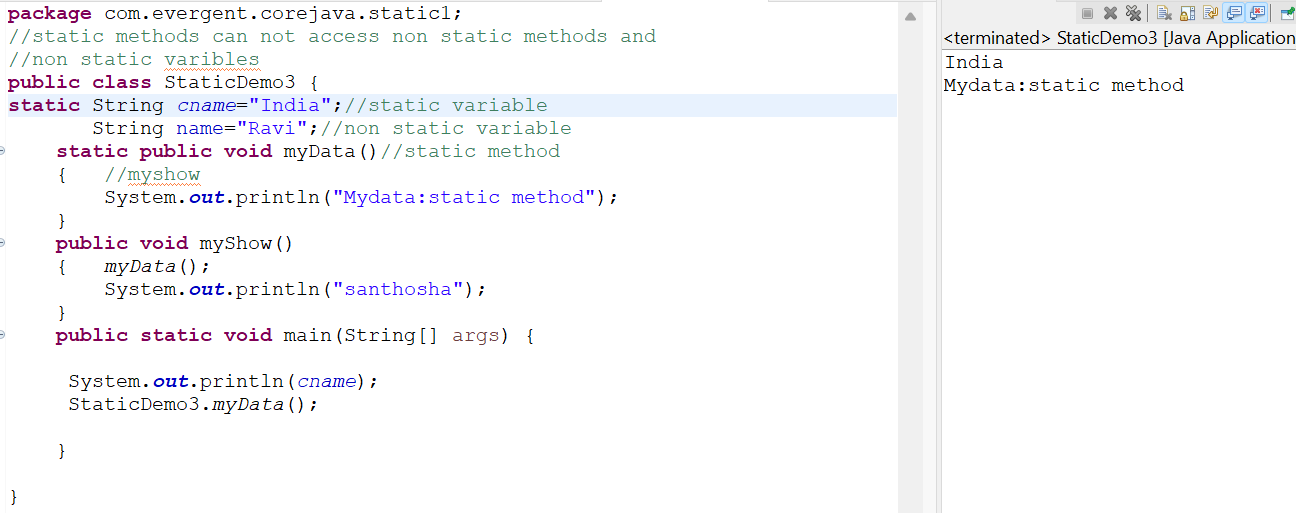
**Program 1:**

****

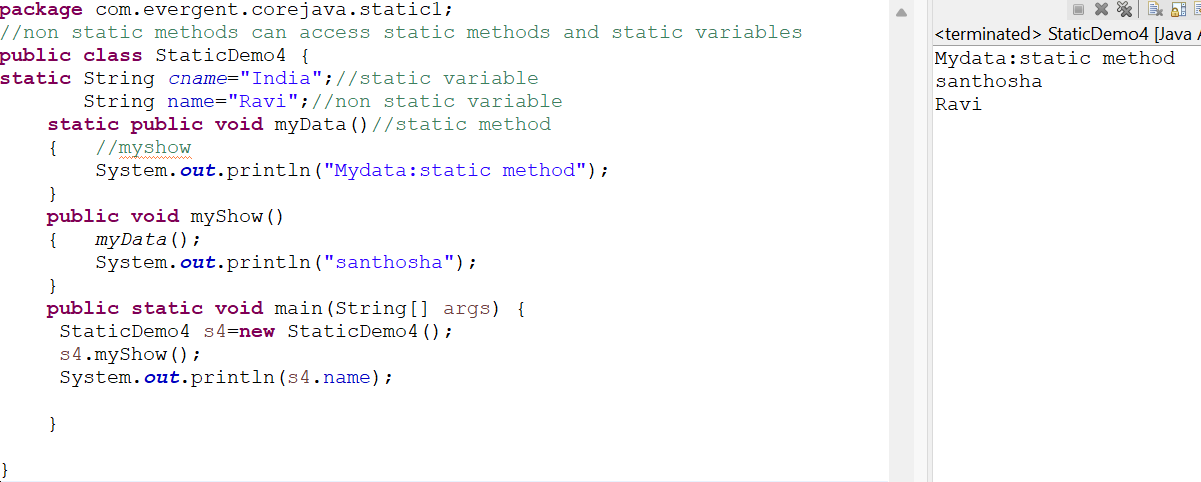
**Program 2:**

****

**Program 3:**

****

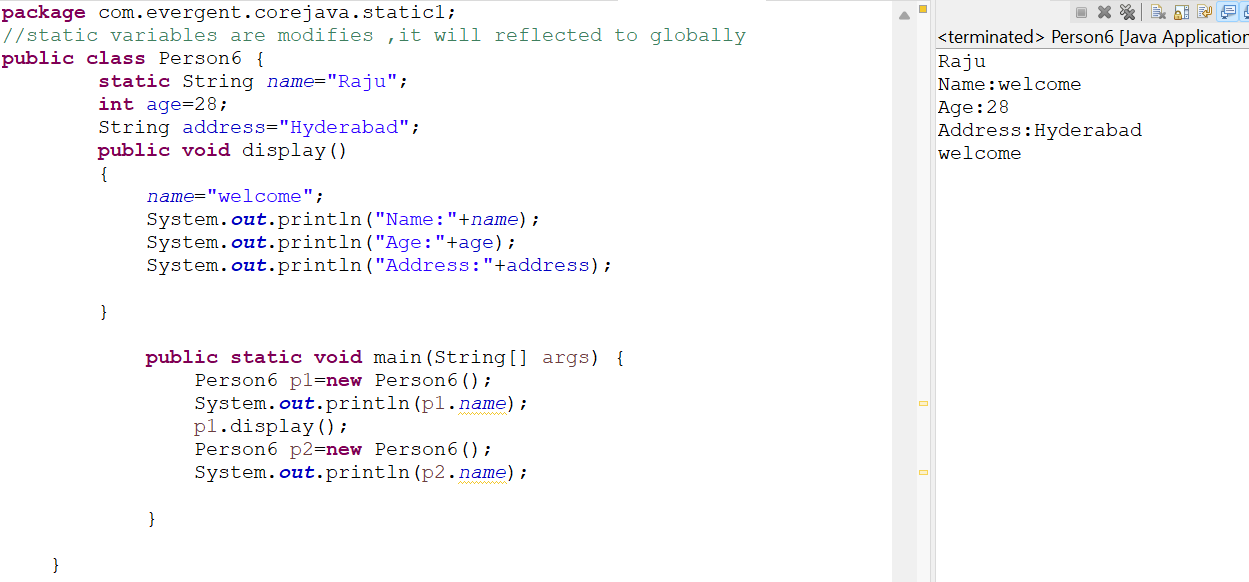
**Program 4:**

****

**Program 5:**

****

**Program 6:**

****

**2.Final:**

1.final is a keyword.

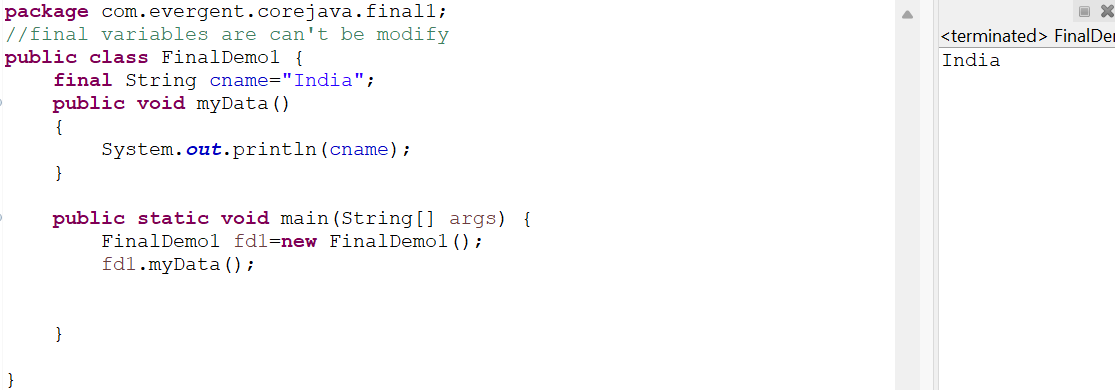
2.we can declare final as variables,methods,class.

3.final variables we can’t modify.

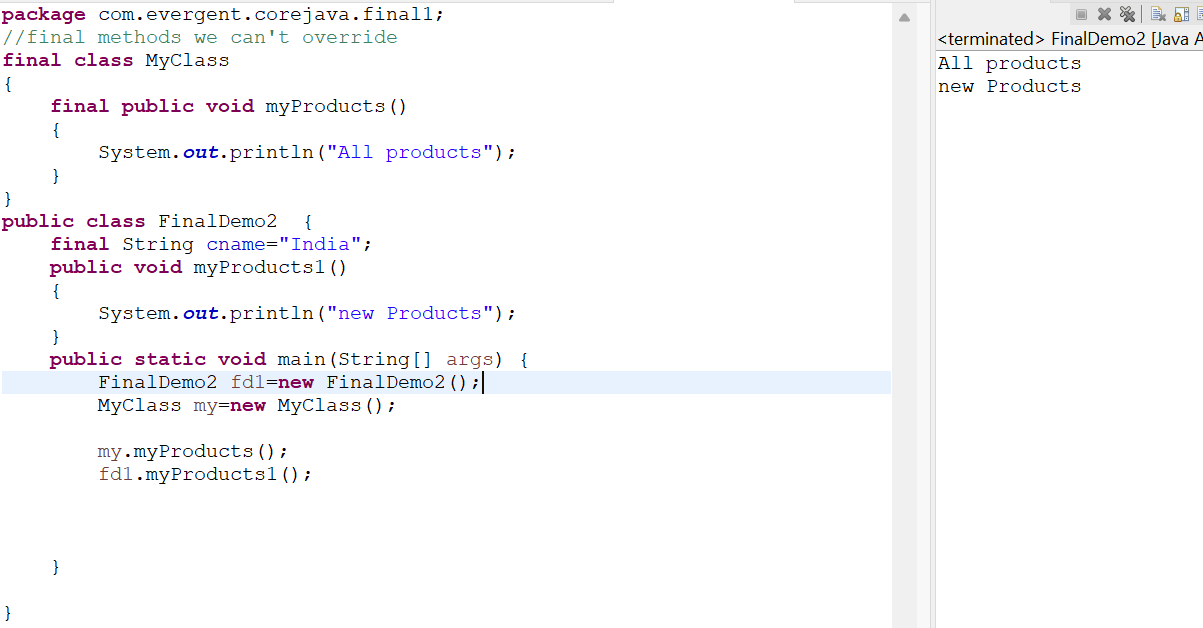
4.final methods we can’t override.

5.final class we can’t inherited(extends).

**Program 1:**

****

**Program 2:**

****

**Program 3:**

****

**Date:12-08-2024**

**Day 6**

**Strings:**

1.String class

2.string buffer

3.string builder

**A.why strings are immutable?**

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

**1.String class**

a. String is a final class

b. Strings are immutable

c. Strings having methods

d. All methods are non-synchronized

**2.StringBuffer**

a. String buffer is a final class

b. String buffer is mutable

c. String buffer having methods

d. All methods are synchronized

**3.StringBuilder(jdk 1.5)**

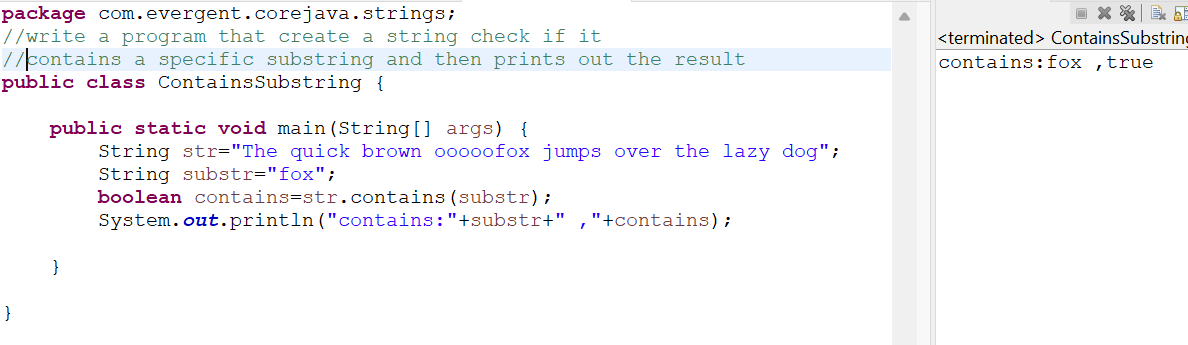
a. String buffer is a final class

b. String buffer is mutable

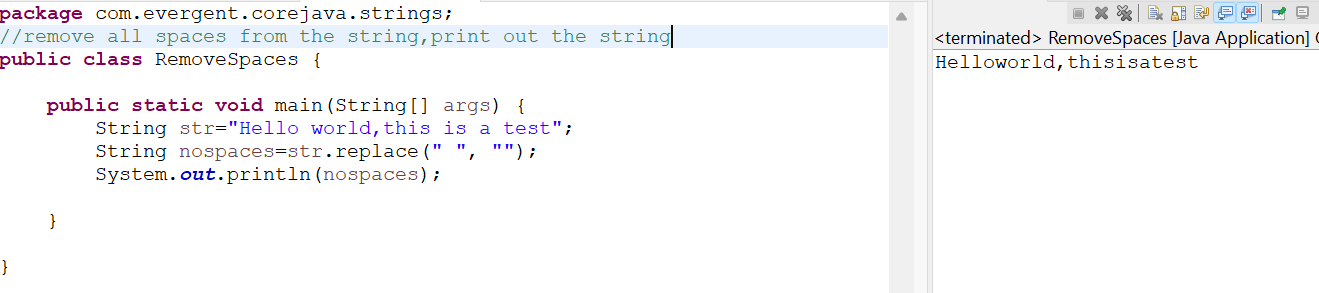
c. String buffer having methods

d. All methods are synchronized

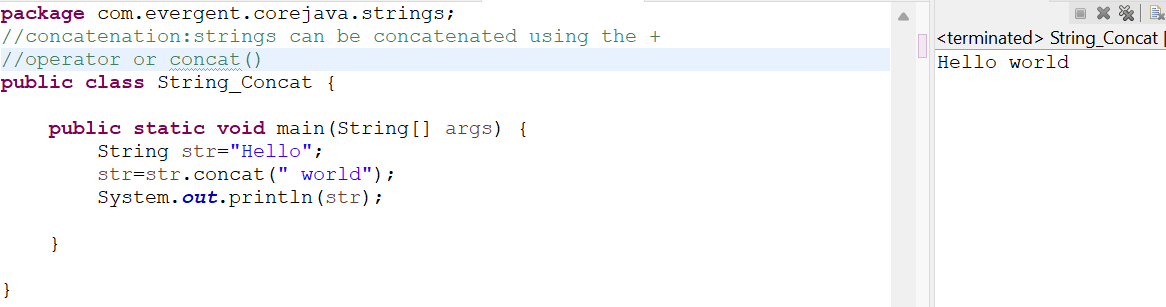
**Program1:**

****

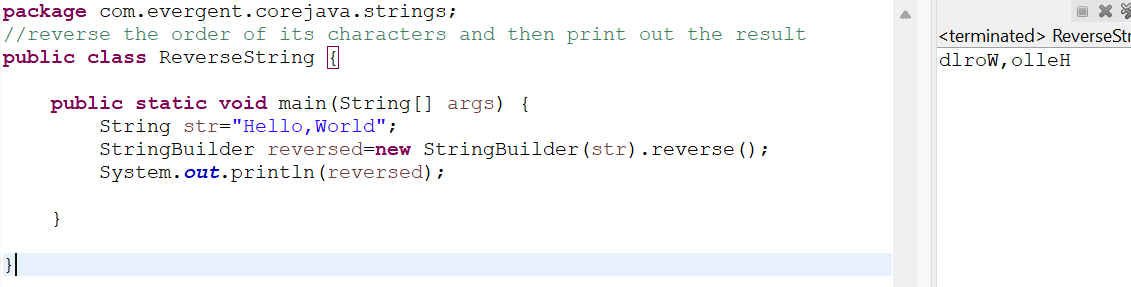
**Program 2:**

****

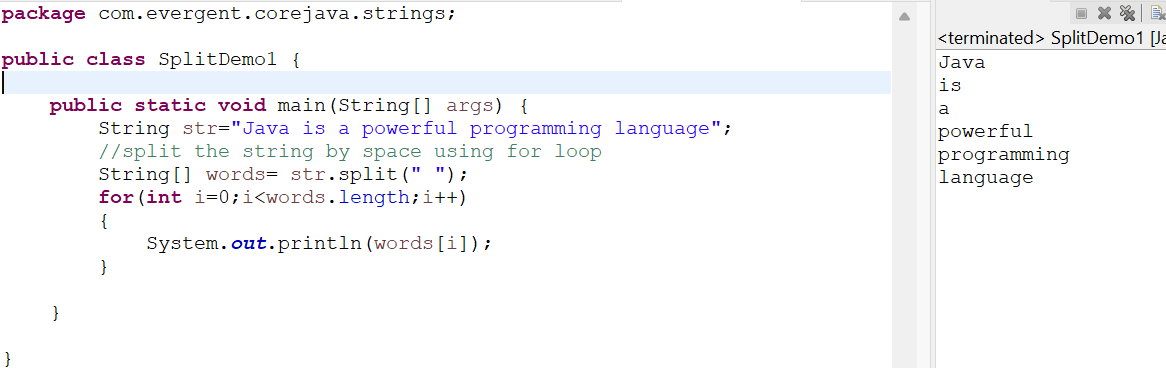
**Program 3:**

****

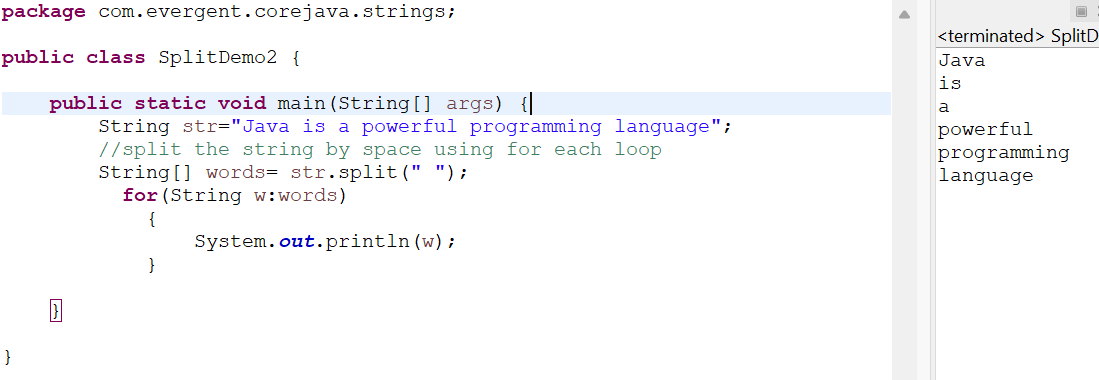
**Program 4:**

****

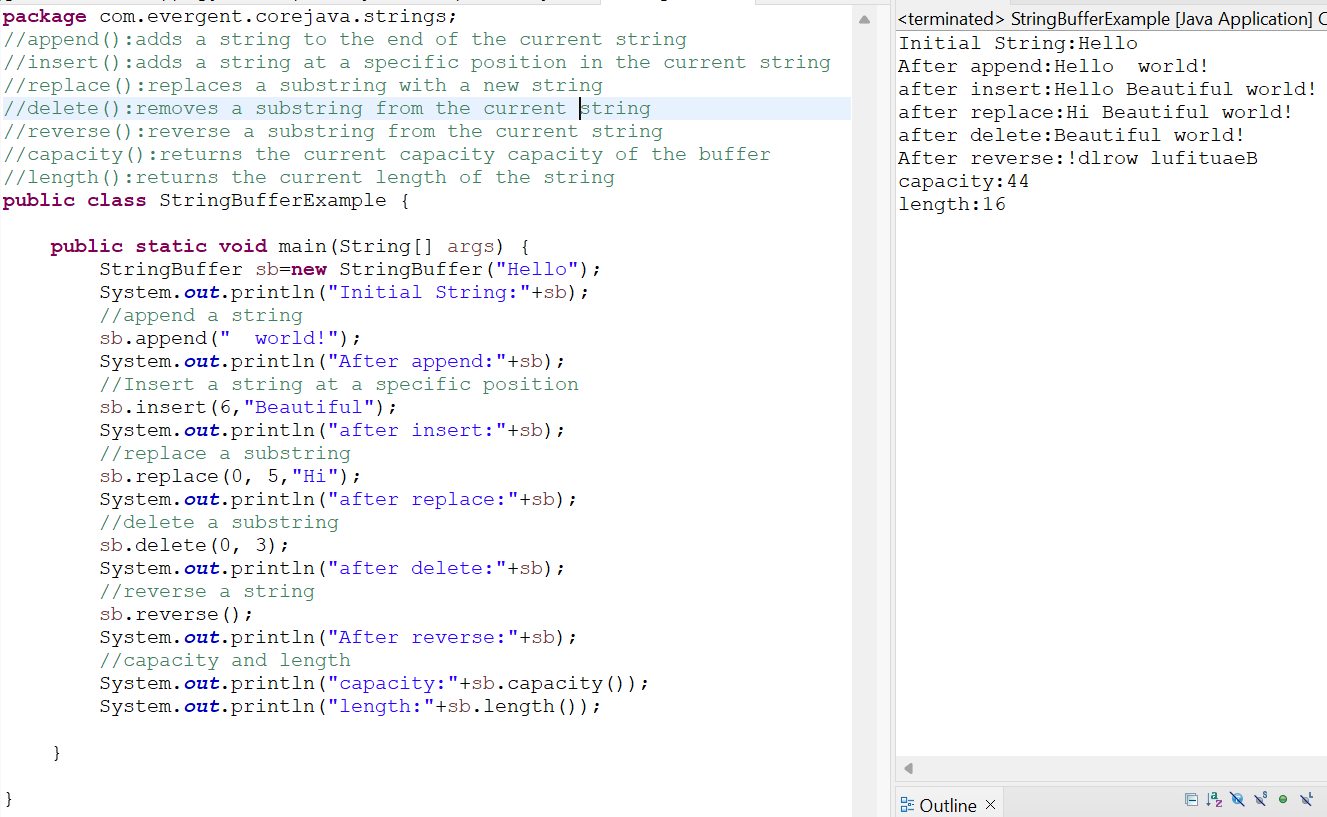
**Program 5:**

****

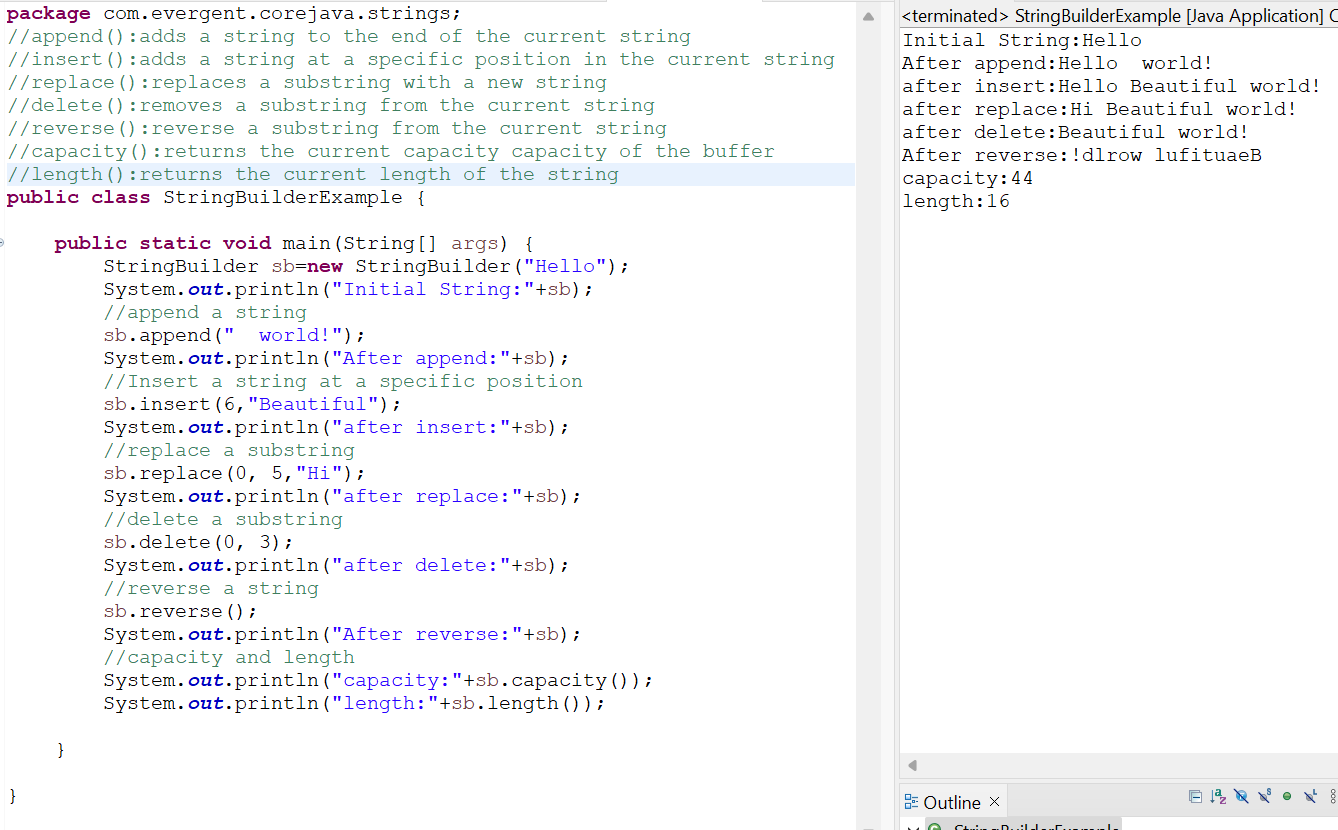
**Program 6:**

****

**Program 7:**



**Program 8:**

****

**Difference between string ,stringbuffer,stringbuilder**

**String:**   **stringbuffer**  **stringbuilder**

A.immutable A.mutable A.mutable

B.non synchronize B.synchronize B.non synchronize

C.final class c.final class c.final class

D.string class having methods D.string class having methods D.string class methods

**String class imp points:**

1.In java, a string is a sequence of characters often used to represent text.

2.strings are objects in java and are instance 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 string object is creates , new string object.

4.Java optimizes memory usage by storing strings in a special area of memory known as the “String pool”.

5.”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”.

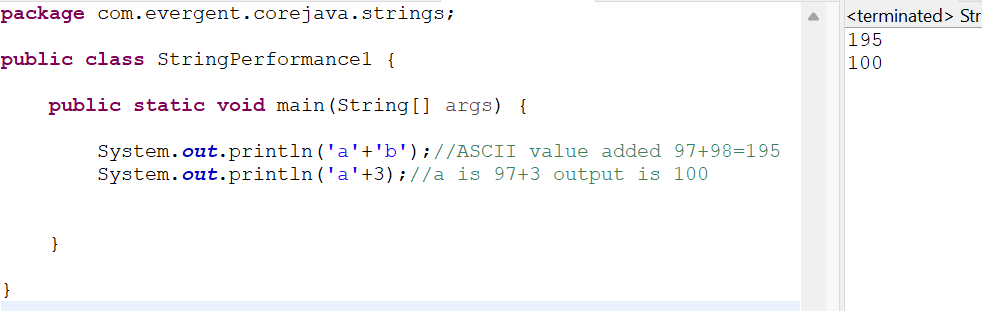
6.we can create a string in java in multiple ways:

A.using string literals:string str=”hello world”;

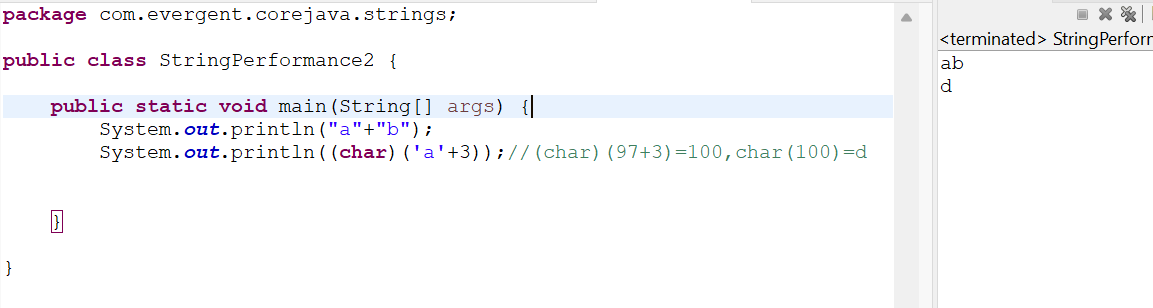
B.using the new keyword

String str=new String(“Hello world!”);

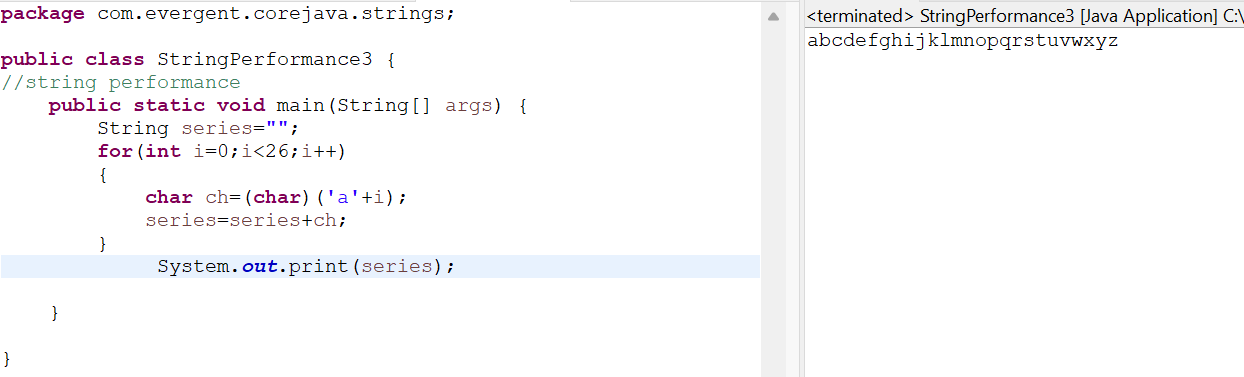
**Program 1:**

****

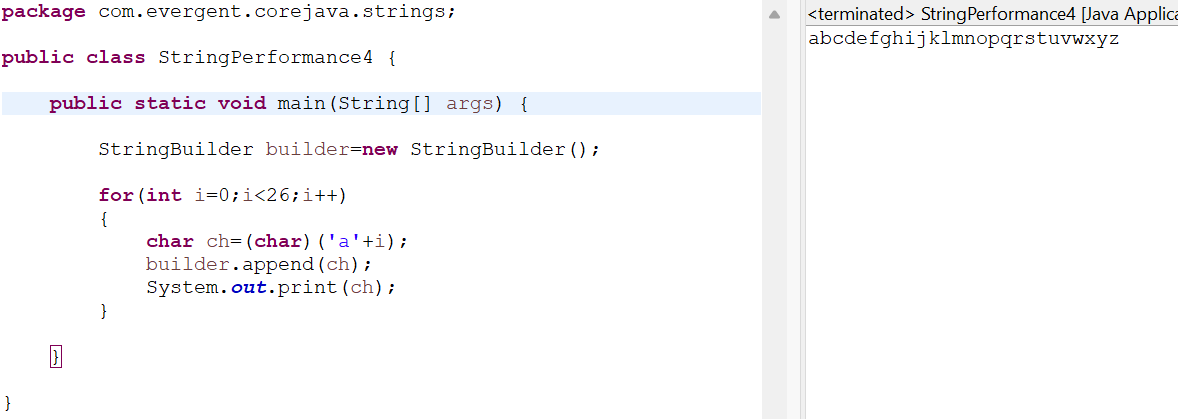
**Program 2:**

****

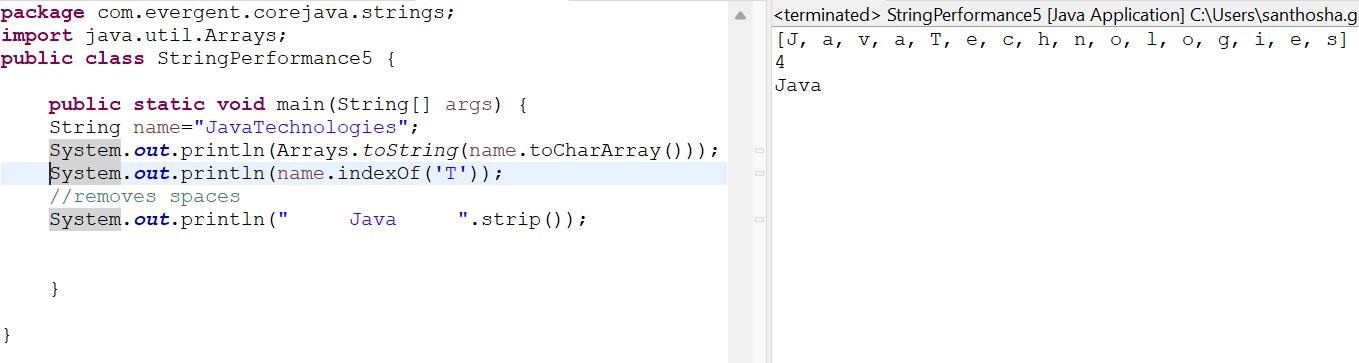
**Program 3:**

****

**Program 4:**

****

**Program 5:**

****

**Date:13-08-2024**

**Day 7**

Q.Can we made our class is immutable?

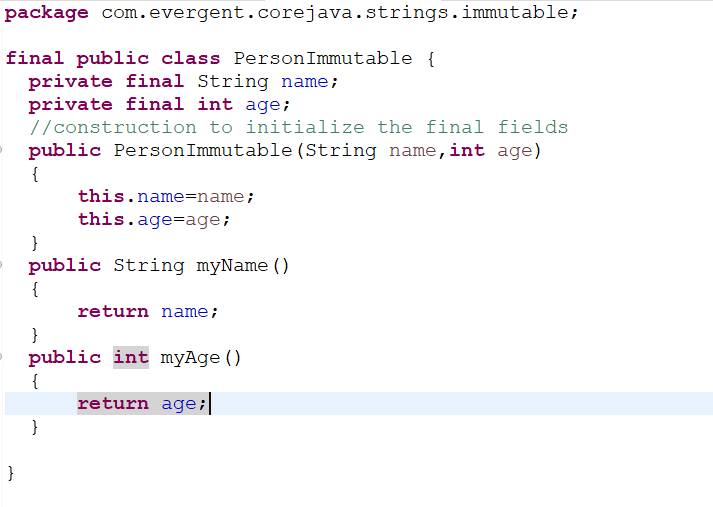
Sol:we can declare as the final class,and attributes of class private and finally.

We can create our own immutable class:

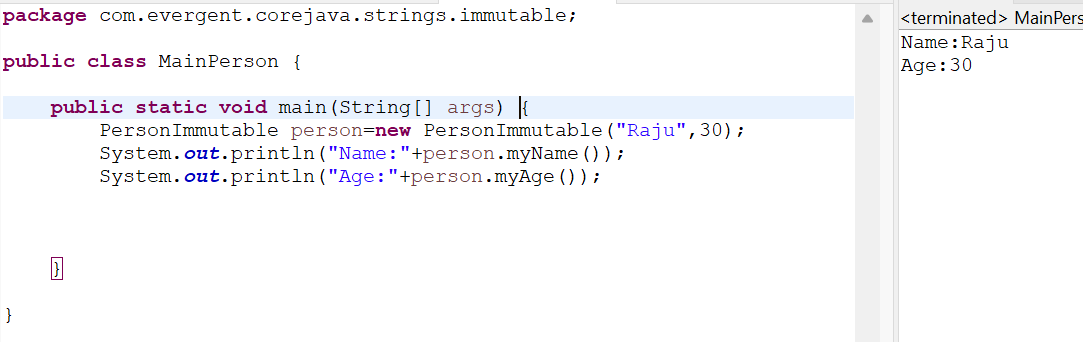
1.we can declare class as final.

2.The class is declared as final so that it cannot be subclassed.

**Program 1:**

****

**Main class:**

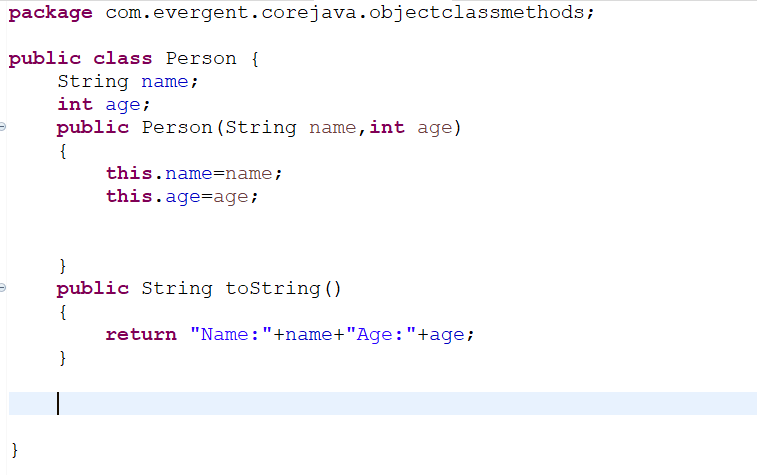
****

**Object class methods:**

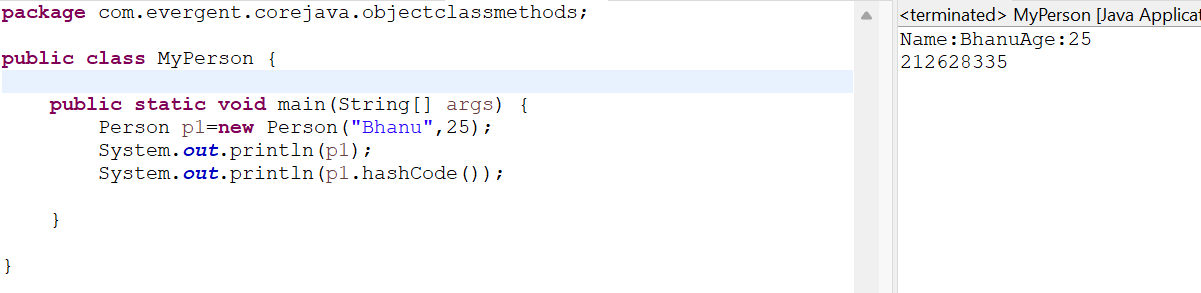
-toString()

-hashCode();

**Program2:**

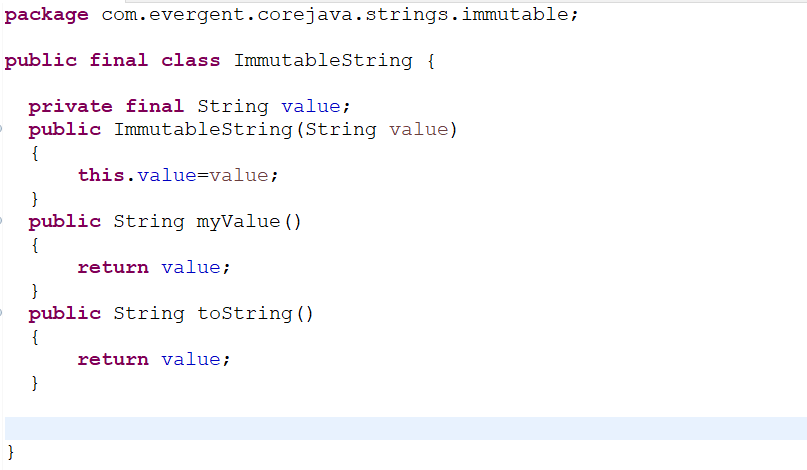
****

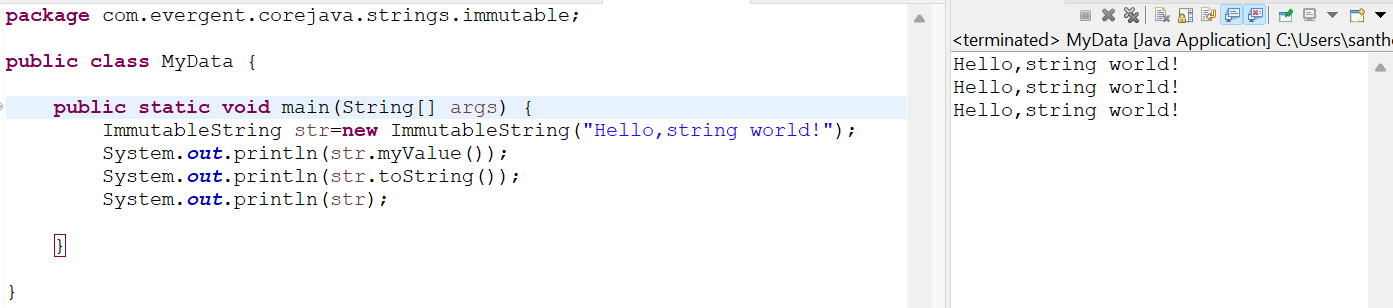
**Main class:**

****

**Program3:**

**Immutable String:**

****

****

**Interfaces:**

1. Interface is a keyword.

2. We can declare methods signature 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 can’t create object to interface. But we can create reference to interface.

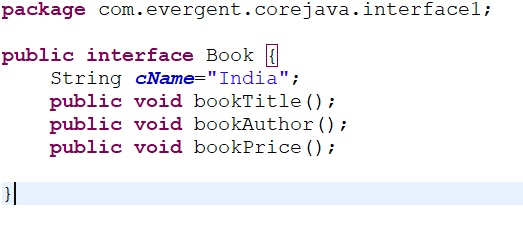
6. We can declare variables inside interface, all are public static final (default).

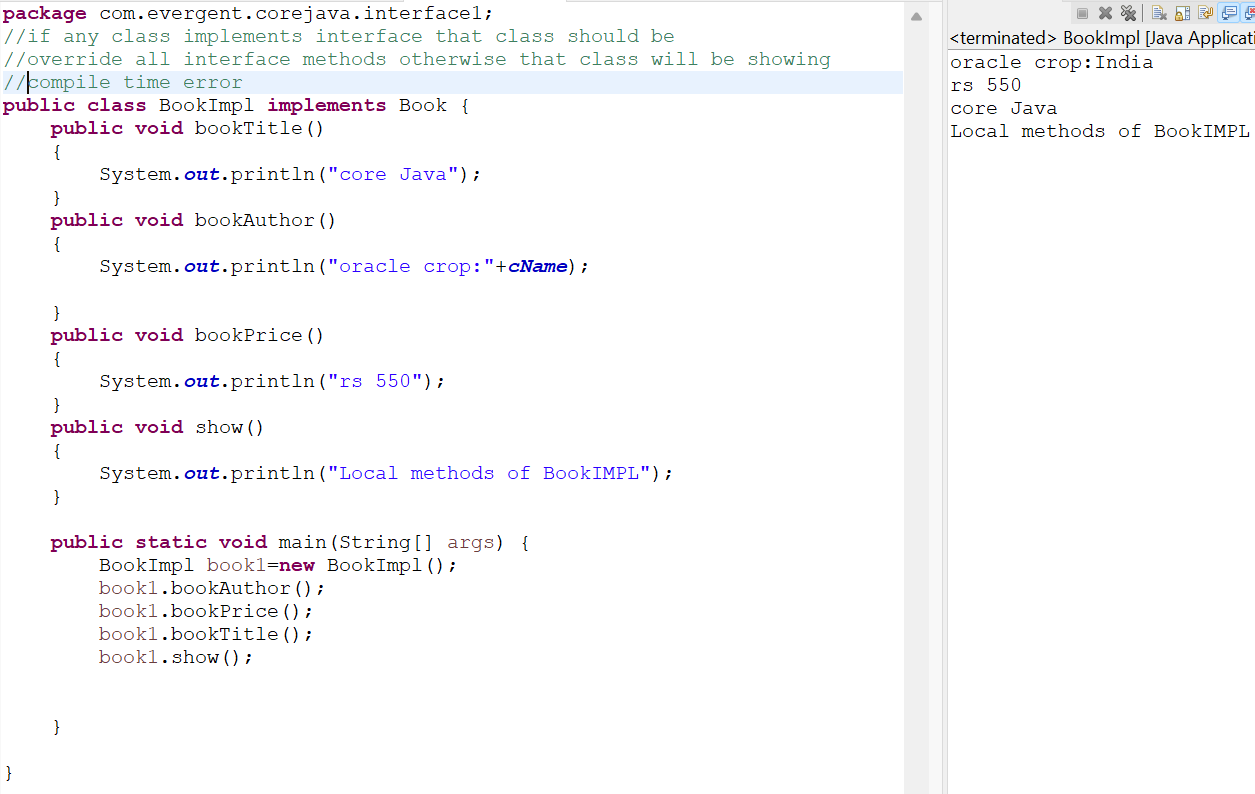
7. Java will support Multiple Inheritance through interfaces.

8. One class can implements Interface.

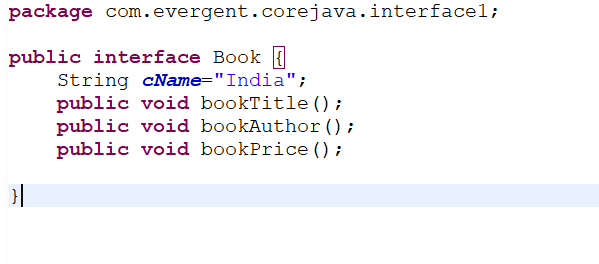
9. One interface can extends other interface.

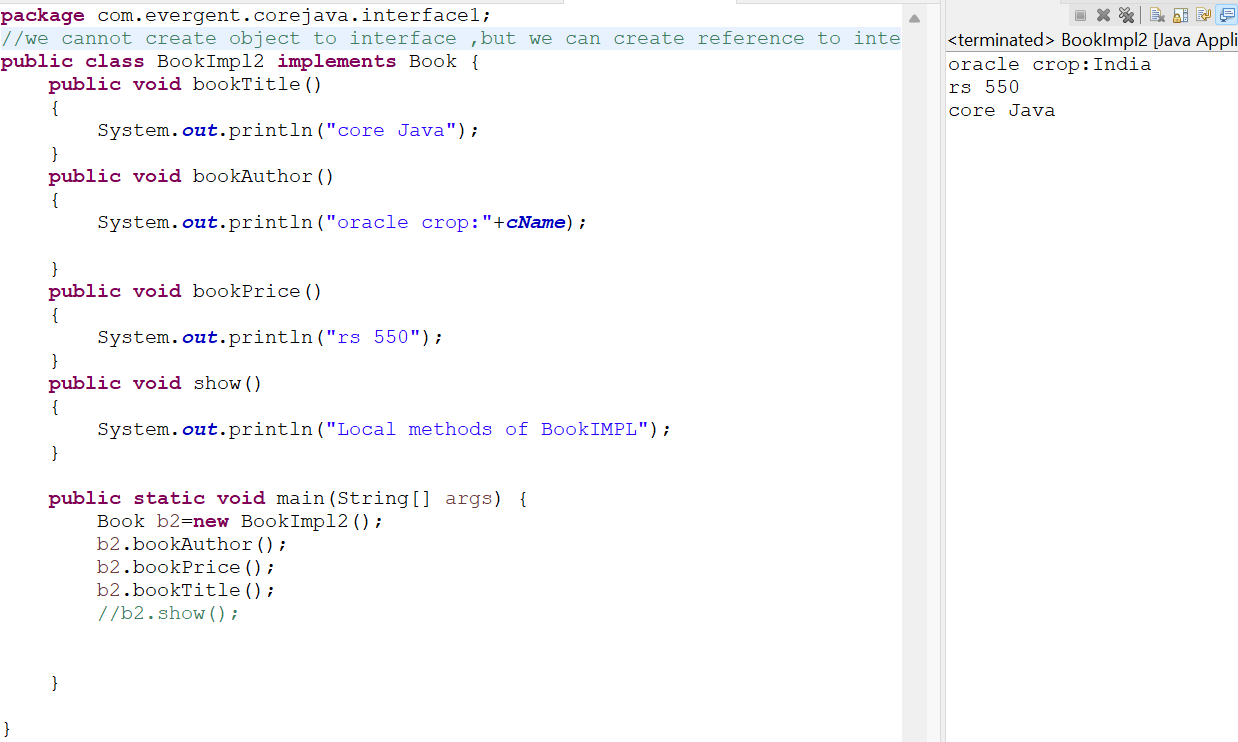
**Program1:Book interface**

****

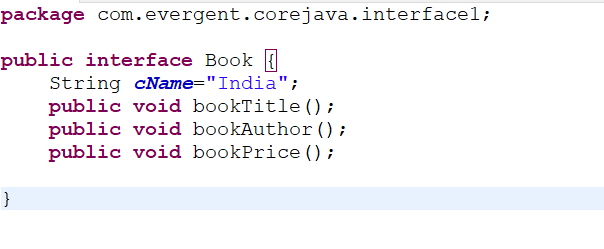


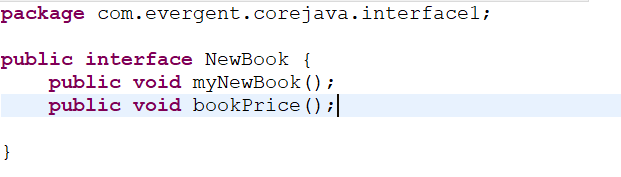
**Program2: Book Interface**

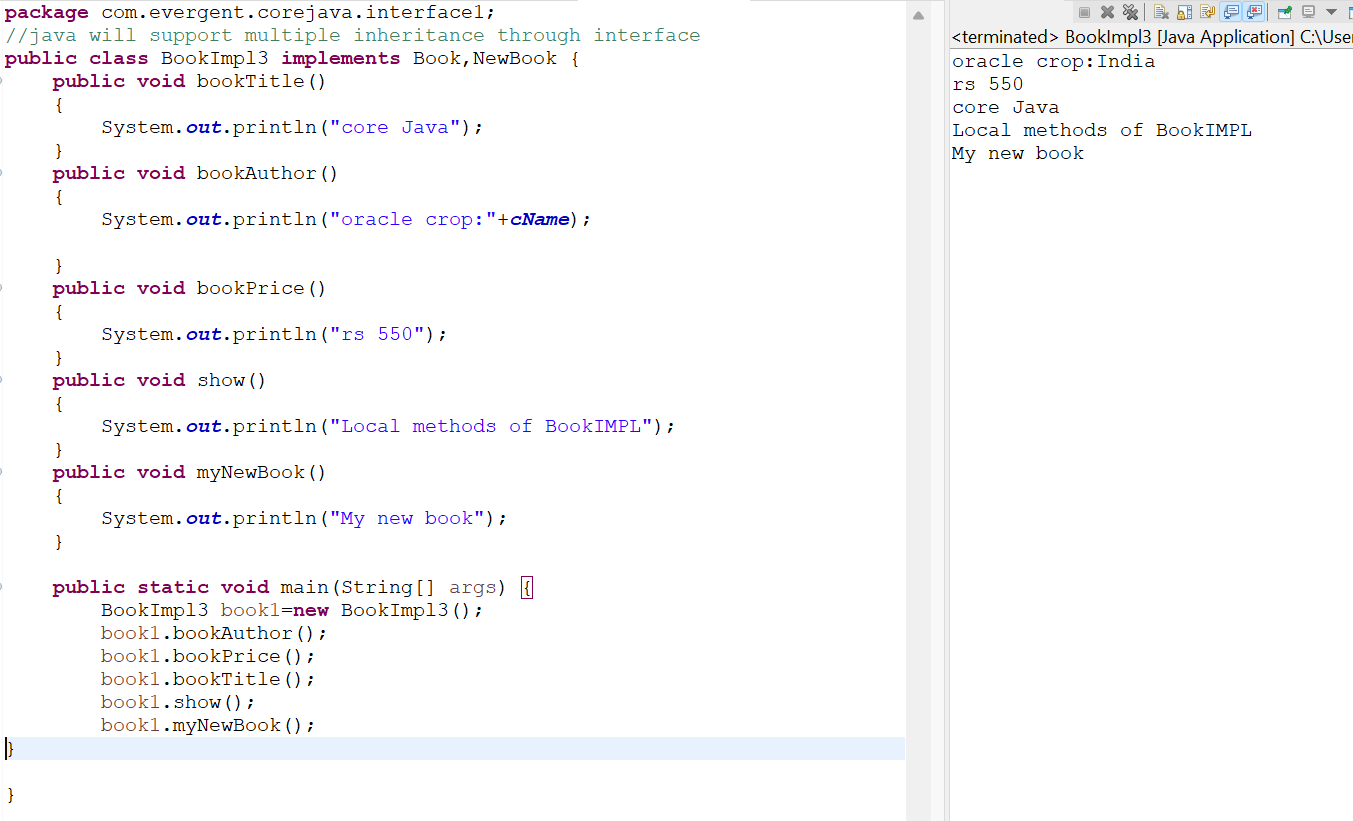
****

****

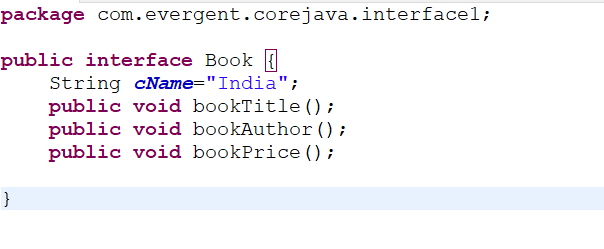
**Program3:**

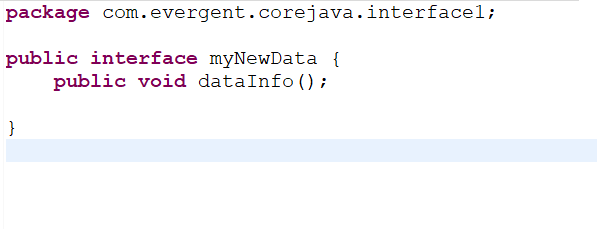
****

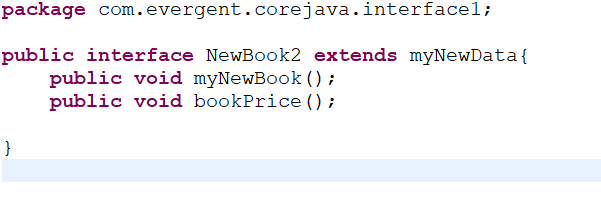
****

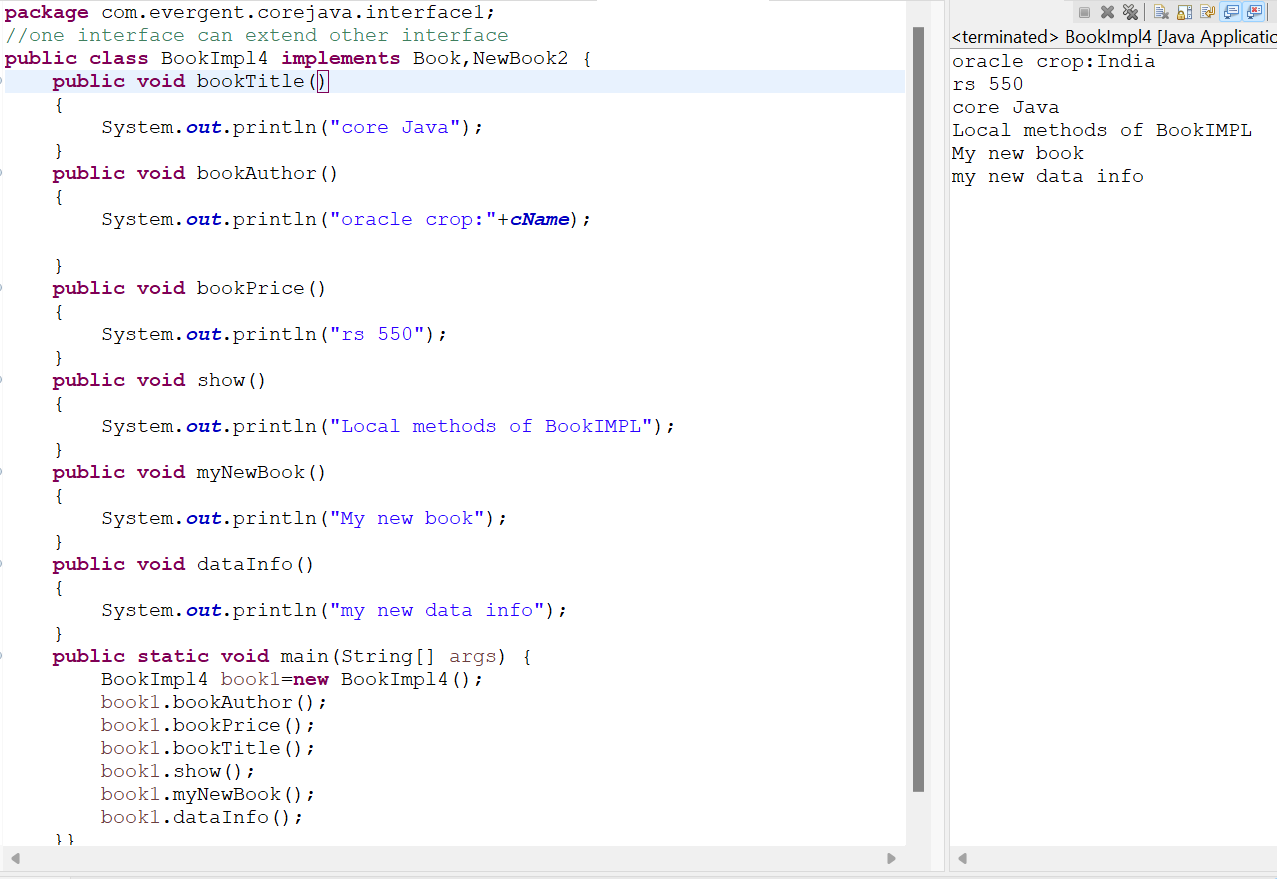
****

**Program 4:**

****

****

****

****

**Date:14/08/2024**

**Day 8**

**Abstract class:**

1.Abstract is a keyword.

2.Abstract class having abstract methods and concret(implemented).

3.If any class having one abstract method ,that class should be declare as abstract keyword,otherwise that class should be showing compile time error.

4.If any class extends abstract class that class should be override all abstract class otherwise that class showing compile time error.

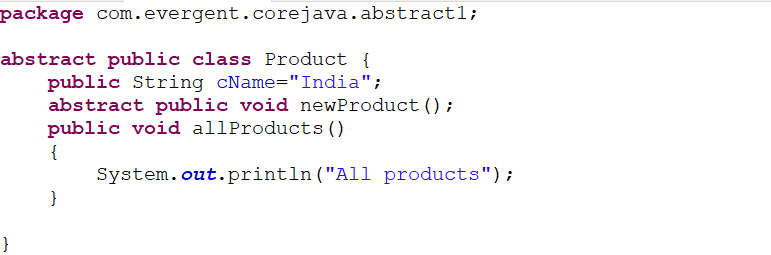
5.We can’t create object to abstract class,but we can create reference to abstract class.

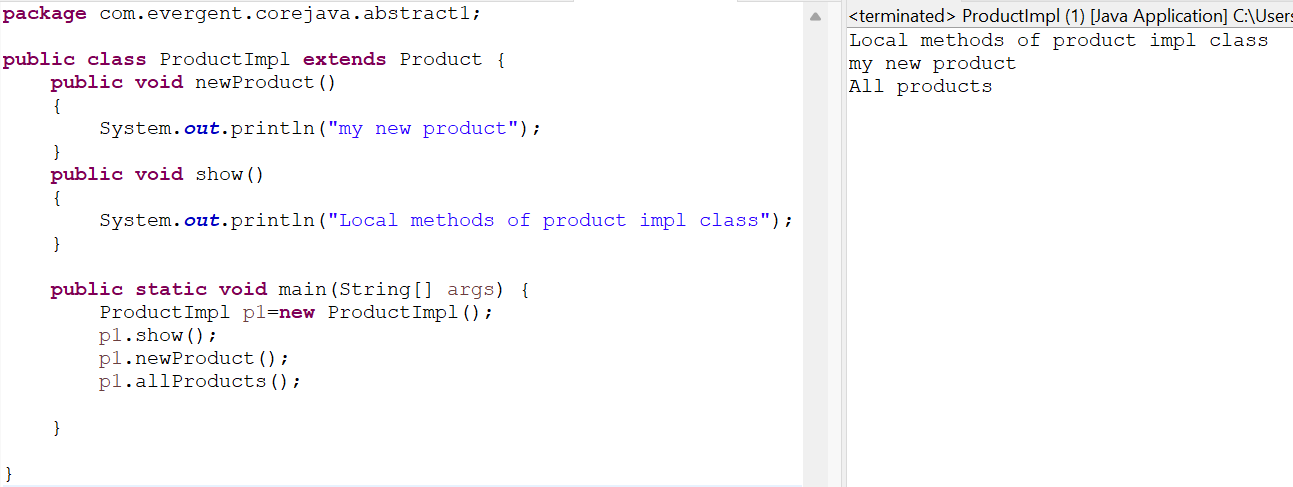
6.we can declare abstract class with zero abstract methods.

7.we can create constructor to abstract class.

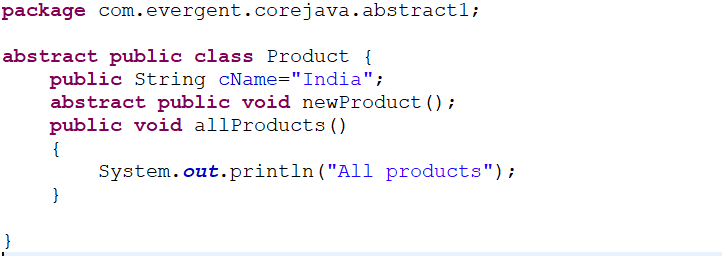
8.we can access abstract class constructor through sub class object creation.

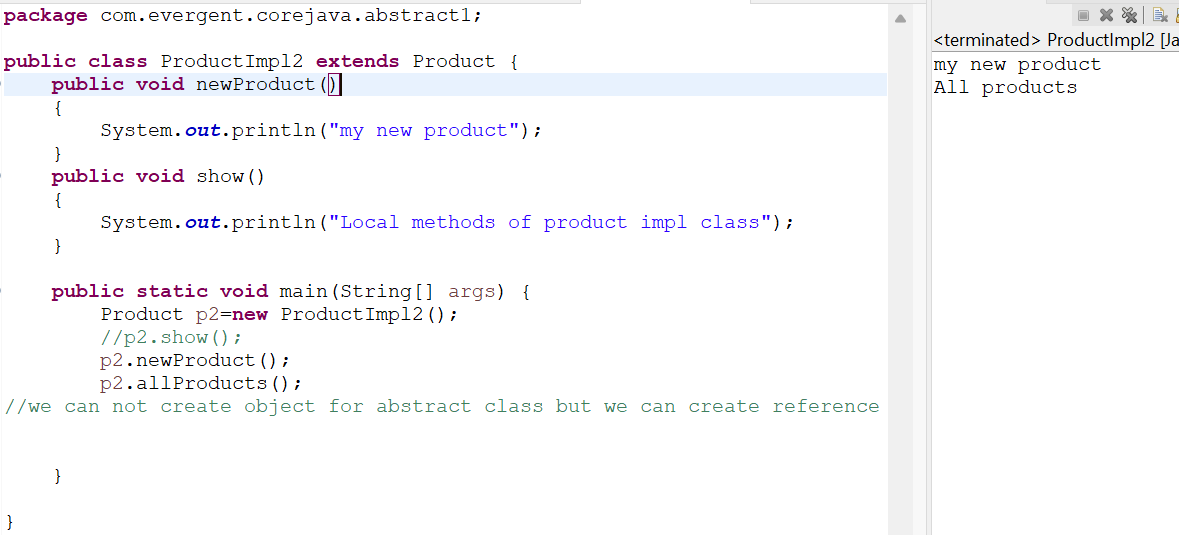
**Program1:**

****

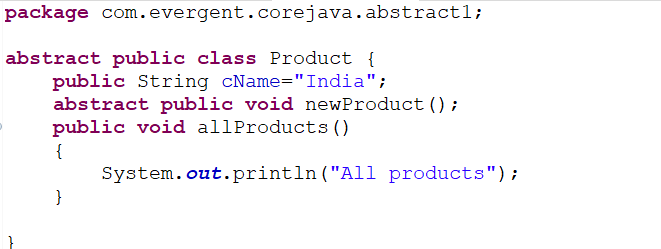
****

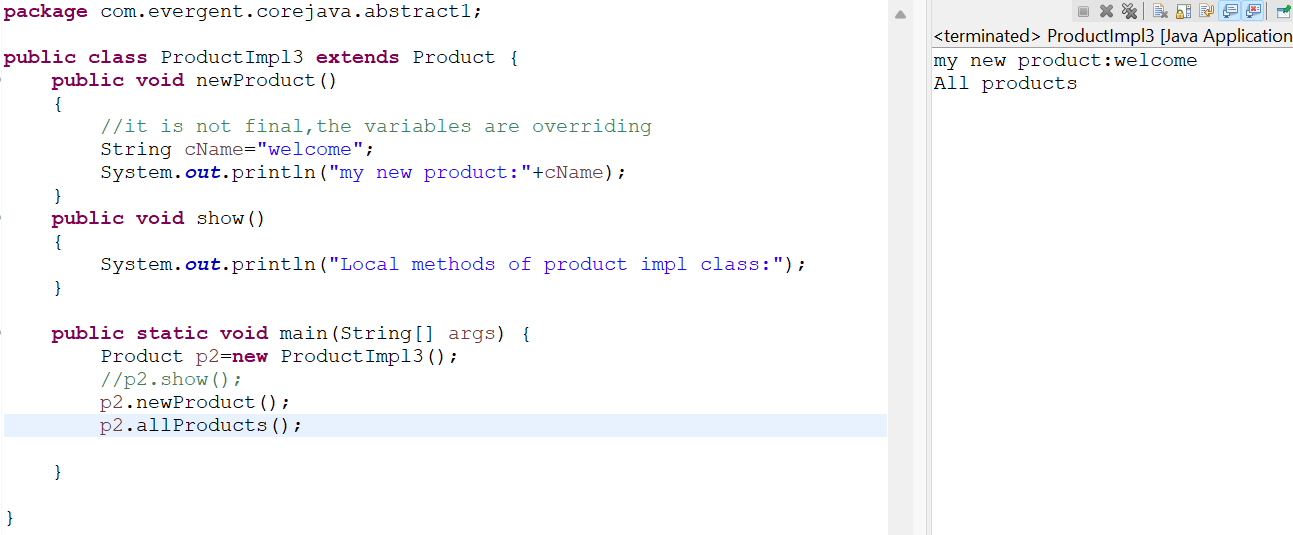
**Program 2:**

****

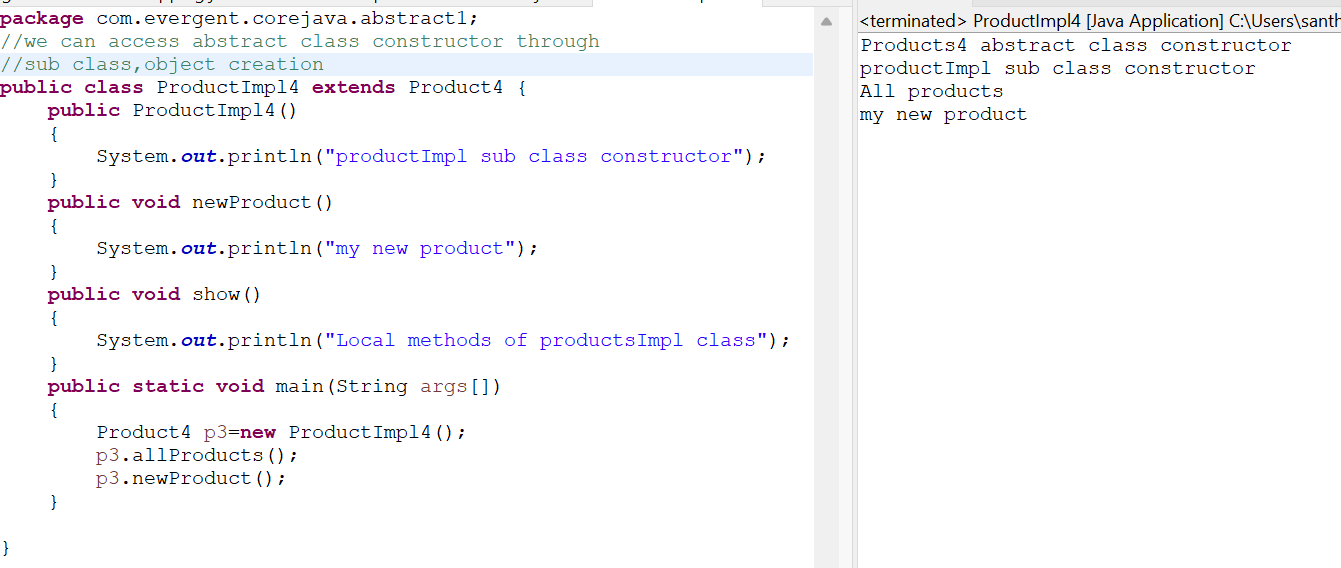
****

**Program 3:**

****

****

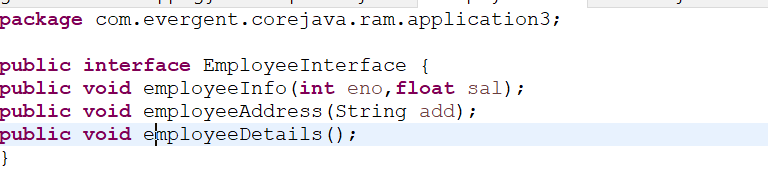
**Program 4:**

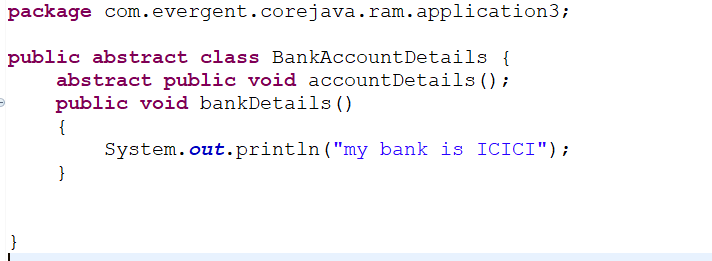
****

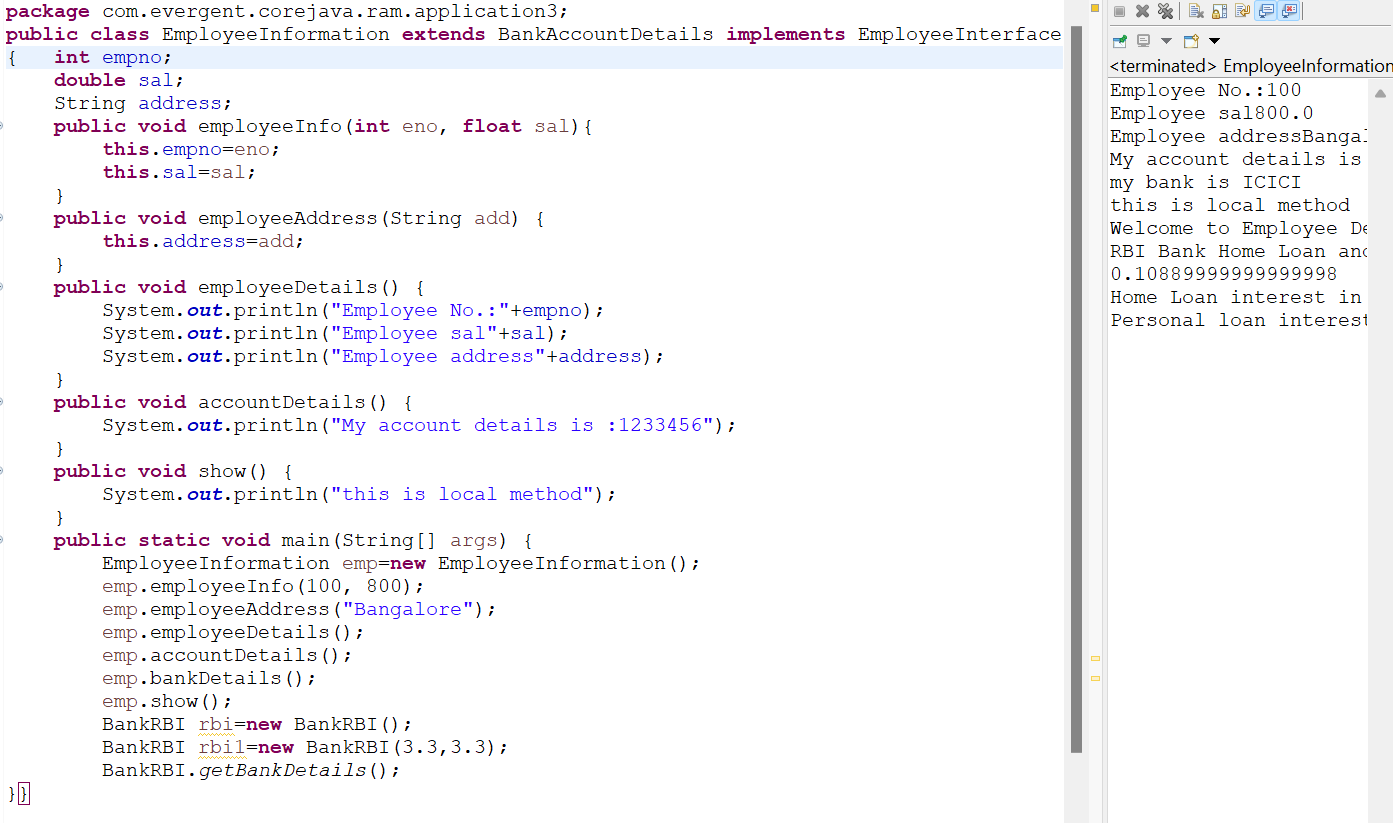
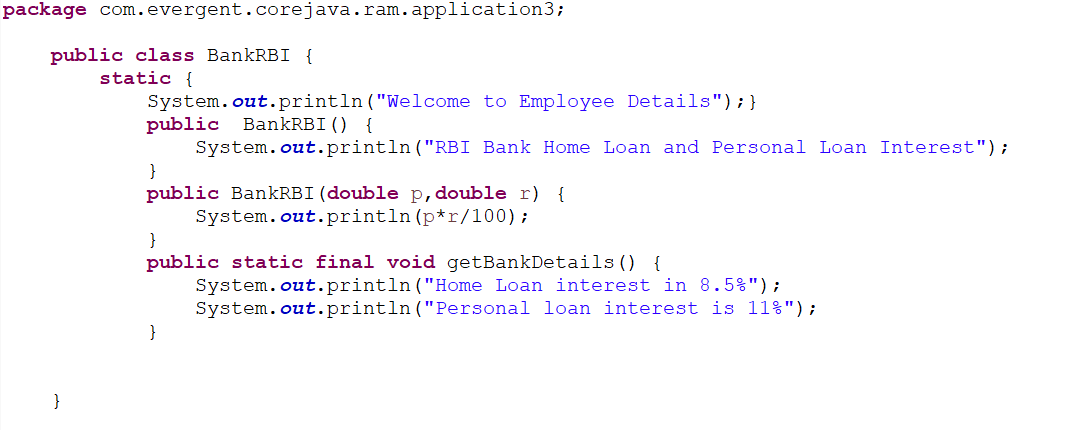
**Date:16/08/2024**

**Day 9**

**Application 3:(Integrated application)**

****

****

****

**Date:19/08/2024**

**Day 10**

1. Exception Handling is a Mechanism.

2. Exceptions are inbuilt mechanism of JAVA.

3. All Exception are executed while abnormal conditions only.

4. Normal flow it won’t execute any exceptions.

5. Once any exceptions are occuring in java, then remaining lines of code is unreachable.

6**.** Java.lang.Throwable is super class for Exception and Error.

7. There are two types of Exceptions in JAVA.

a. Checked Exceptions

b. Unchecked Exceptions

8. All Checked Exceptions are compile time exceptions.

9. All Unchecked Exceptions are Run time exceptions.

10. There are 5 keywords in Exception Handling:

a.Try b. catch() c.finally d. throws e.throw

11. Try is for business logic.

12. Catch is for handling exceptions.

13. Finally is block, if exception is occur or not finally block will be executed.

14. Throws an exception will be executed method by method.

15. Throw is for runtime exceptions and will call predefined exceptions or user defined exceptions.

16. Try followed by either catch block or finally block.

17. We should follow exception hierarchy.

18. We can create our own (user defined) exceptions.

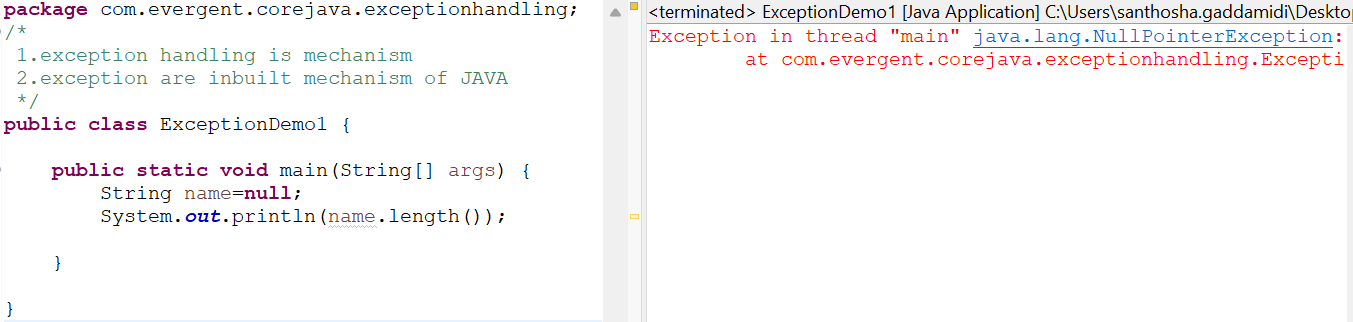
19. Our own exception extends Exception or Runtime Exception.

20. All Exception classes are in to java.lang package.

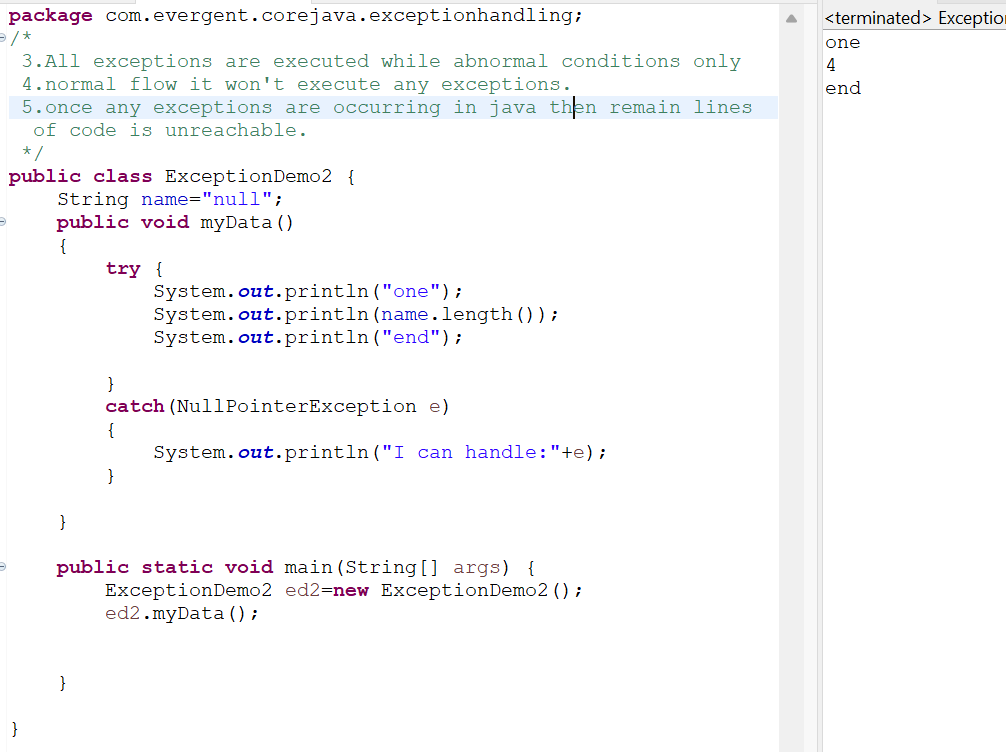
21. There is two exceptions in class, Developer should be handle one after one.

22. Developer can’t handle Errors.

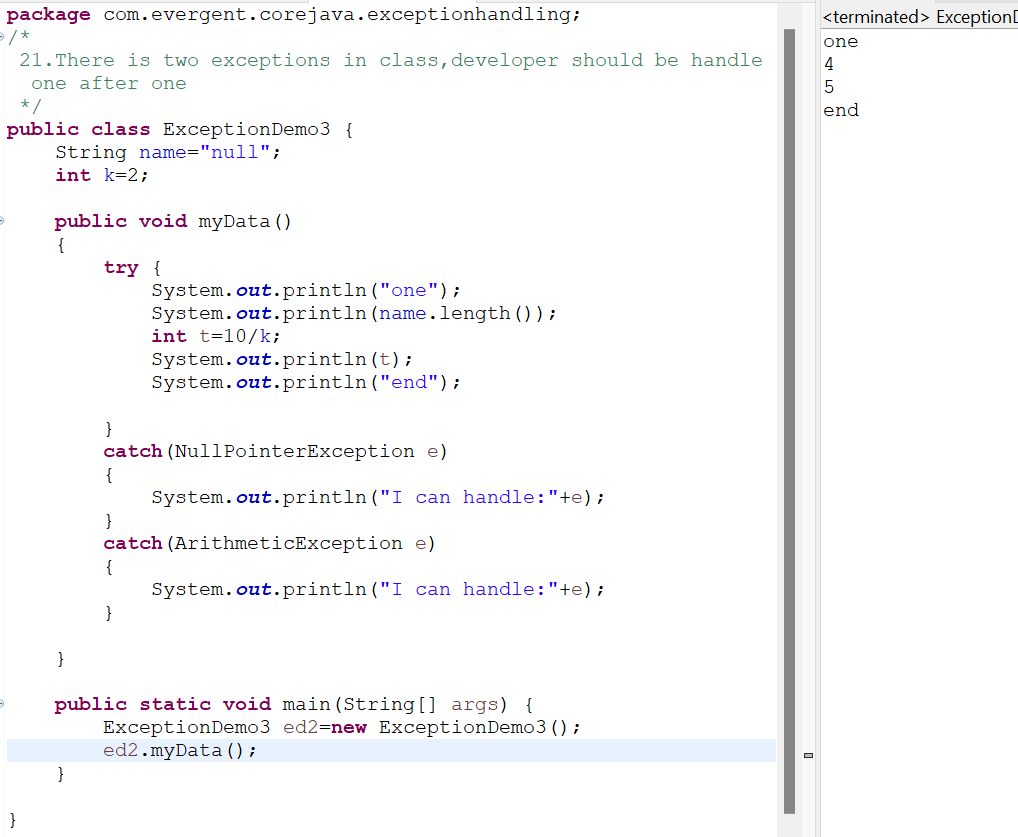
**Program 1:**

****

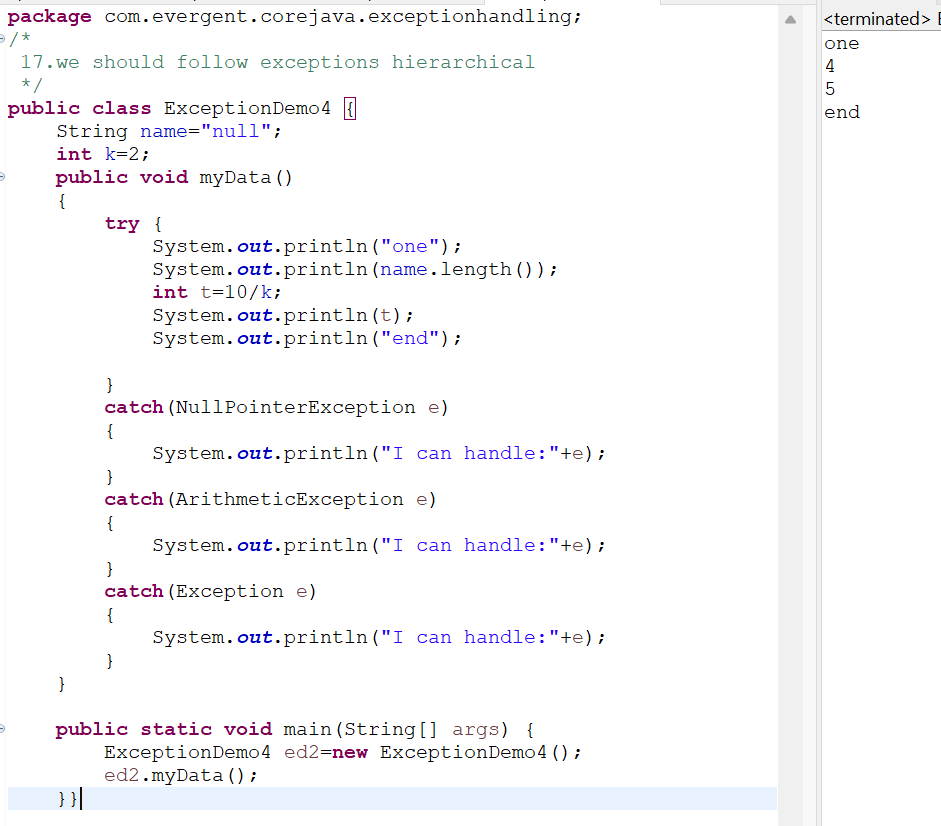
**Program 2:**

****

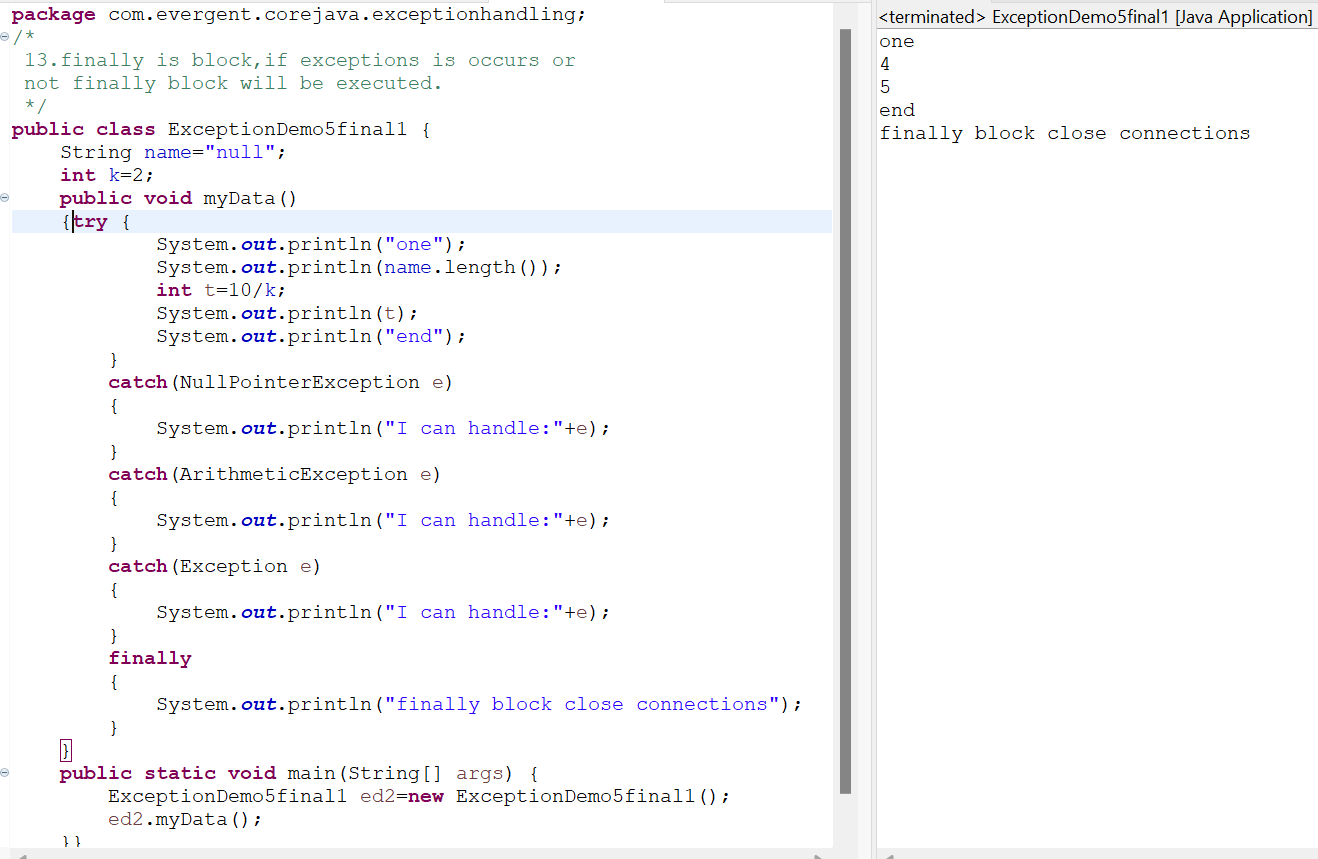
**Program 3:**

****

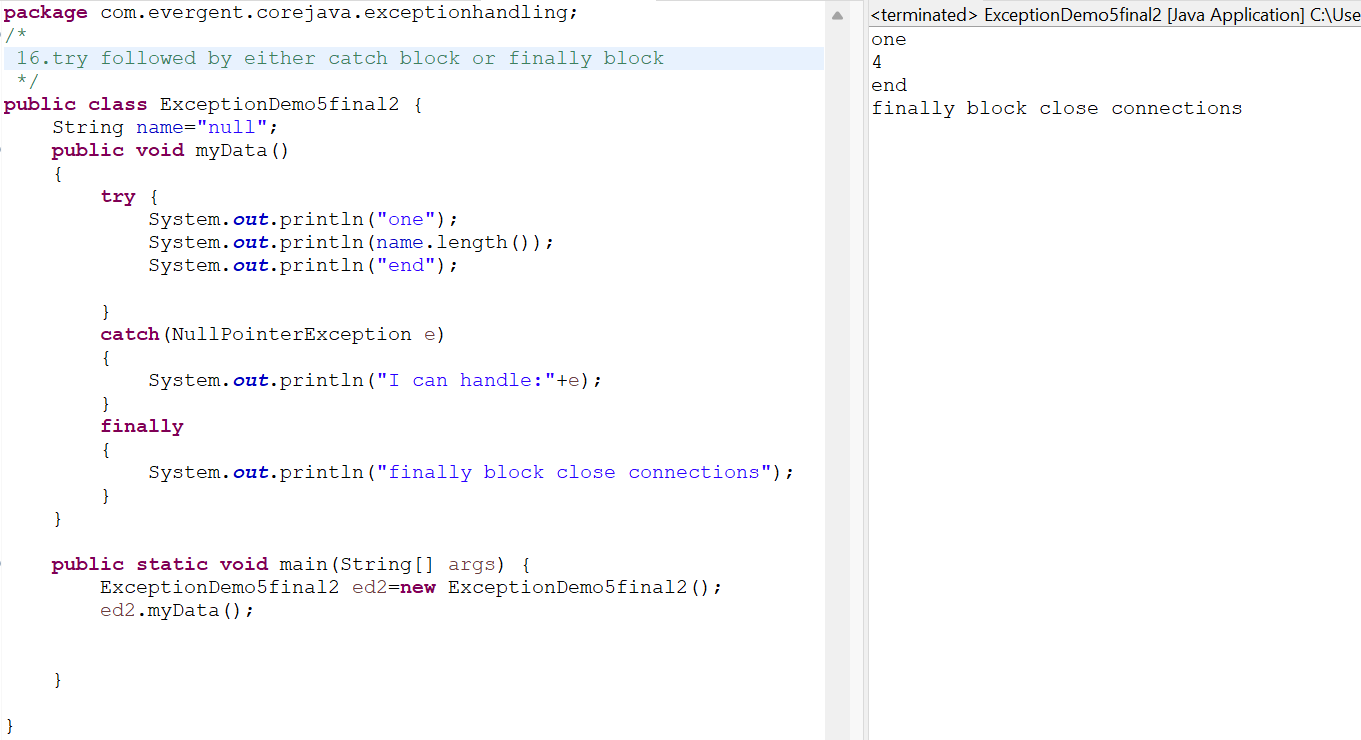
**Program 4:**

****

**Program 5:**

****

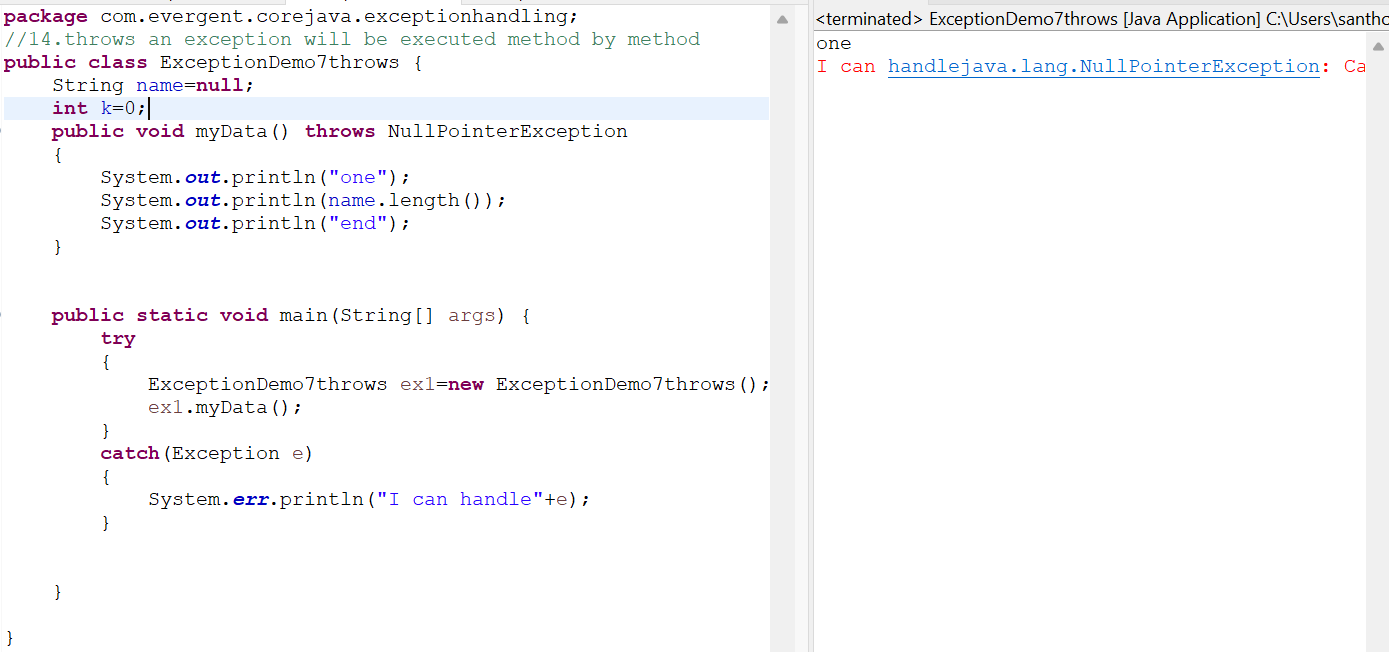
**Program 6:**

****

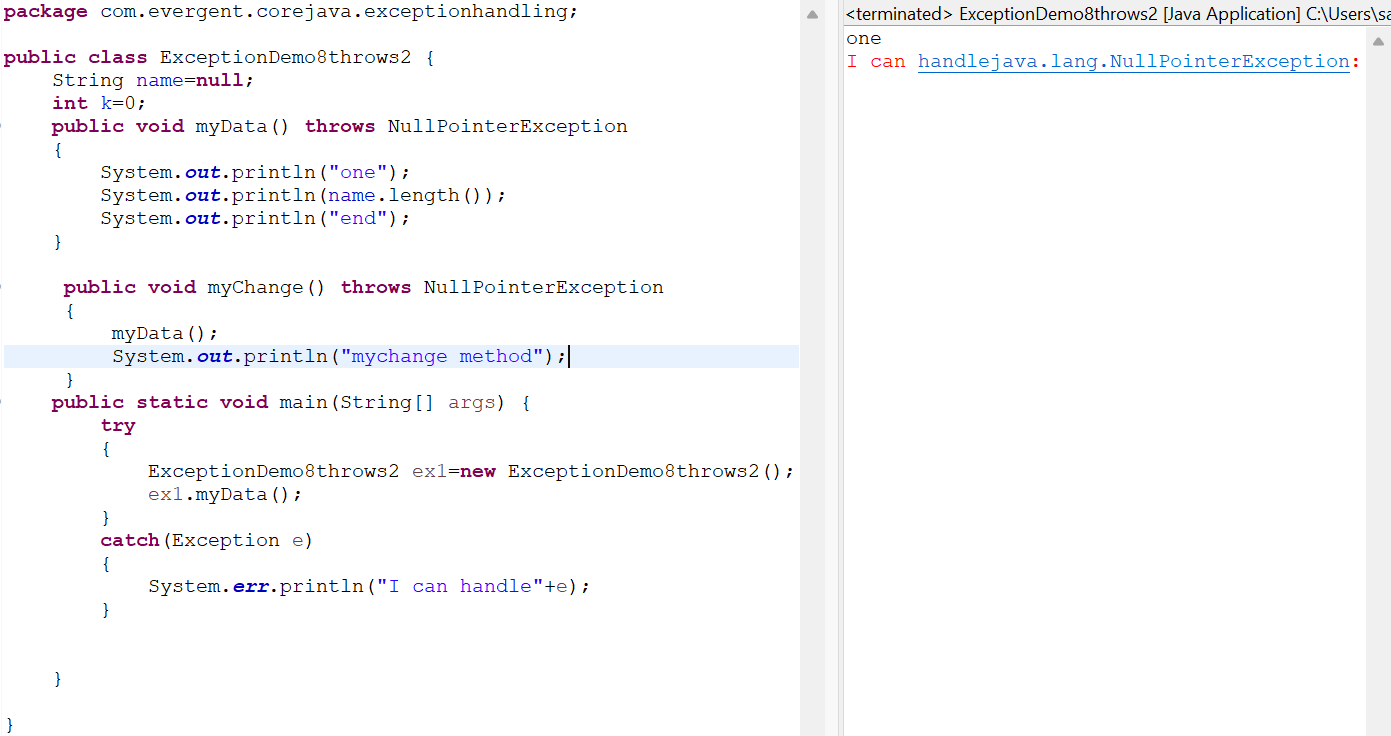
**Date:20/08/2024**

**Day 11**

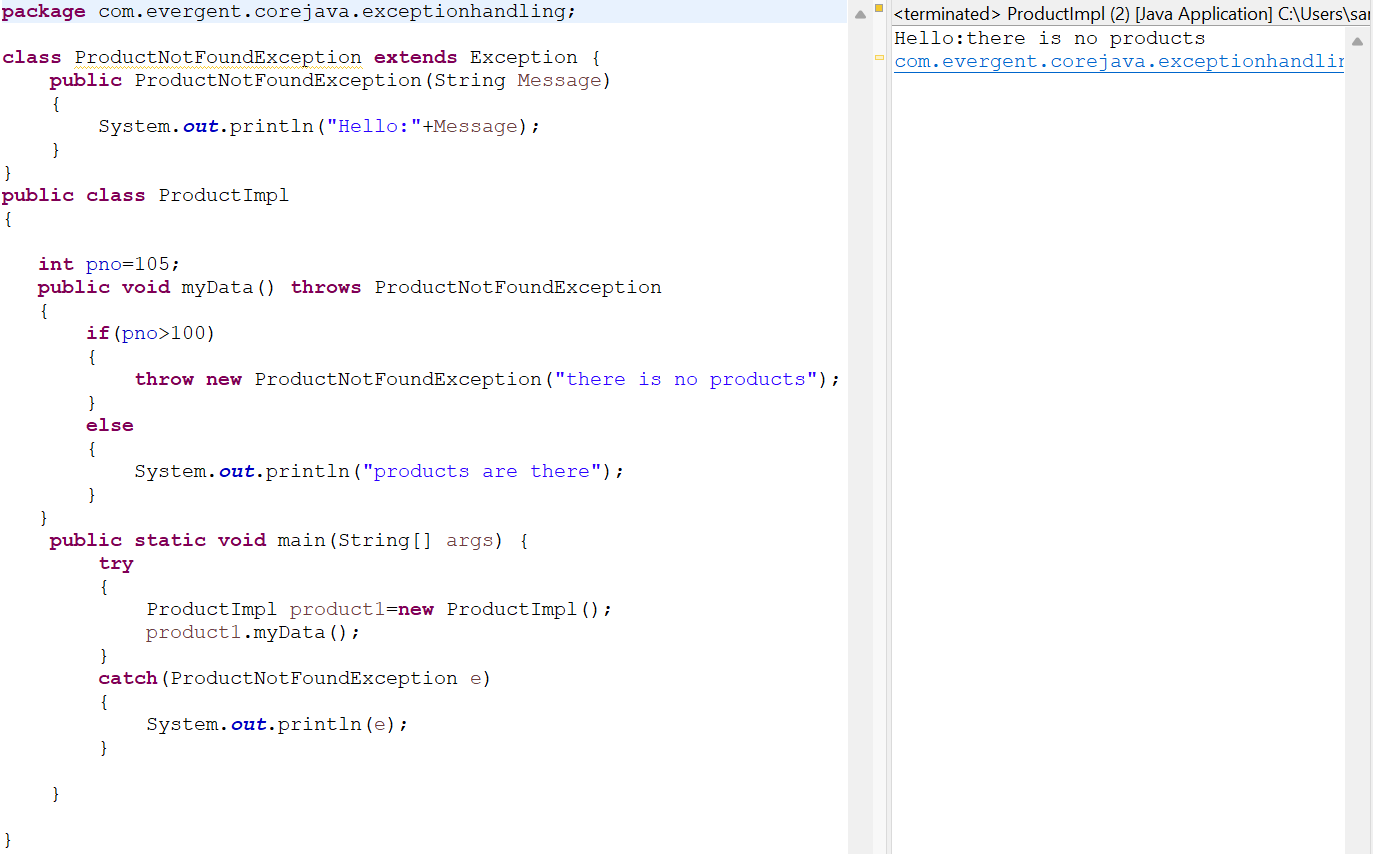
**Program 7:**

****

**Program 8:**

****

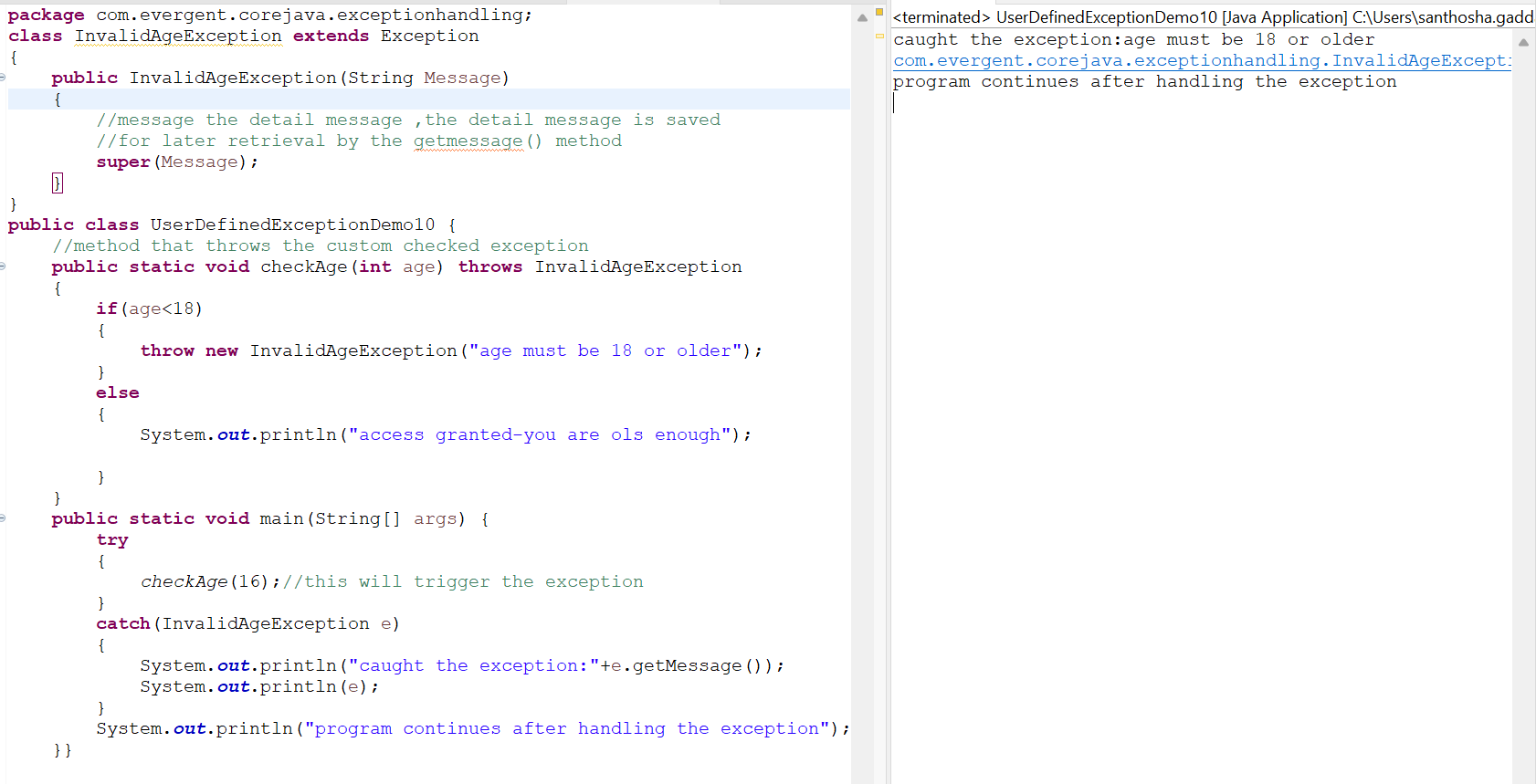
**Program 9:**

****

**Date:21/08/2024**

**Day 12**

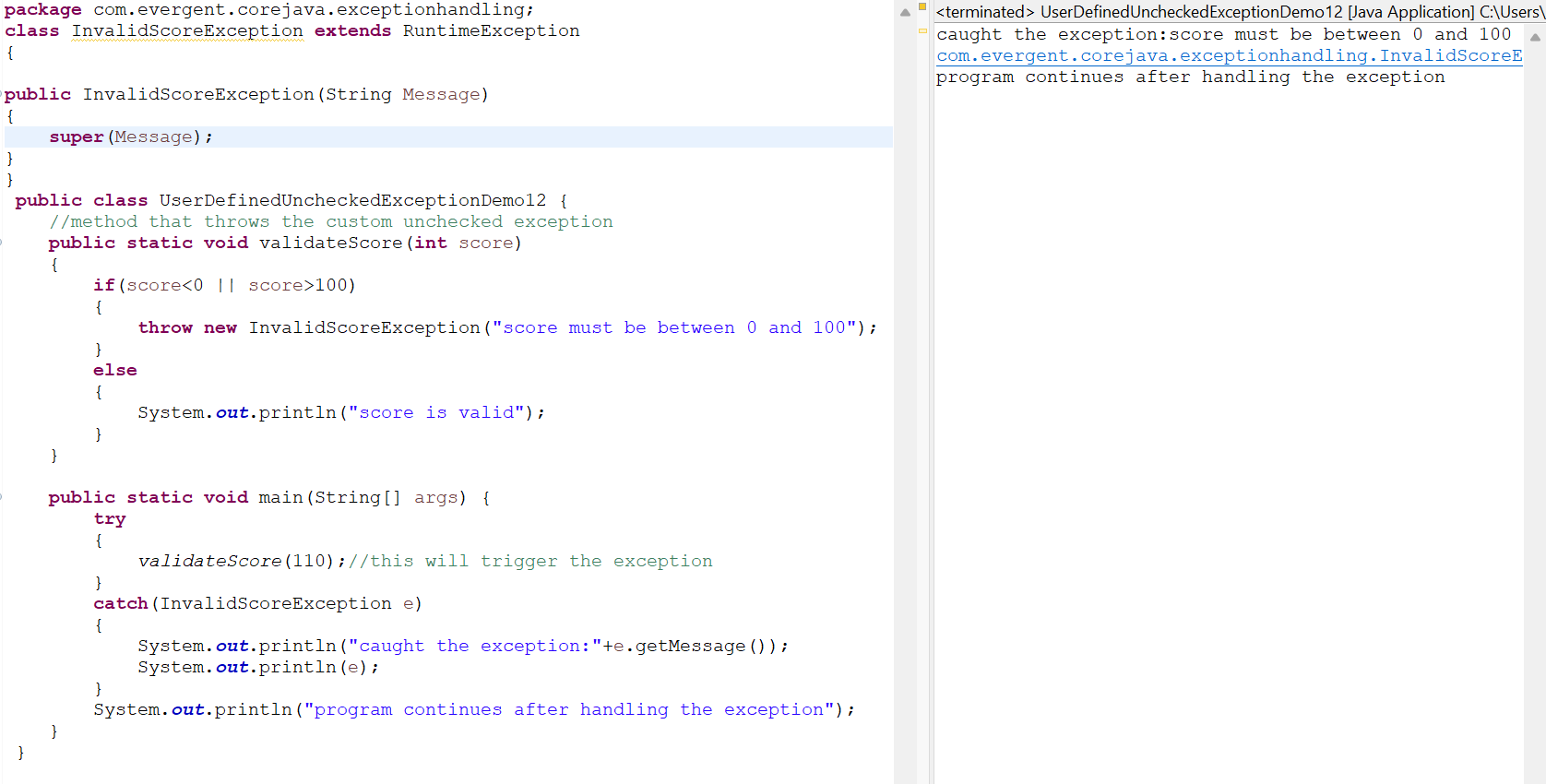
**Program 10:**

****

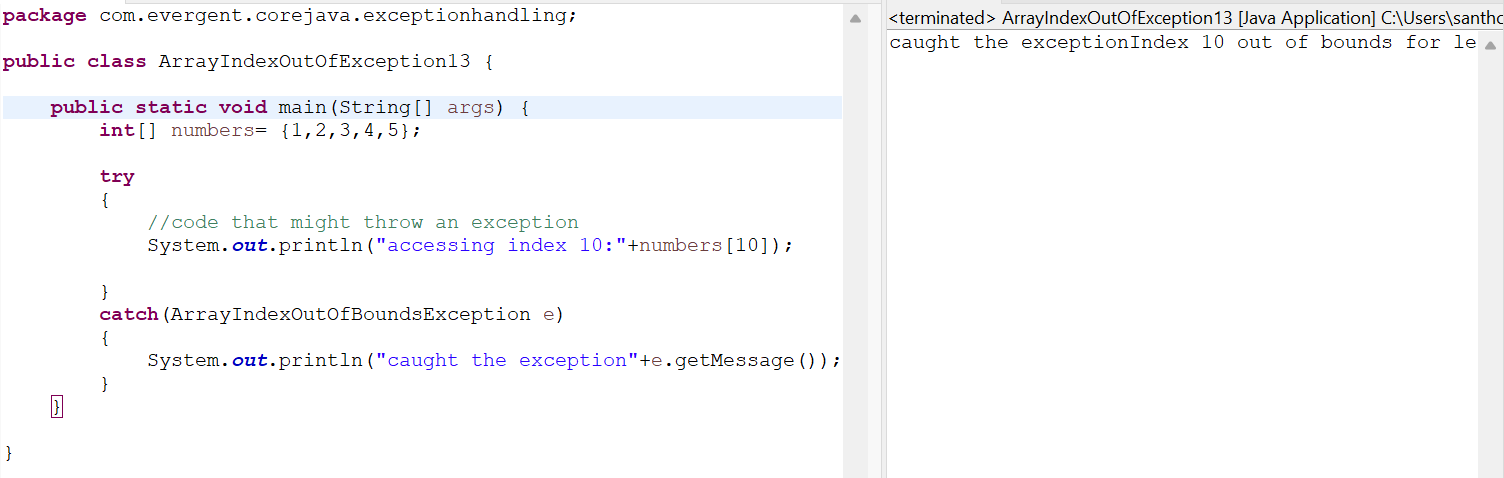
**Program 11:**

****

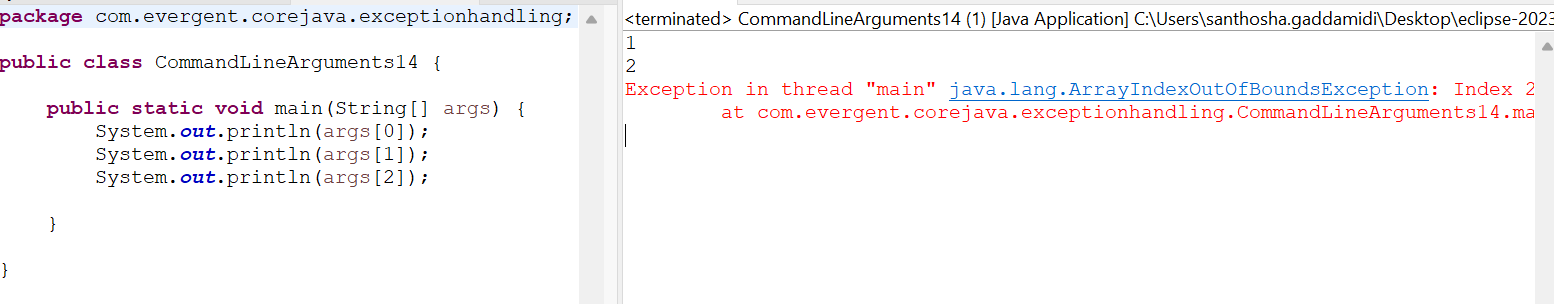
**Program 12:**

****

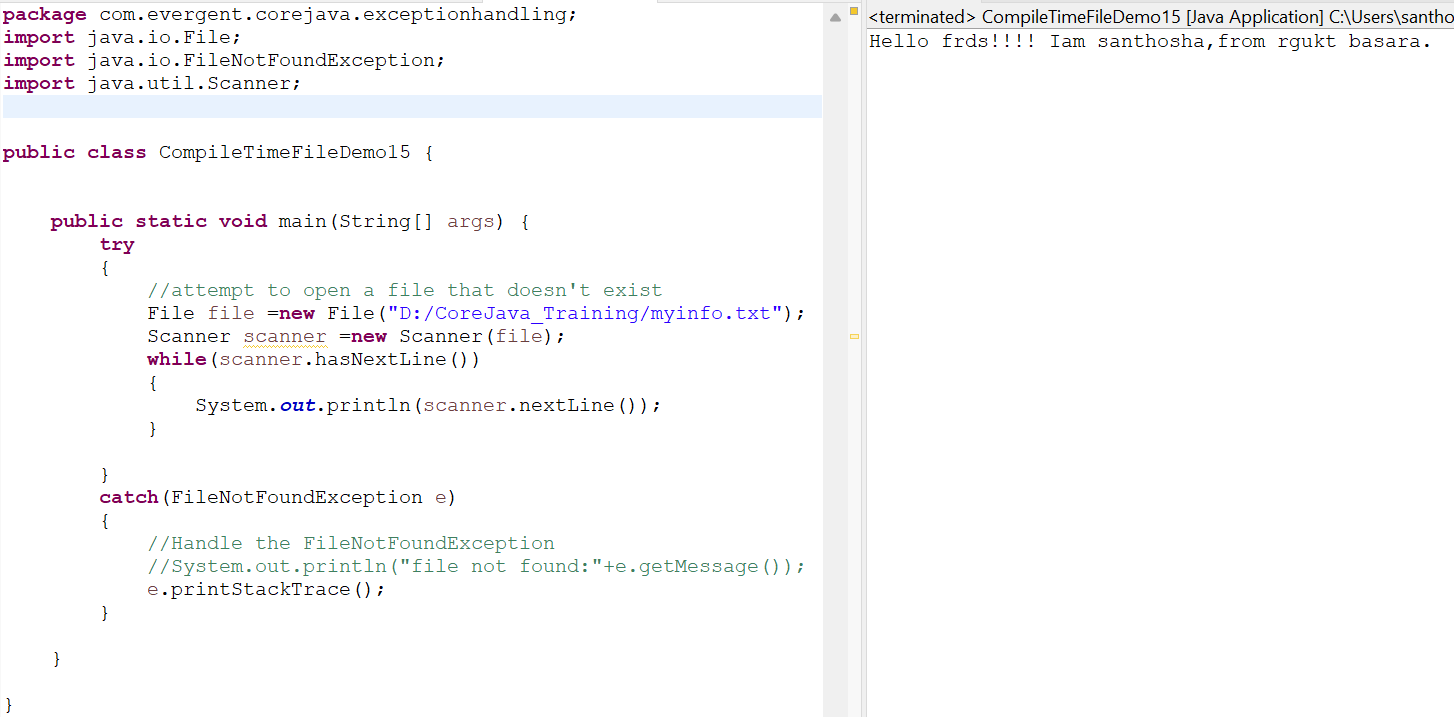
**Program 13:**

****

**Program 14:**

****

**Program 15:**

****

**JAVA BEANS:**

1.Java bean is a mechanism.

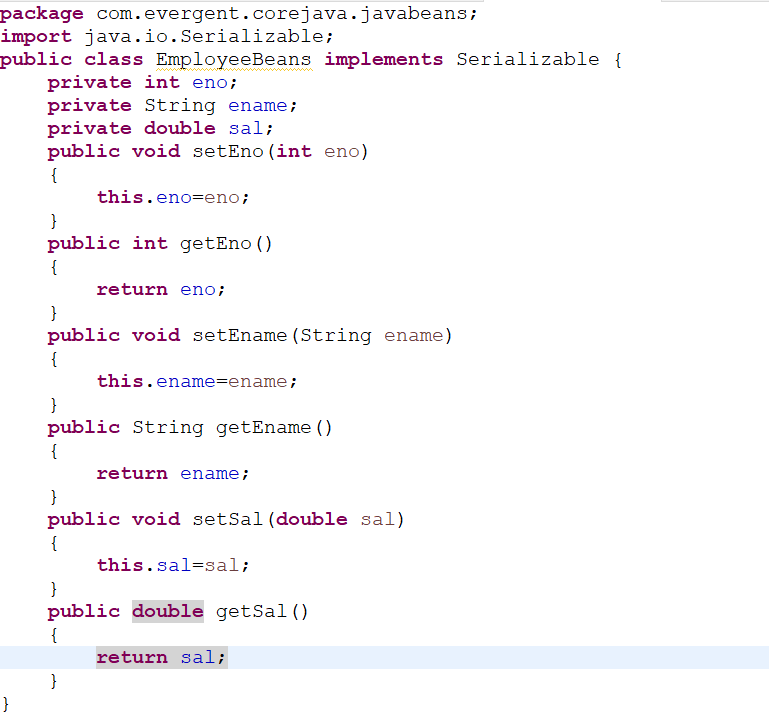
2.Java bean is light weight.

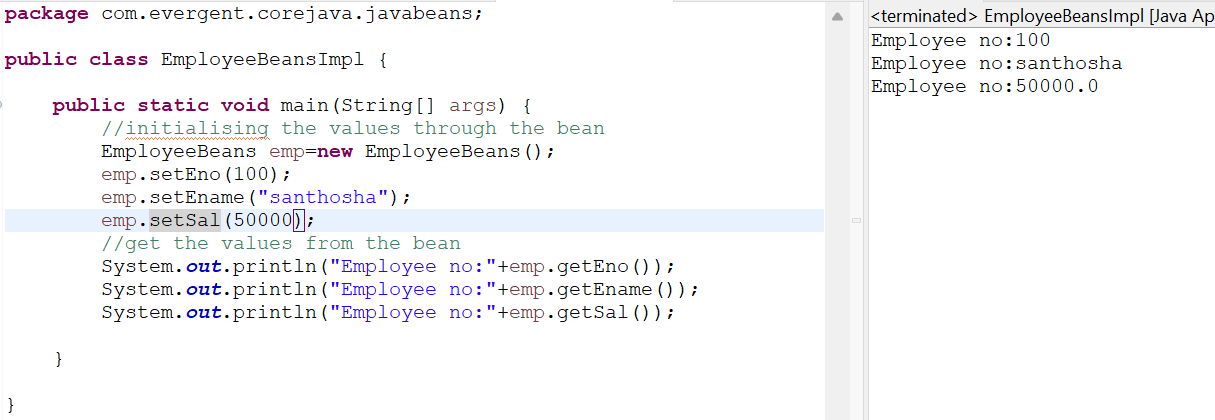
3.All attributes are private and get/set methods are public.

4.implements java.io.serializable interface.

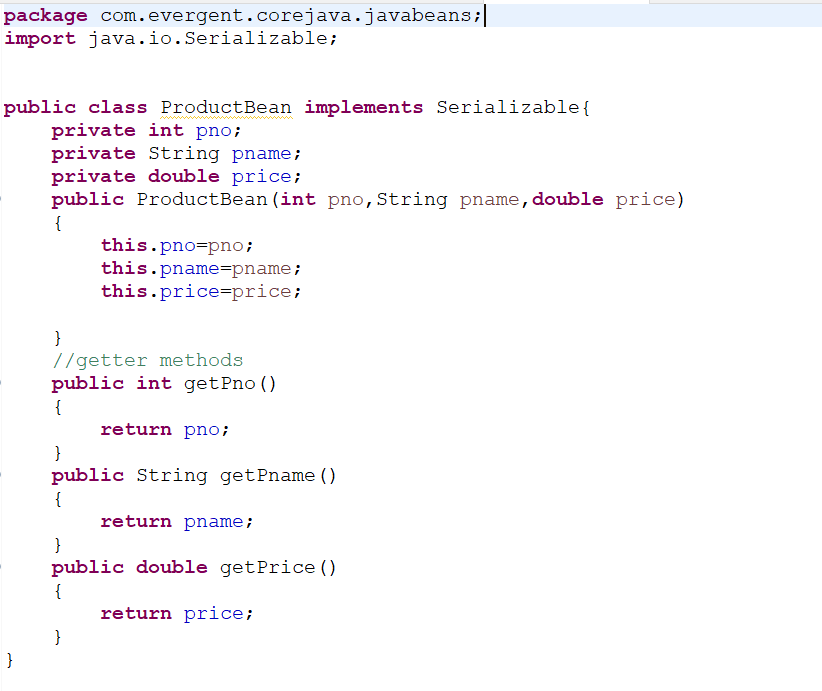
5.we can achieve tightly encapsulation through java beans.

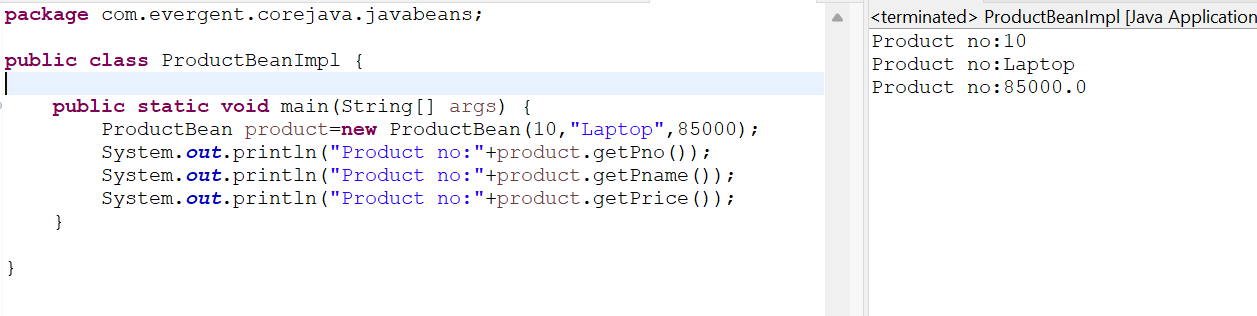
**Program 1:**



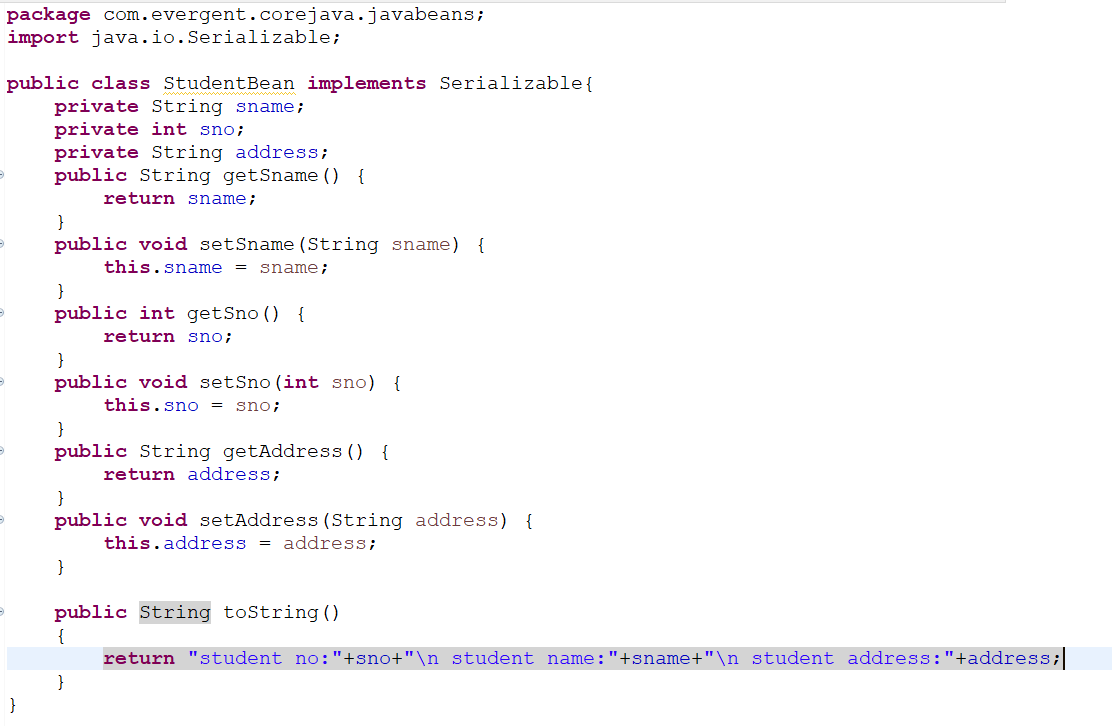


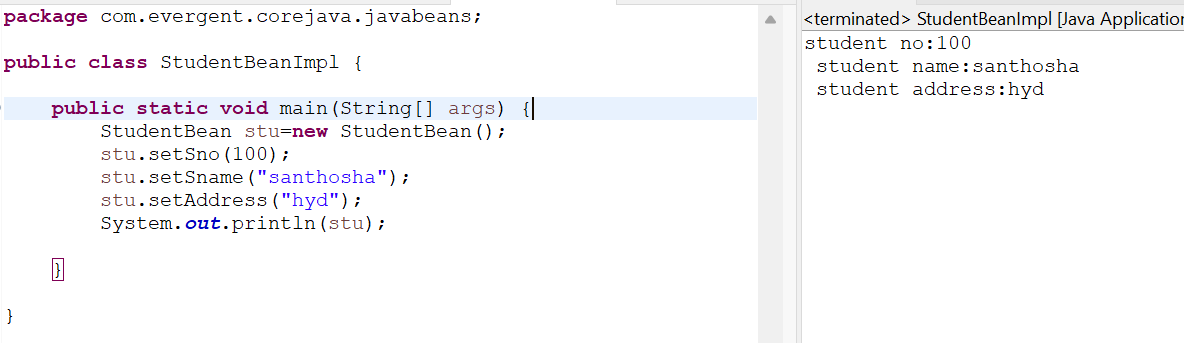
**Program 2:**

****

****

**Program 3:**

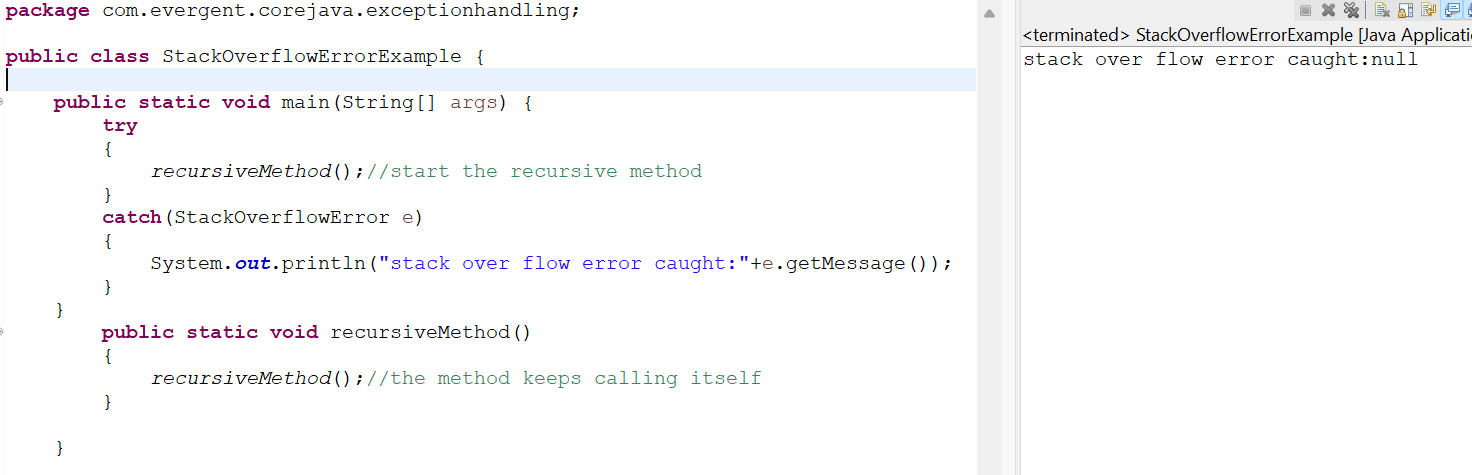
****

****

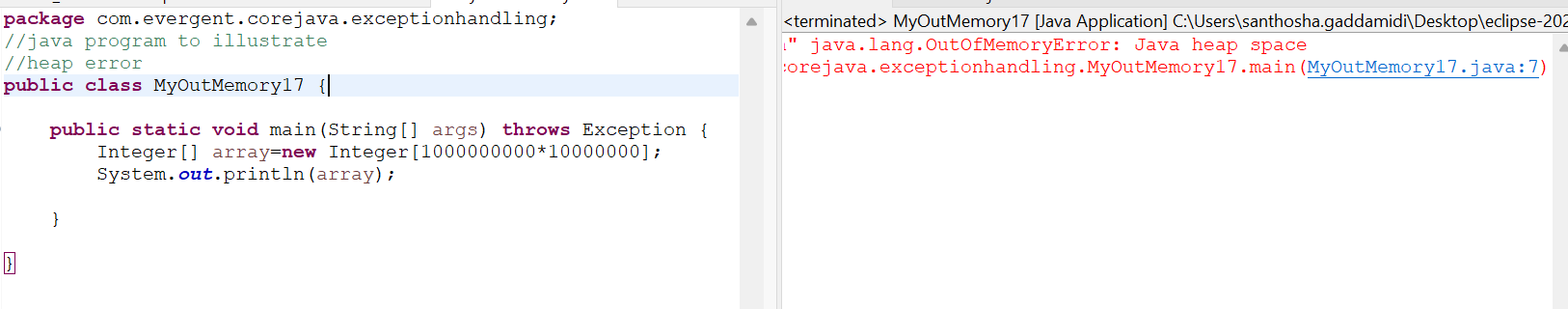
**Date:22/08/2024**

**Day 13**

**Program 16**

****

**Program 17**

****

**Git Commands:**

1.git init

2.git status

3.git add .

4.git status

5.git commit -m “abstract”

6.git branch

7.git remote add origin <https://github.com/santhoshagaddamidi/Core_Java.git>

8.git push -- force origin master