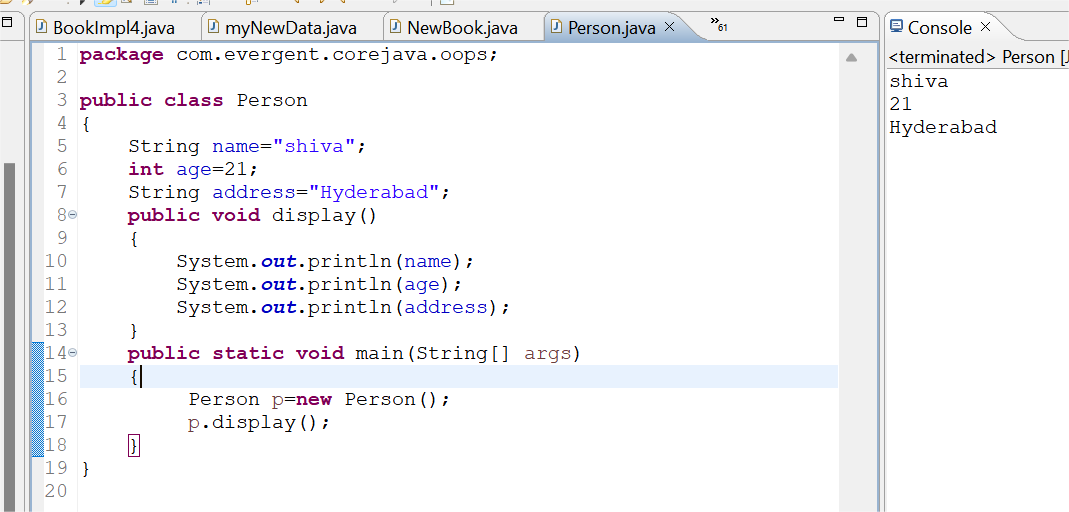
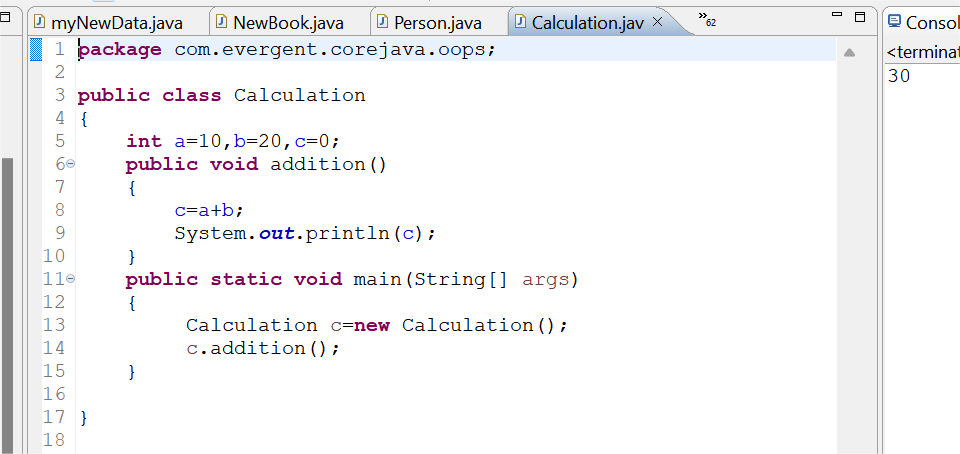
**Core JAVA training - index**

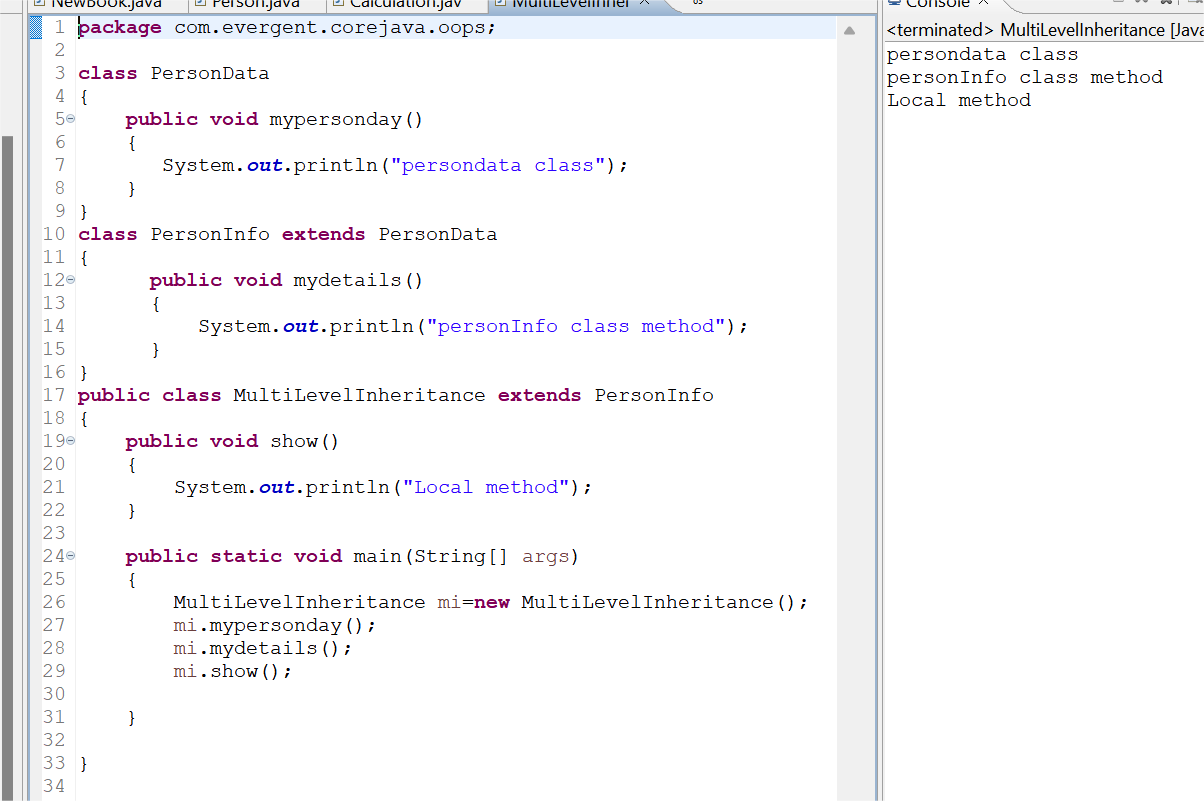
**Day 3: 07-Aug-2024**

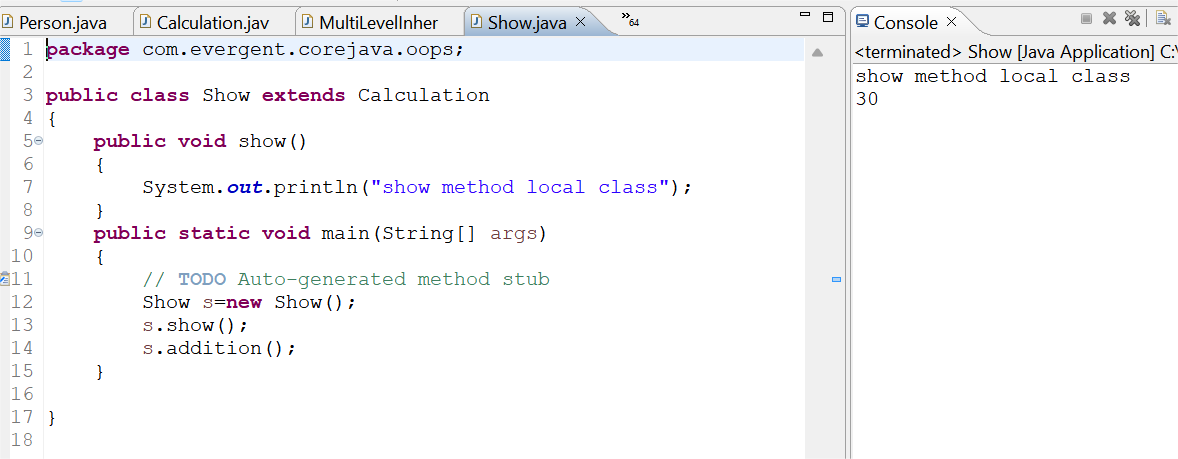
1. **OOPS**
   1. **Encapsulation:**
      1. Binding attributes and methods together inside a class and object creation is called Encapsulation.
      2. We cannot access attributes and methods without object creation. But we can access attributes and methods through object creation with reference.

****

****

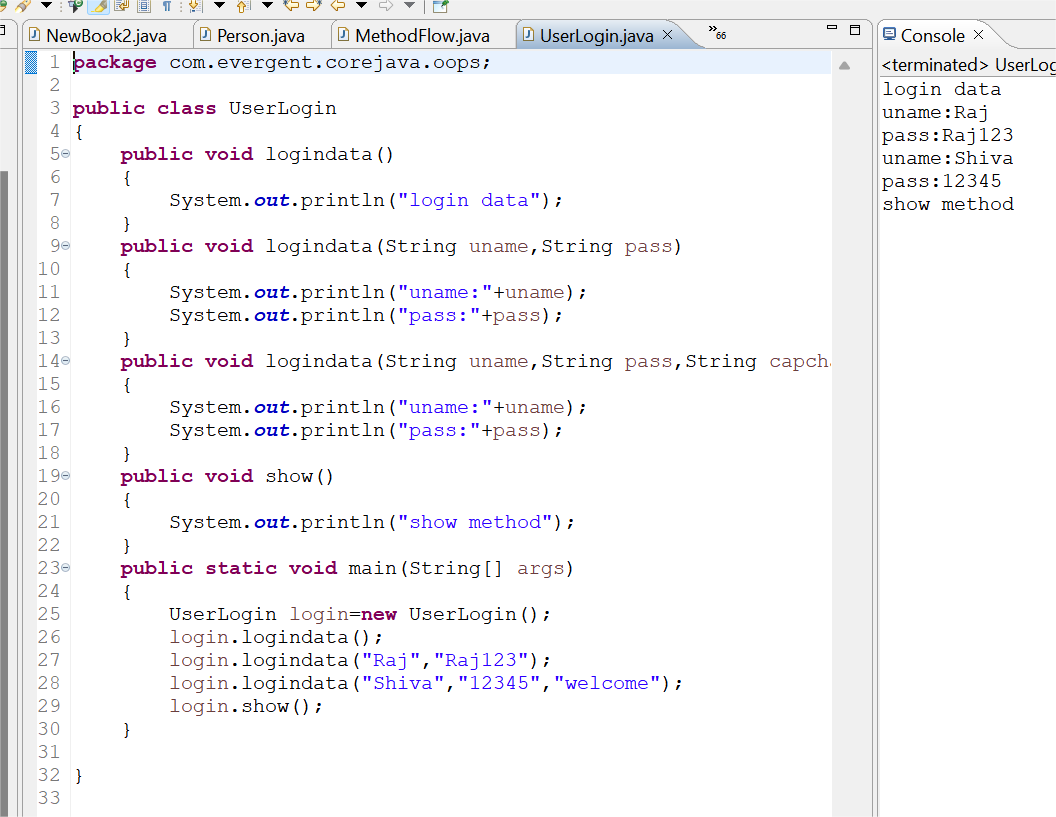
* 1. **Inheritance:**
     1. Re-usability of existing functionalities from super class to its subclass.
     2. Java won’t support multiple inheritance through classes, but it will support through Interfaces.

****

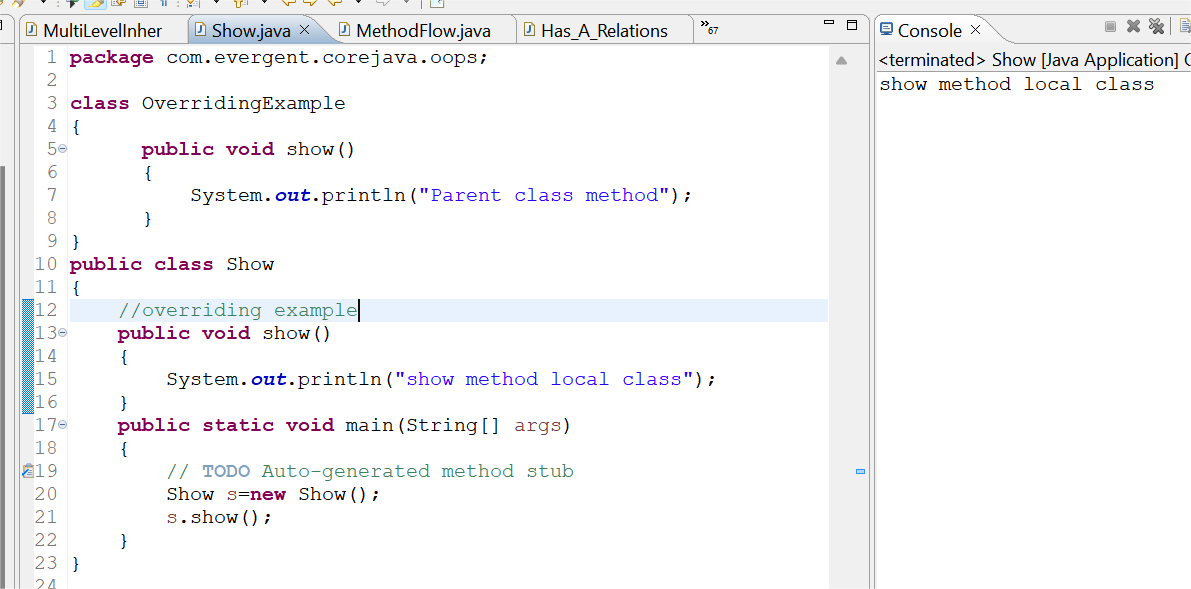
****

* 1. **Polymorphism:**
     1. Overloading:
        1. Method names are same, parameters should be different, return type may or may not be same, It happens in same class or different class.

**Overloading**

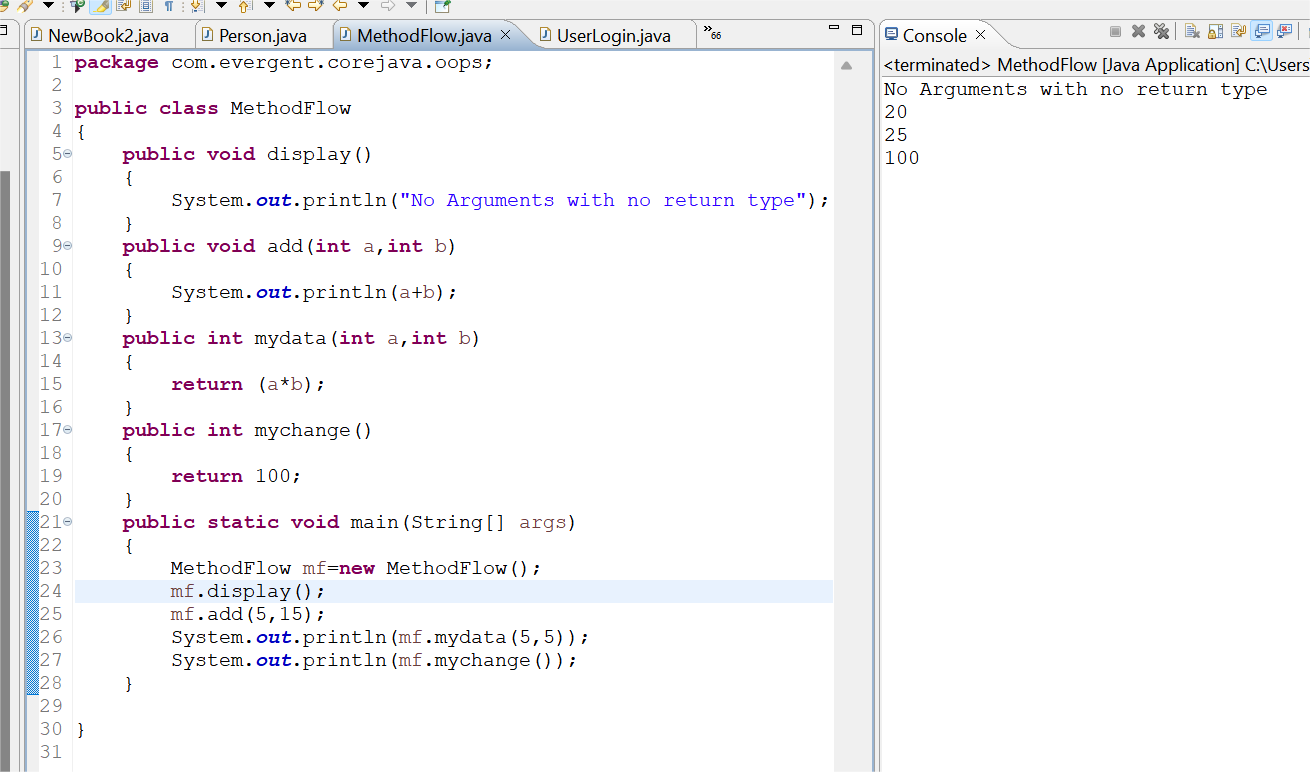
****

Overriding

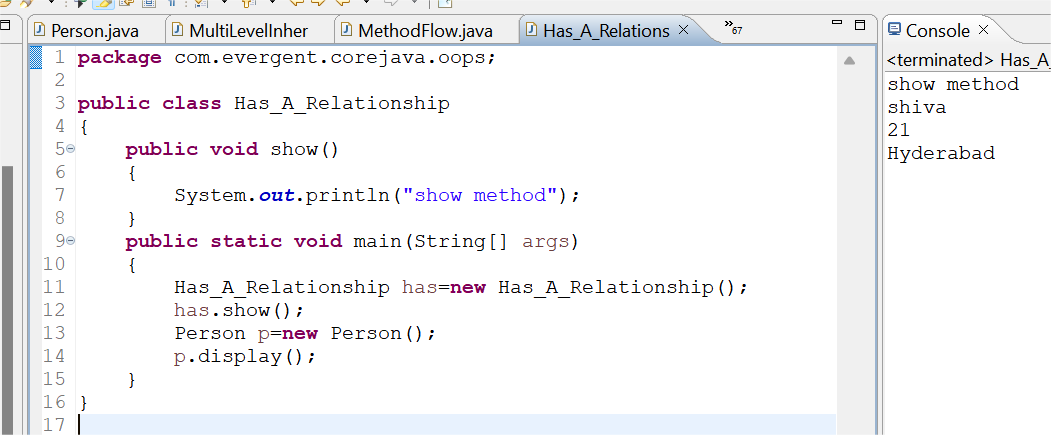


* 1. **Abstraction:**
     1. Hiding the irrelevant data and showing the relevant data to the end user.
  2. **Method Flows:**
     1. With parameters, With return values.
     2. With parameters, Without return values.
     3. Without parameters, With return values.

Without parameters, Without return values



Has-a Relationship



1. **System class**
   1. System - is a class
   2. Out - is reference of PrintStream
   3. Println - method

**Date:05/08/2024**

1. Language And Applications
2. JAVA Features

* Why Java is Independent?
* Oops
* Exception Handling
* Multi threading
* Web Application
* Open Source
* Security
* Support Networking
* Memory Management

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

**Date:06/08/2024**

**Morning(11:00 am)**

1. Nested Loops
2. One Dimensional Array
3. Two Dimensional Array
4. Logical Programming

**After Noon(3:30 pm)**

1. SwitchCase
2. Scanner Class
3. Java.lang

* Object Class Methods

1. Enum
2. Event Management Application

**Date:07/08/2024**

**Mrng(11:00 am)**

1.oops

* Encapsulation

Programs

Calculation

Person

Method Flow

**After Noon(3:30 pm)**

1. Inheritance
2. Polymorphism

**.**Method overloading

**.**Method Overriding

1. Abstraction
2. IS-A (Inheritance)
3. HAs- A (Object Creation).

**Date:08/08/24**

**Constructor**

i. Class name and constructor name should be same

ii. There are 2 types of constructors

a. Default Constructor

b. Parameterized Constructor

iii. We can access constructor while creation of object

iv. Constructors are mainly for initializing

v. Constructor doesn’t have any return type not even void. If you declare as a void the compiler will consider as a method not a constructor

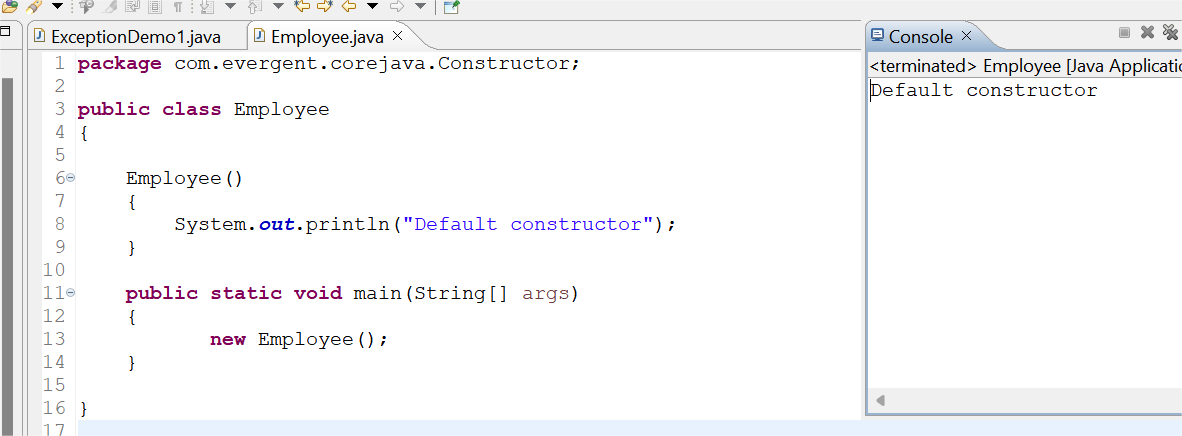
vi. Every class needs atleast 1 default constructor

vii. this, super This

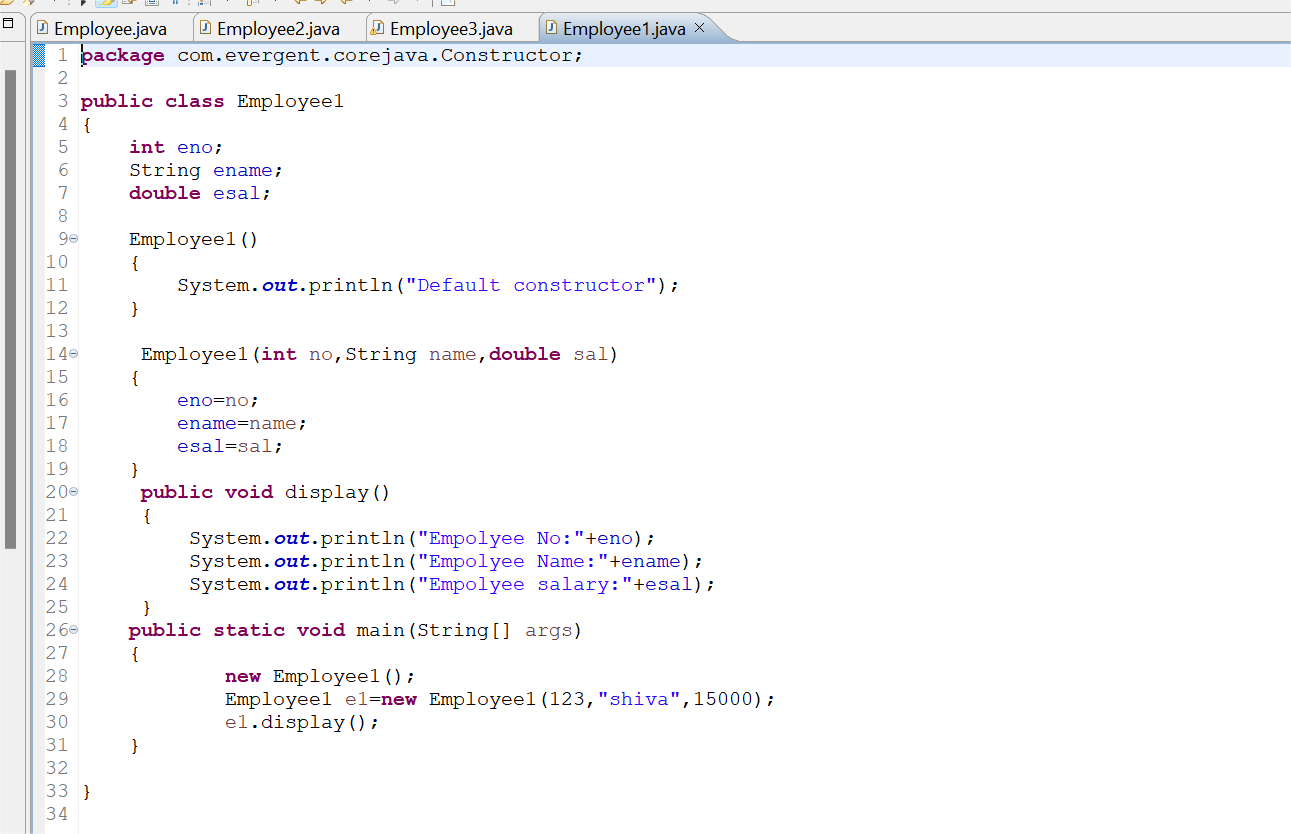
--> this is a keyword always refers to instance variables

viii. Always constructor are overloaded

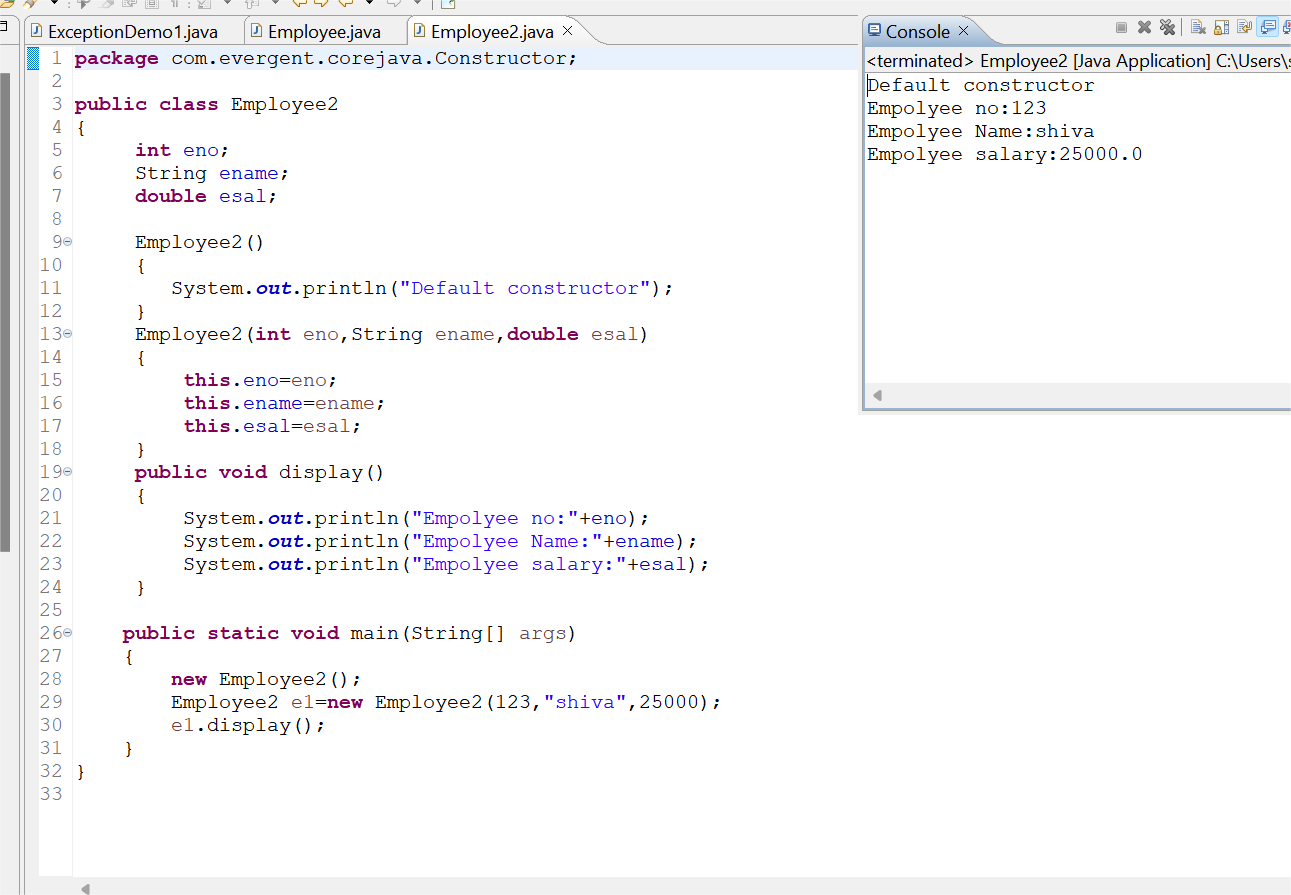
**Program1:**

****

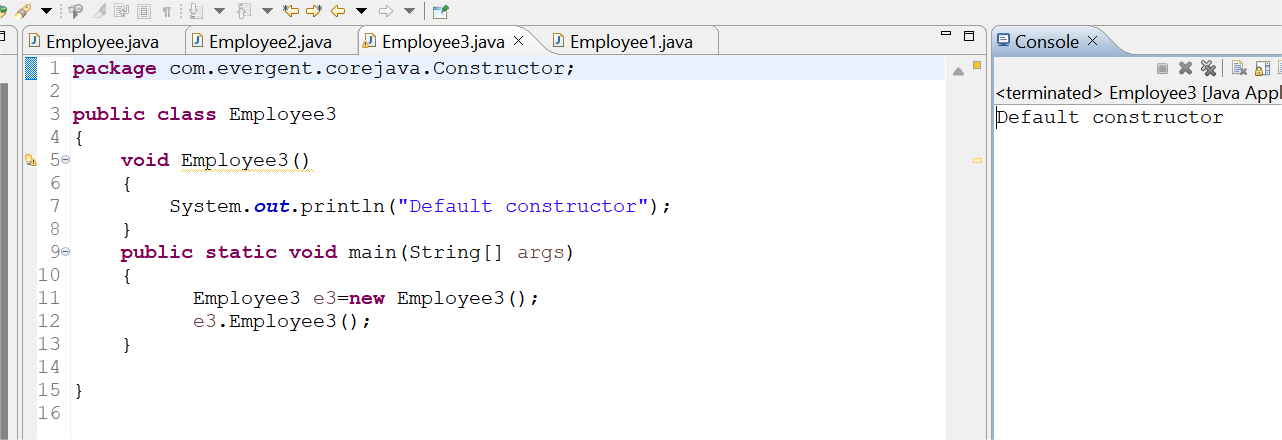
**Program2:**

****

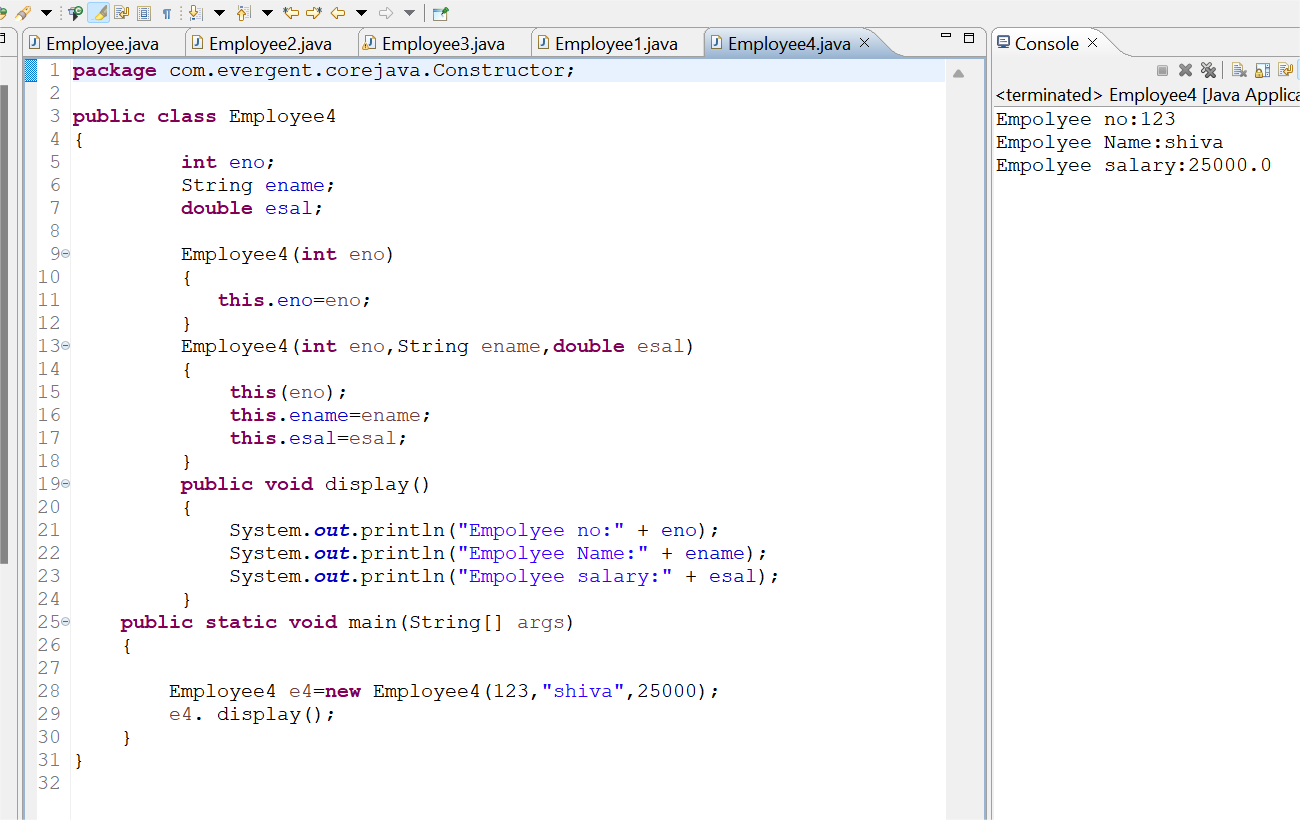
**Program3:**

****

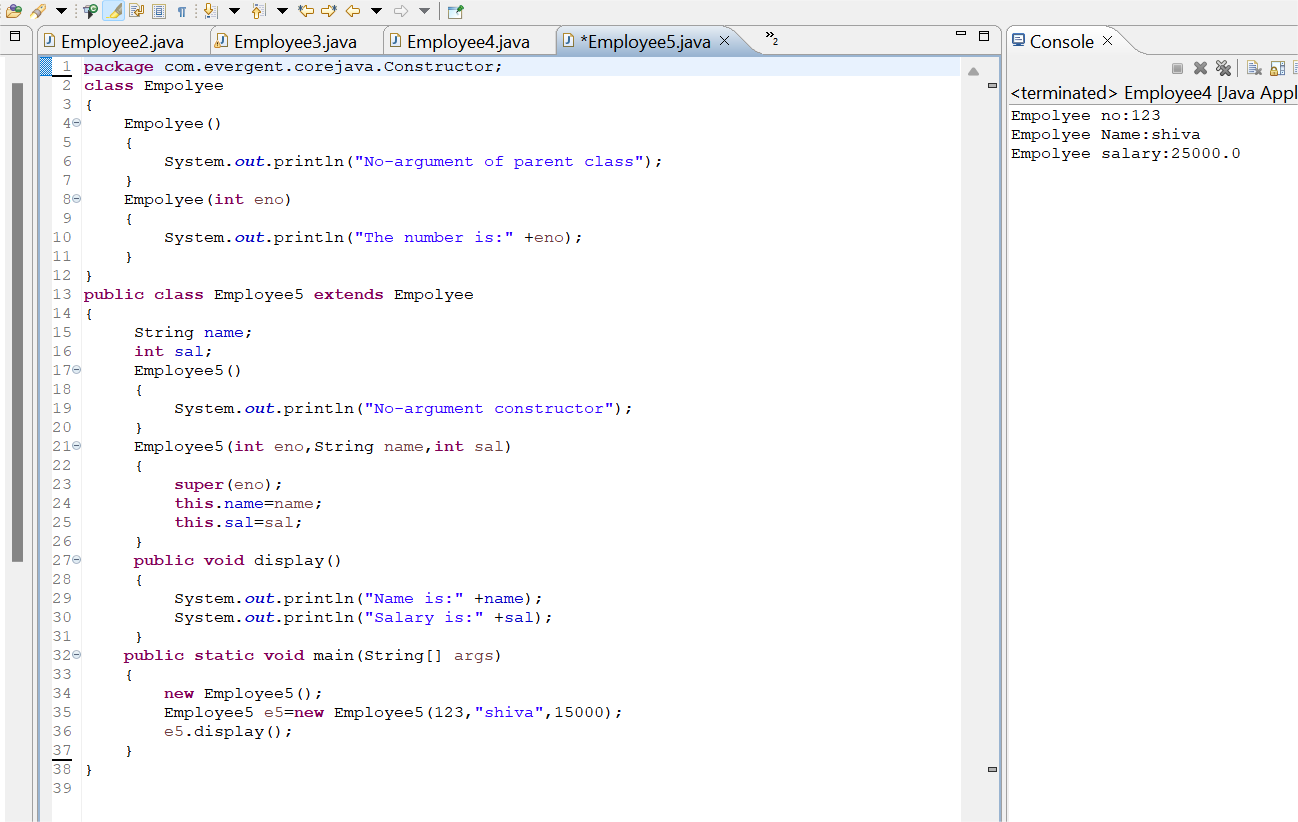
**Program4:**

****

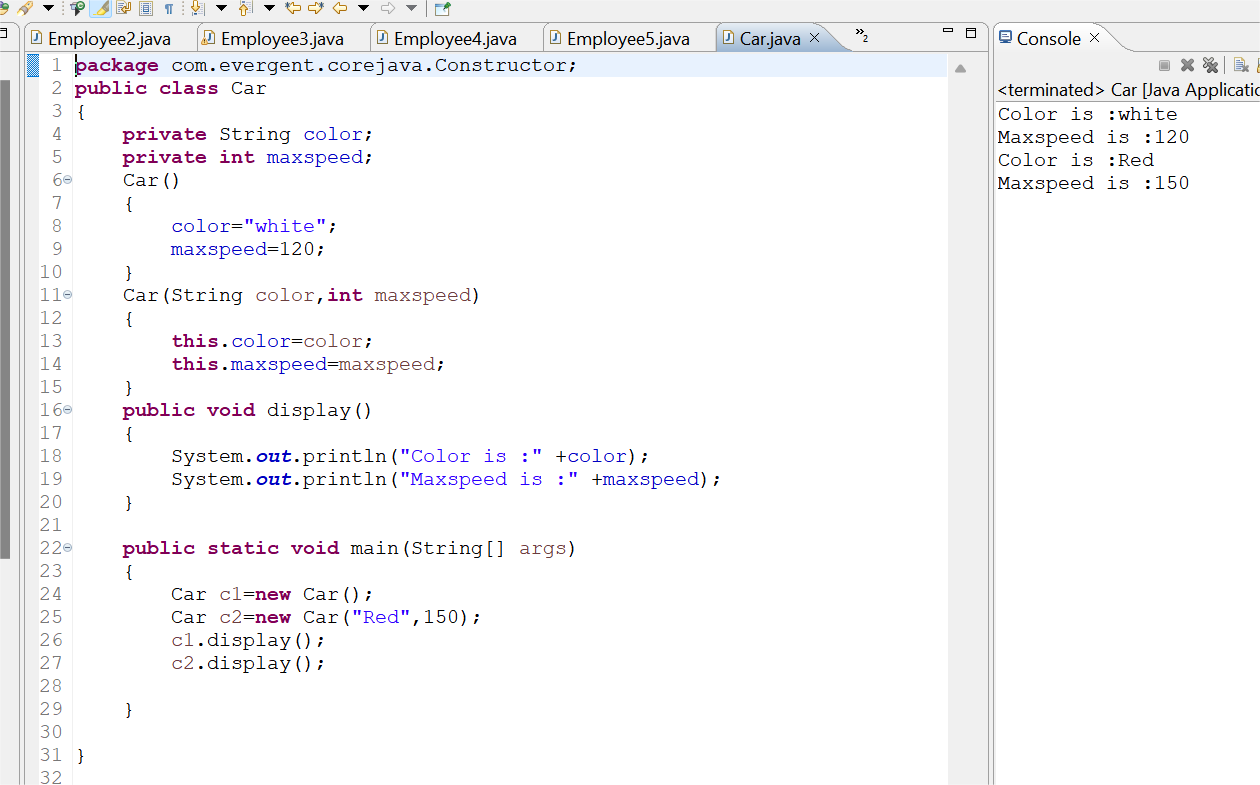
**Program5:**

****

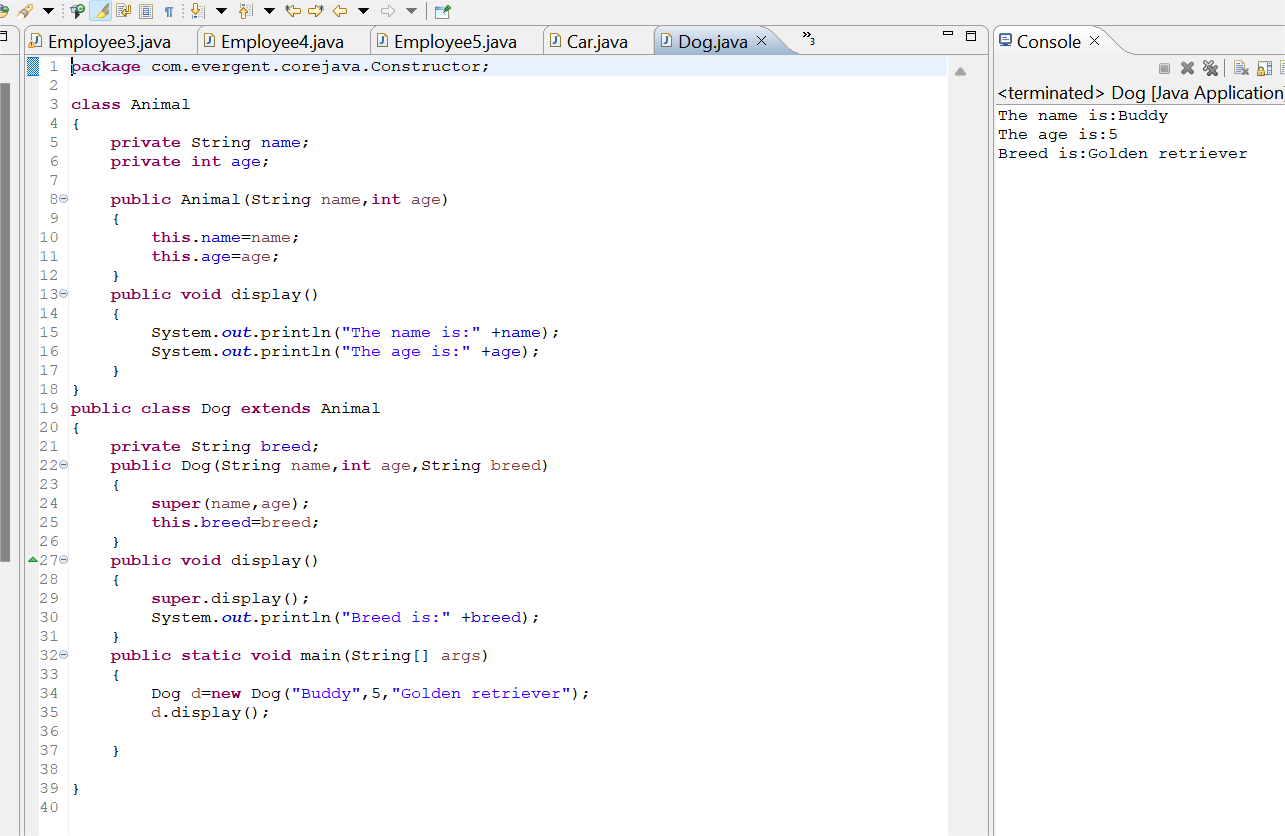
**Program 6:**

****

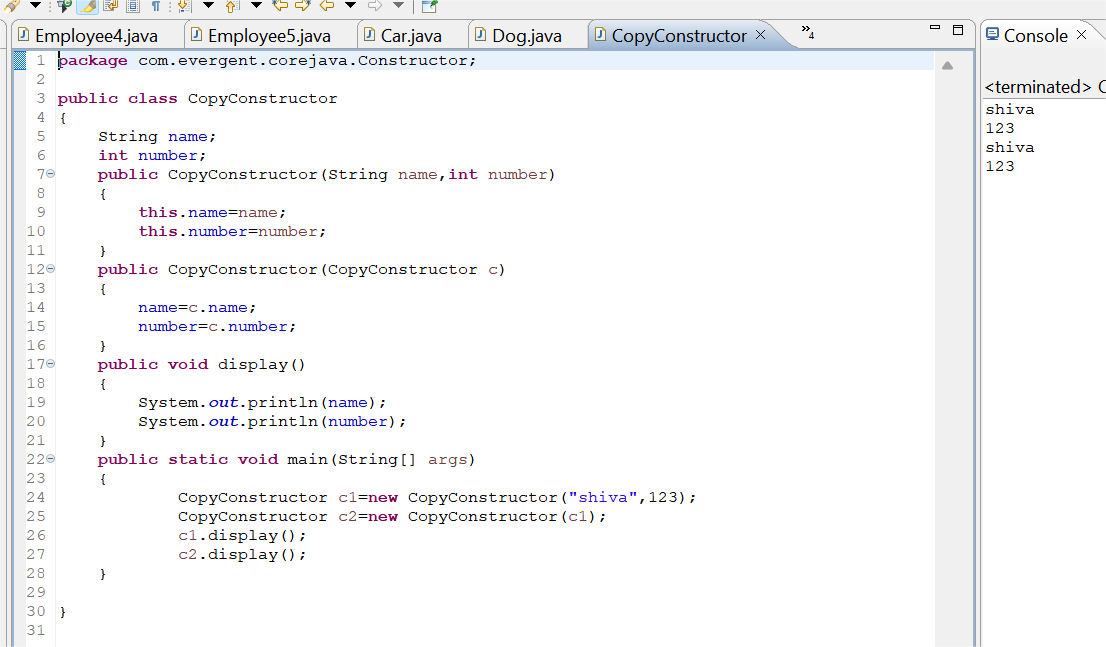
**Program 7:**

****

**Program8:**

****

**Program9:**

****

**Date:09/08/2024 - Day5**

1. **Static**

a. Static is a keyword

b. We can declare variables and methods as static

c. We can access static variables and static methods directly through calssname.methodname and classname.variablename respectively.

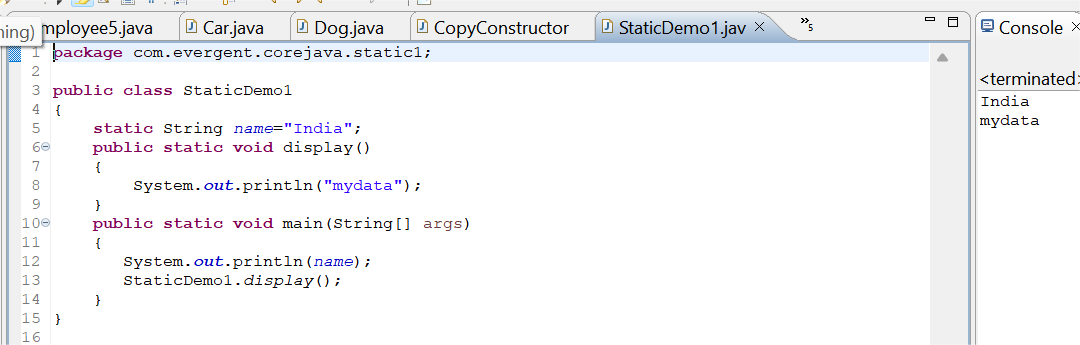
d. Static methods can access static methods and static variables only.

e. Static methods cannot access non static methods and non static variables.

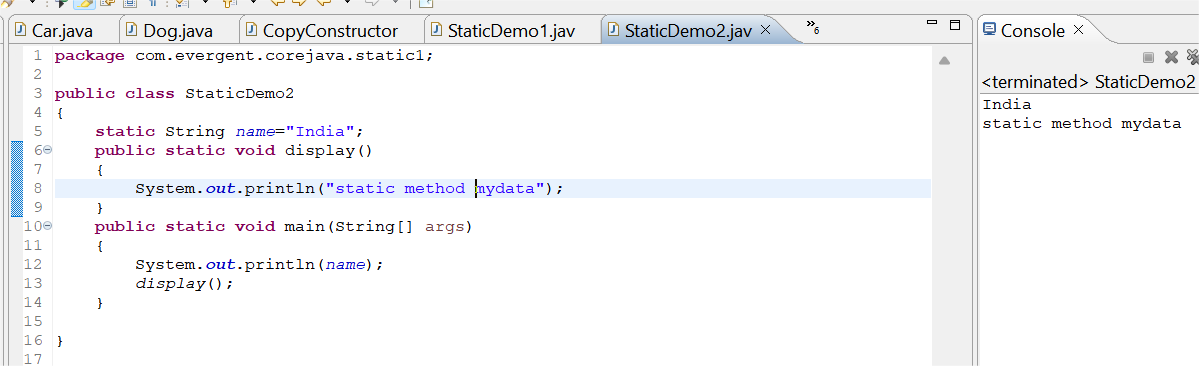
f. Non static methods can access static methods and static variables.

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

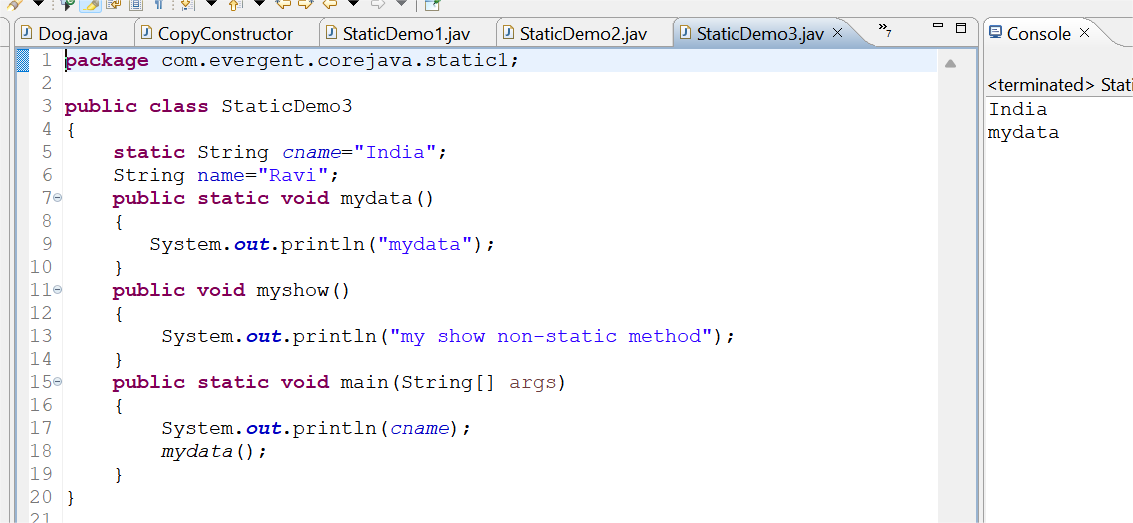
**Program1:**

****

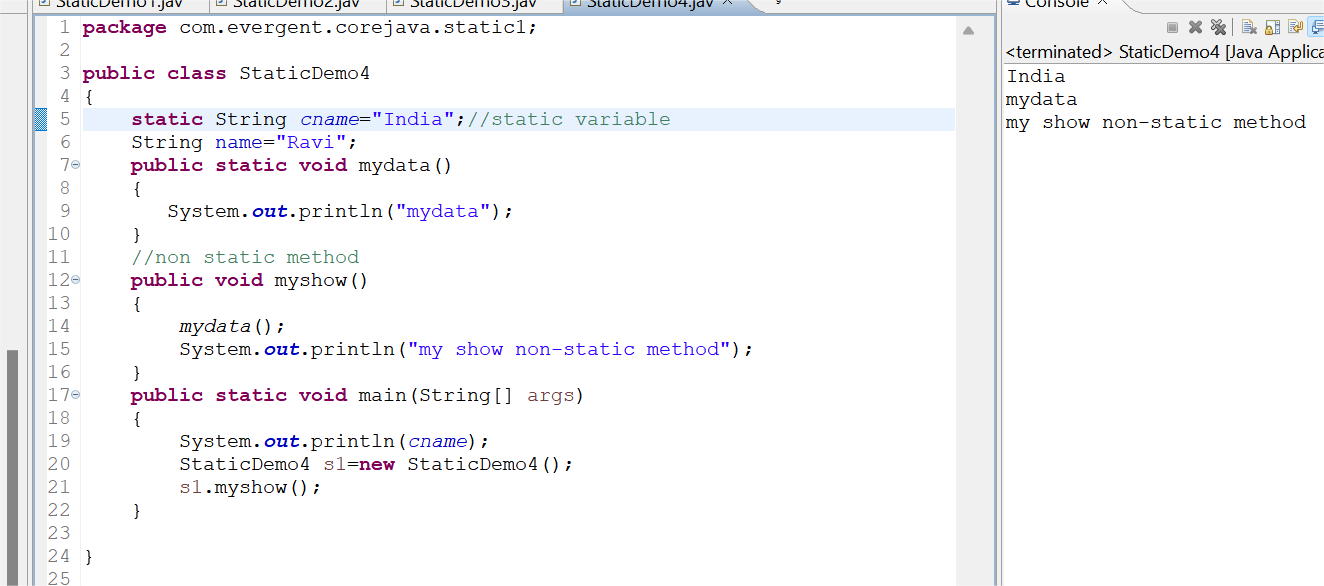
**Program2:**

****

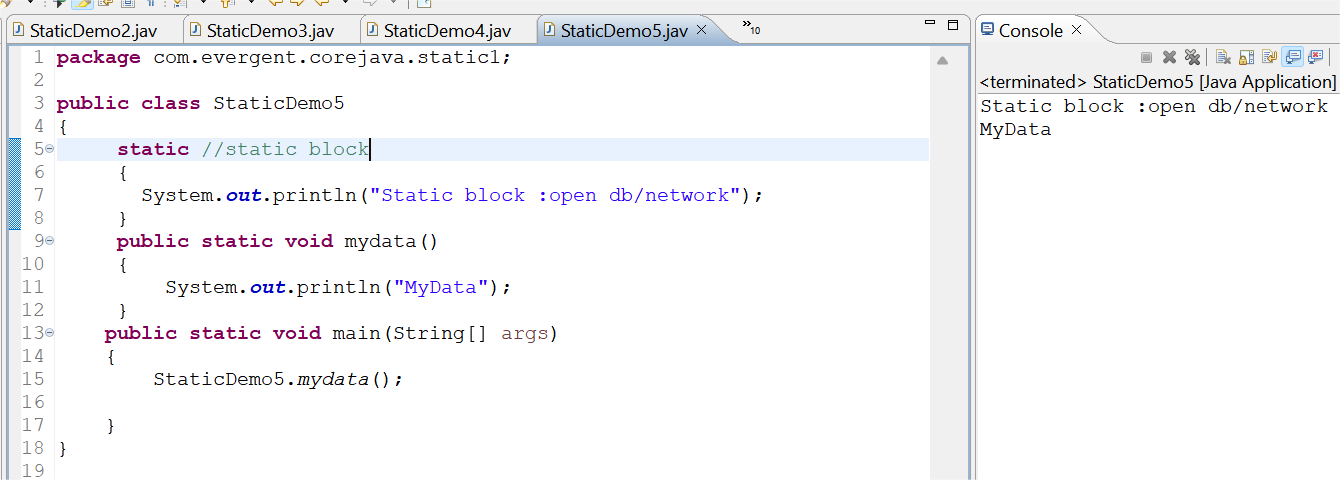
**Program3:**

****

**Program4:**

****

**Program5:**



**Program6:**

****

**2. Final**

a. Final is a Keyword.

b. We can declare a variable, method, or a class as final.

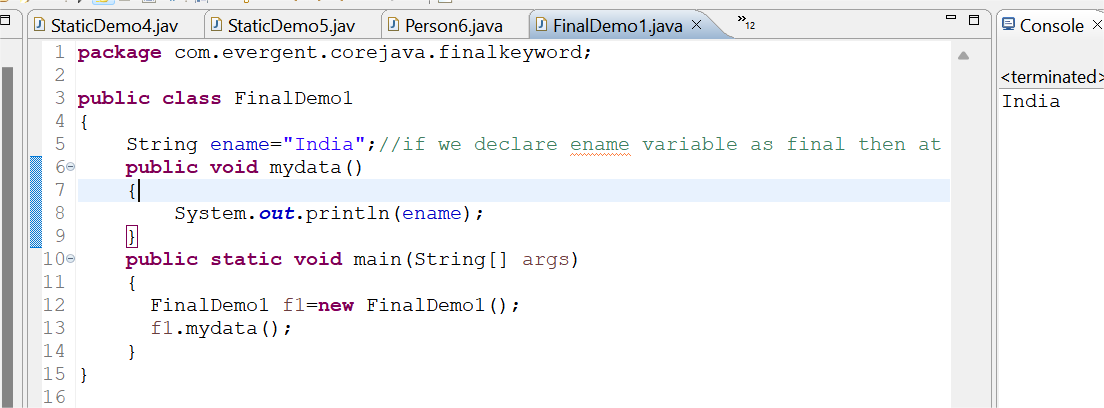
c. Final variable cannot be modified.

d. Final Method cannot be overrided.

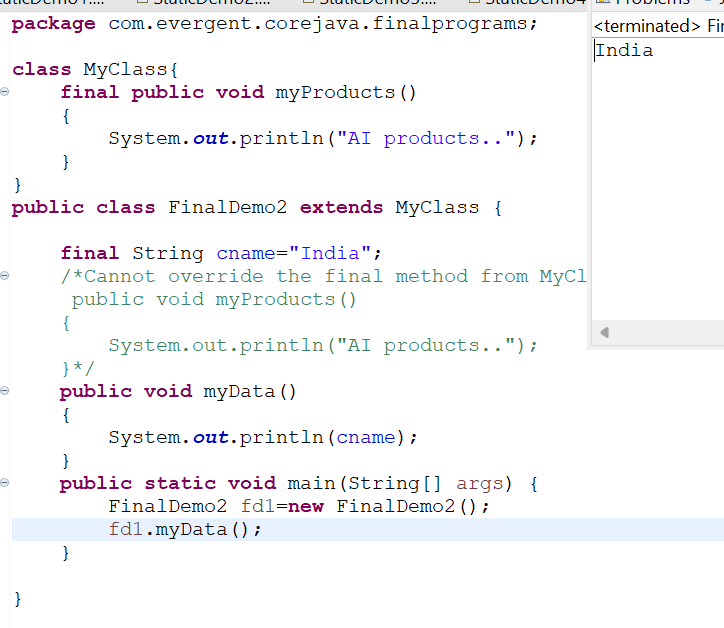
e. Final class cannot be inherited.

d.we can intialize final variables through constructor

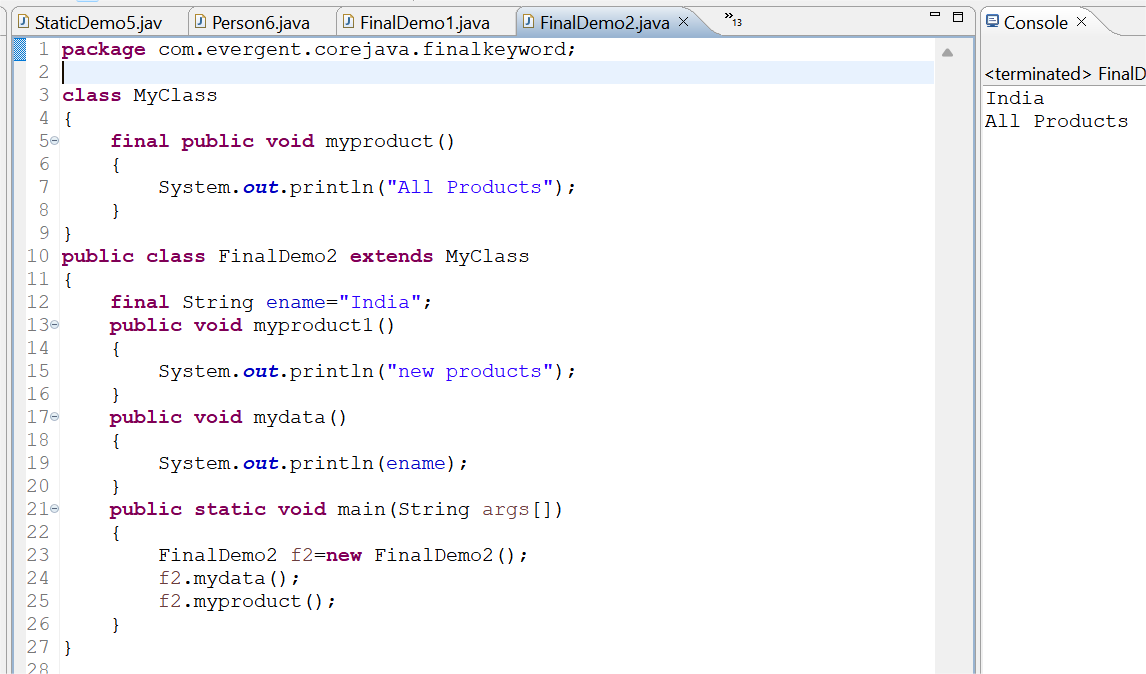
**Program1:**

****

**Program2:**

****

**Program3:**

****

**12/08/24 :**

**Strings:**

**-Why string is immutable?**

1. String is a final class
2. Strings are immutable
3. Strings having methods
4. All methods are non-synchronized

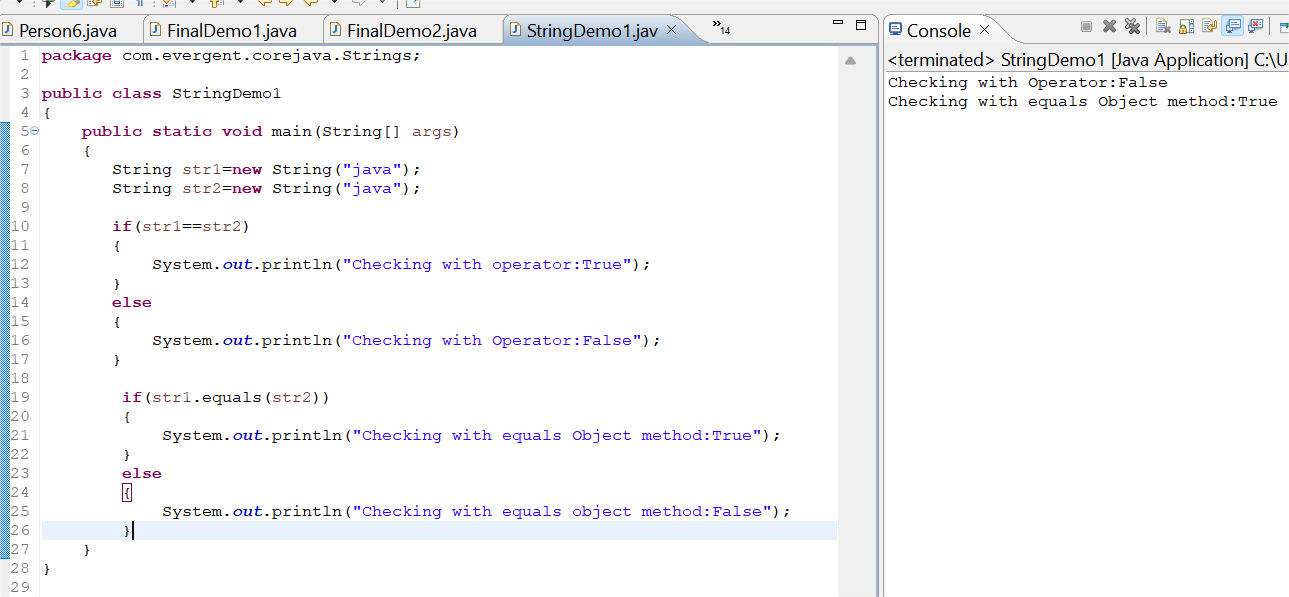
**StringBuffer:**

1. String buffer is a final class
2. String buffer is mutable
3. String buffer having methods
4. All methods are synchronized

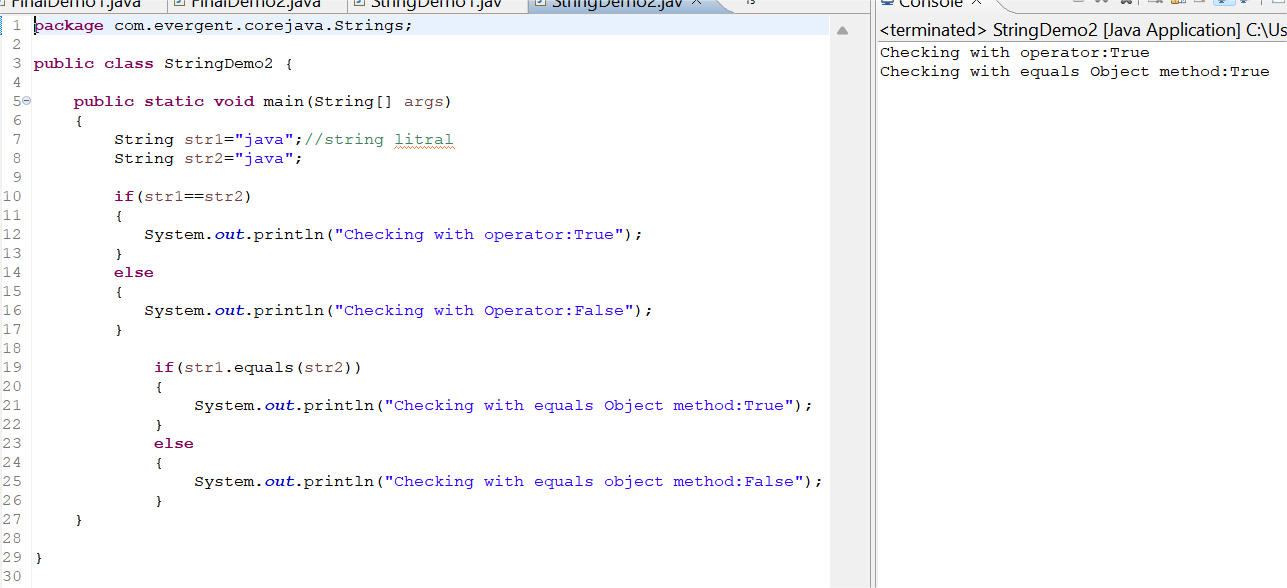
**String Builder:**

1. String builder is a final class
2. String builder is mutable
3. String builder having methods
4. All methods are non-synchronized

Program: using equals() and == operator:



Program: String initialization without using new keyword.



Program:

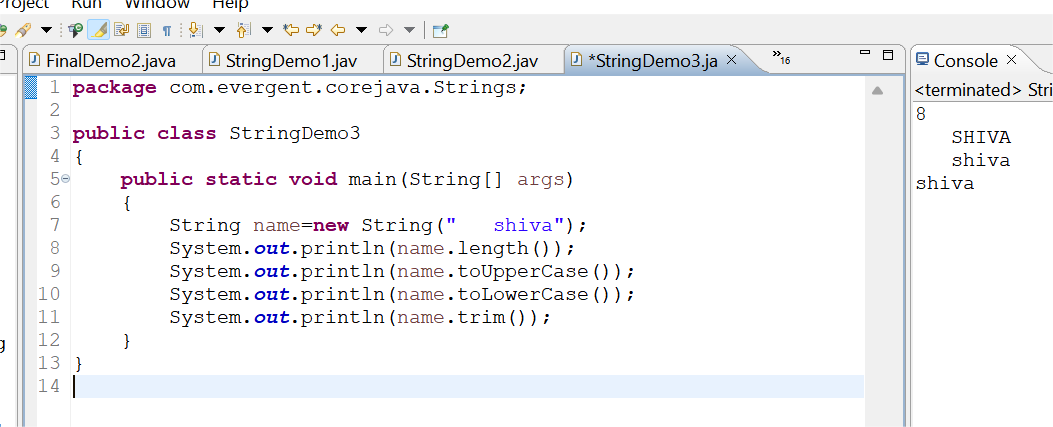
String methods:

-length()

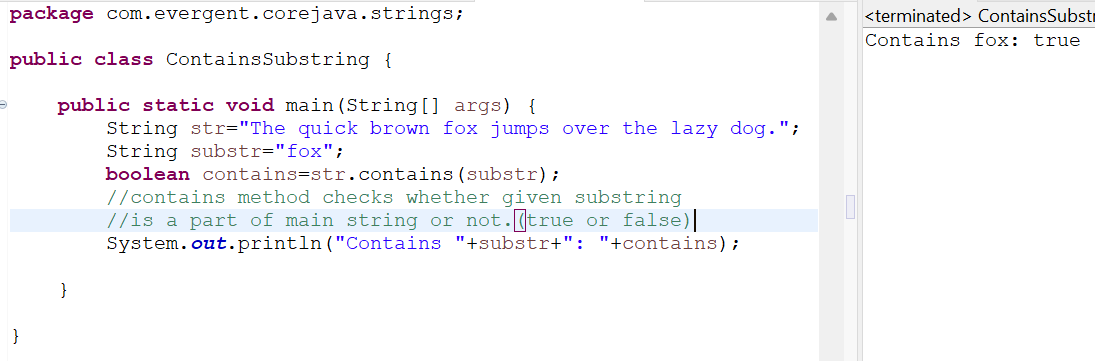
-toLowercase()

-toUpperCase()

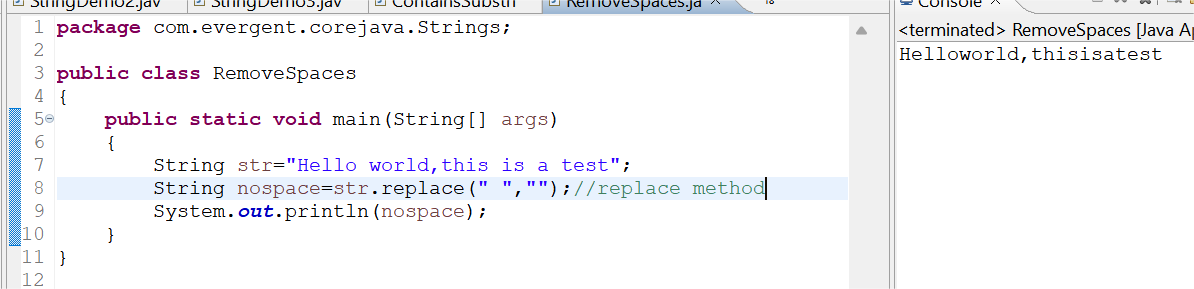
-trim()



1.Create a java program that creates a string and checks if it contains specific substring and then prints out the result.

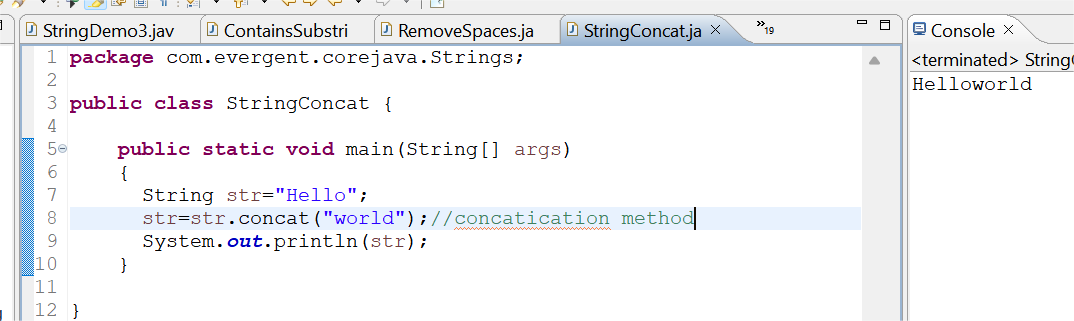


1. Write a java program to create a String ,remove all spaces from the string and then print out the result.

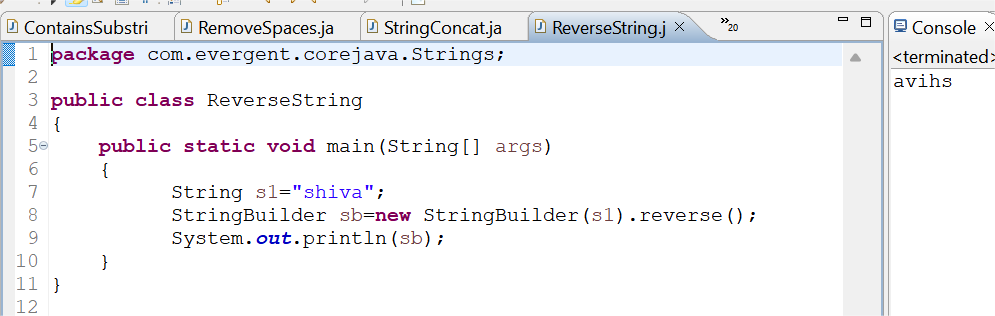


**String concatenation:**

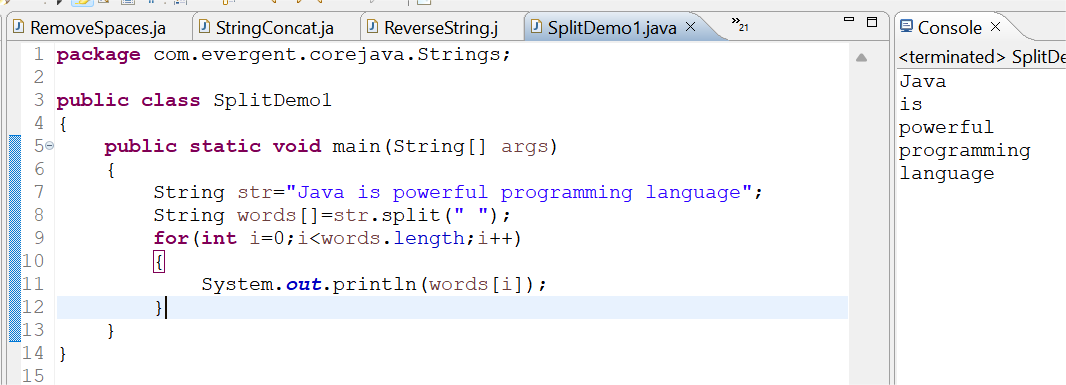
-Strings can be concatenated using + operator (or) concat() method.



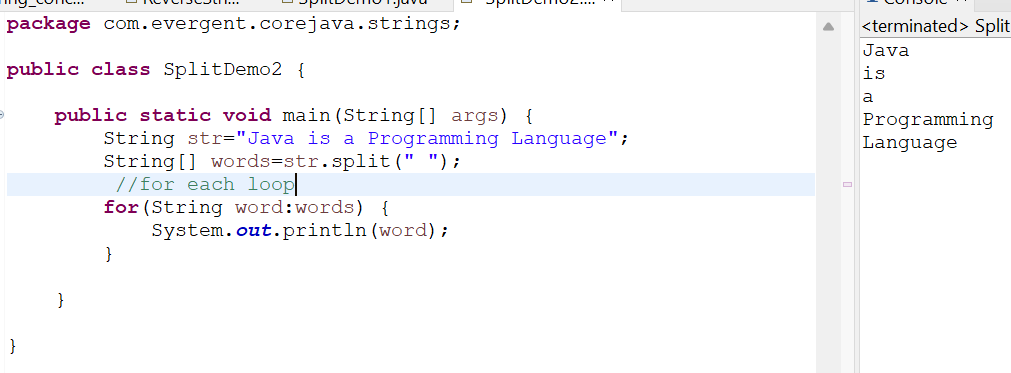
Reverse of a String:



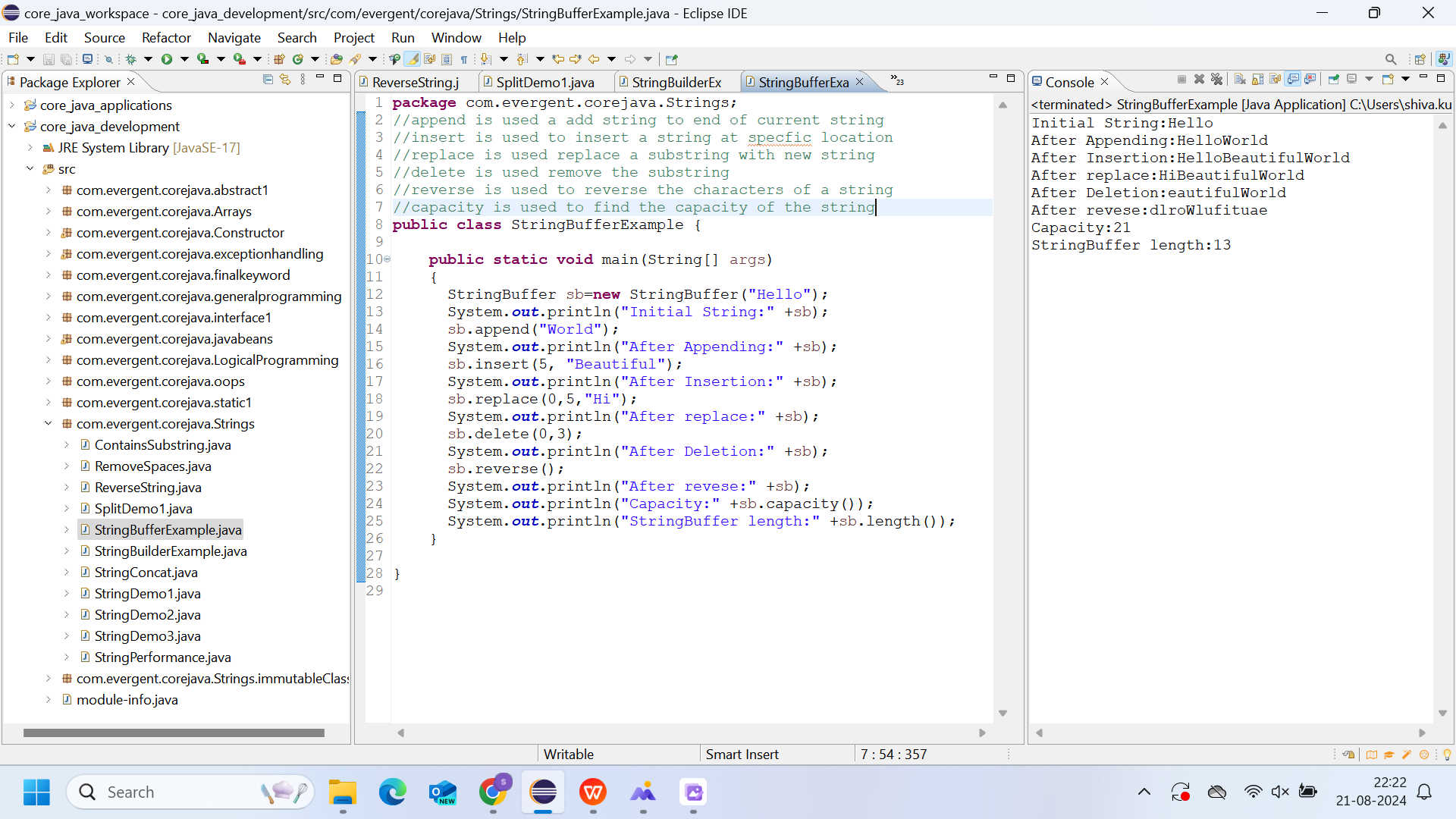
Program: split given string by white spaces.



Using for each loop:



StringBuffer methods:



StringBuilder Methods:

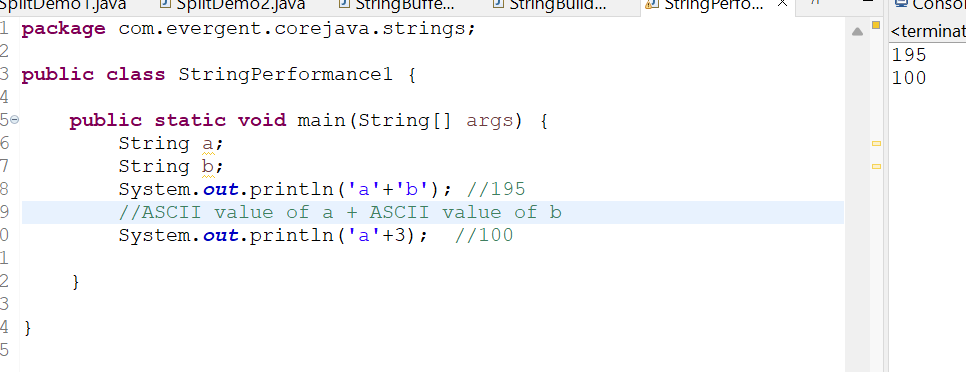


**String class important points:**

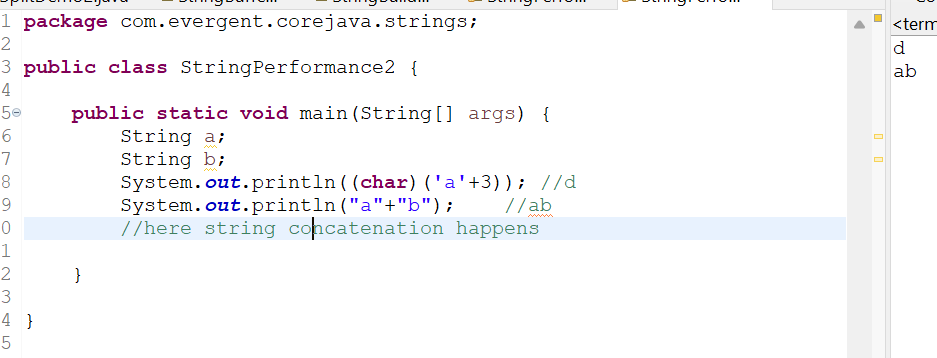
1. In java a string is a sequence of characters ,often used to represent text.
2. Strings are objects in java and are instances of the string class,which is part of the java java.lang package
3. Key features of strings in java:
4. Immutable:once a string object is created ,it cannot be changed .
5. Java optimizes memory usage by storing strings in special area of memory as string pool
6. If two strings have the same value and are created without using new keyword they will refer to same object in the string pool.
7. We can create a string in java in multiple ways:
8. Using string literals :str=”hello world”;
9. Using the new keyword

String str=new String(“hello, world”);

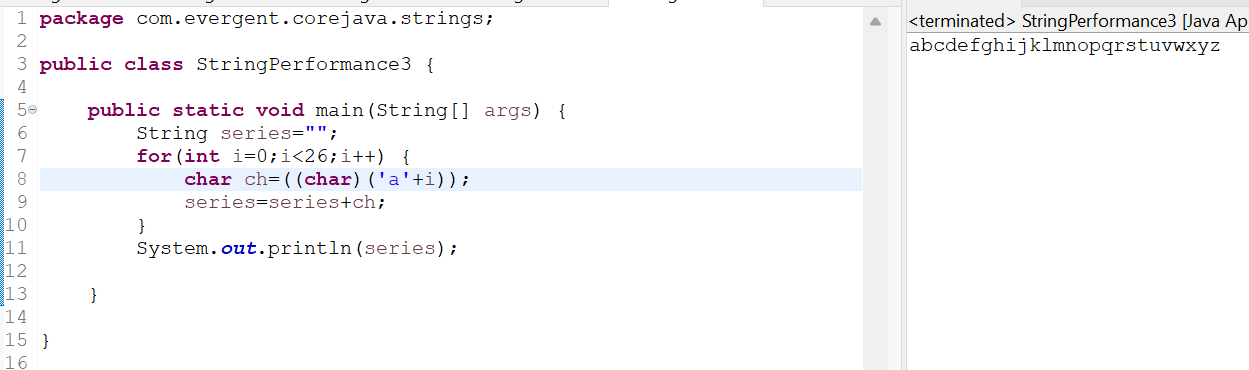
String Performance1:



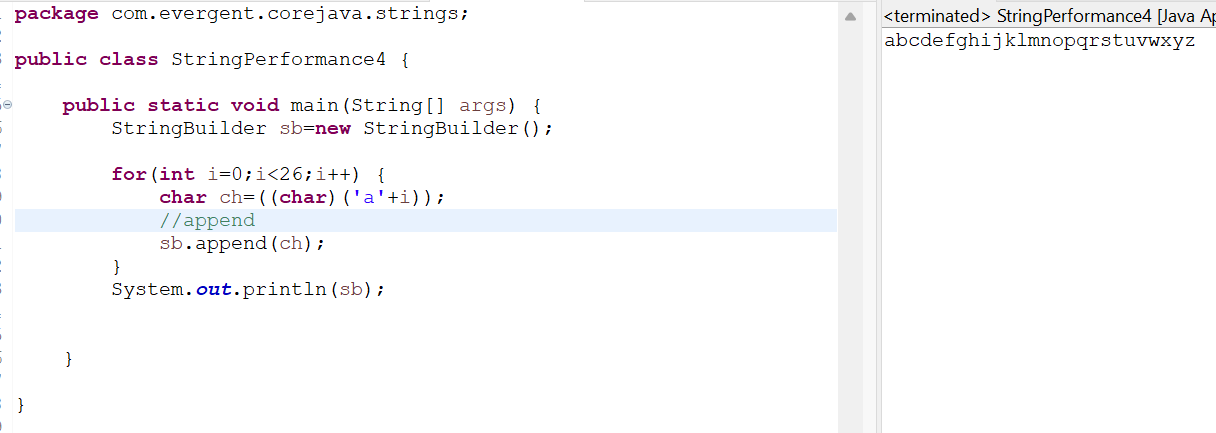
String Perfomance2:



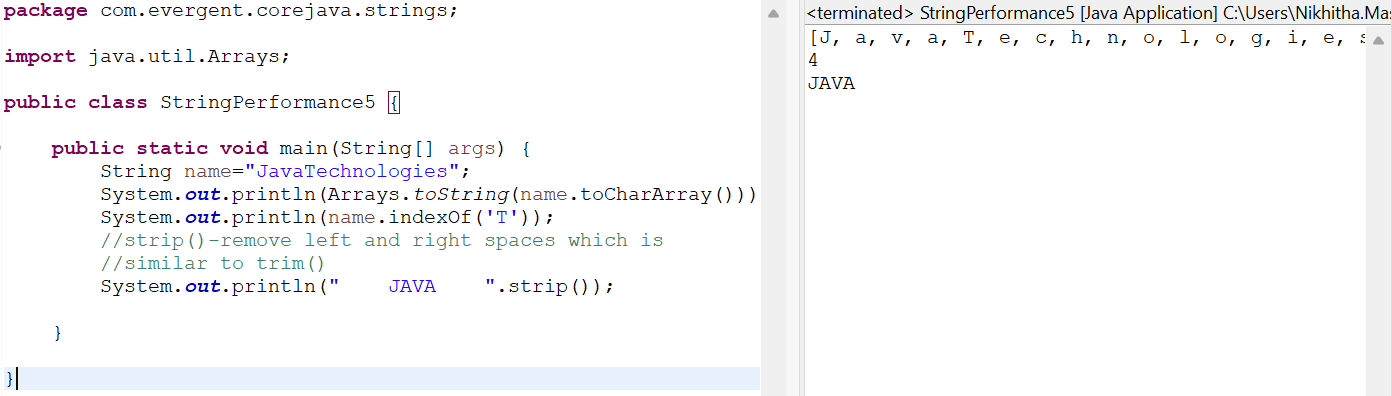
String Performance3:



String performance4:



String Performance5:



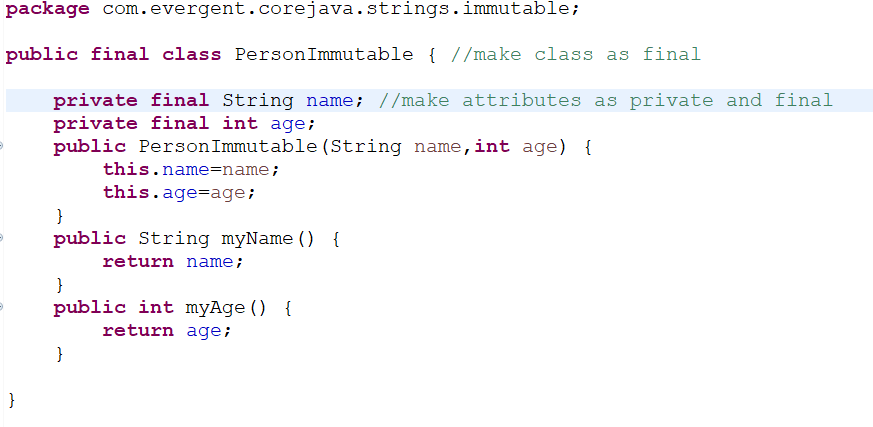
**13/08/2024- DAY7**

**-** final variables also initializes while constructor calling.

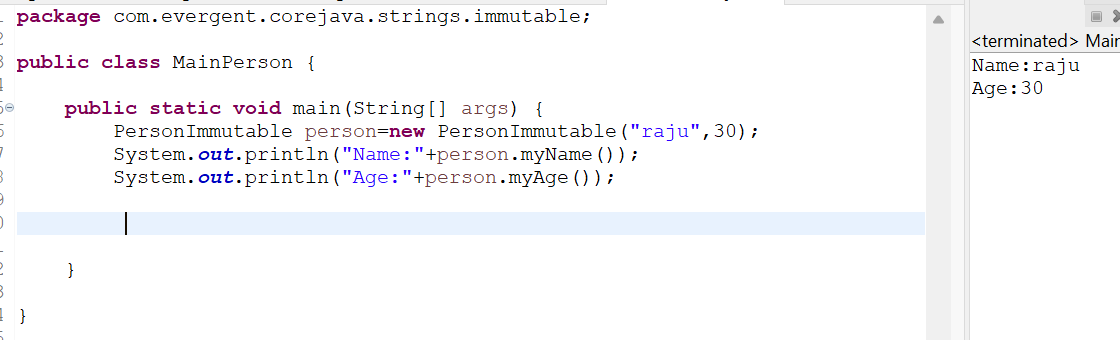
1. Can you make a class as Immutable?

Yes, we can declare our class as a final class and attributes as private and final.

Program:



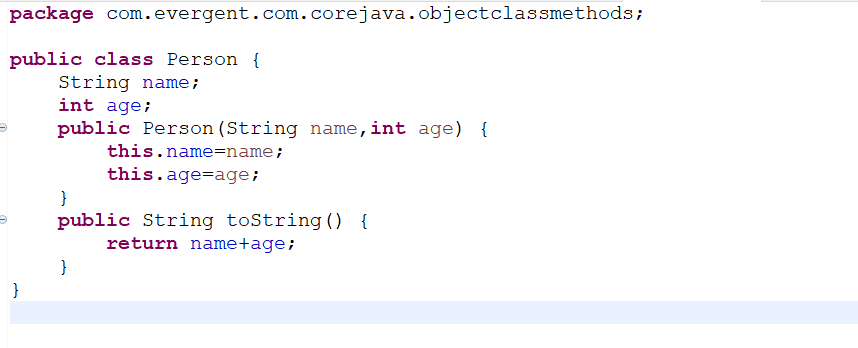
MainPerson class:



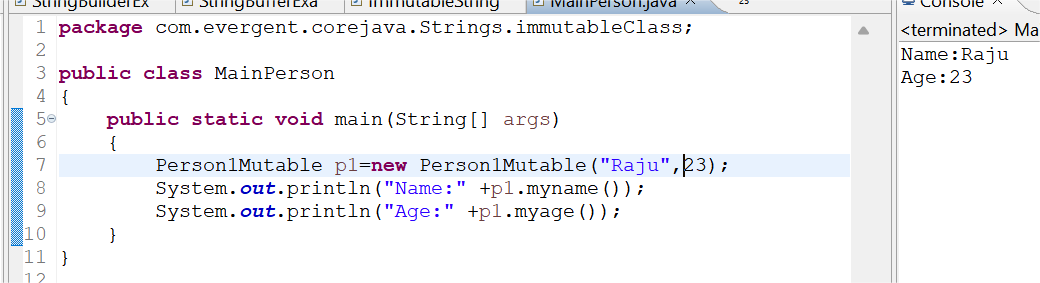
Object class methods:

-toString()

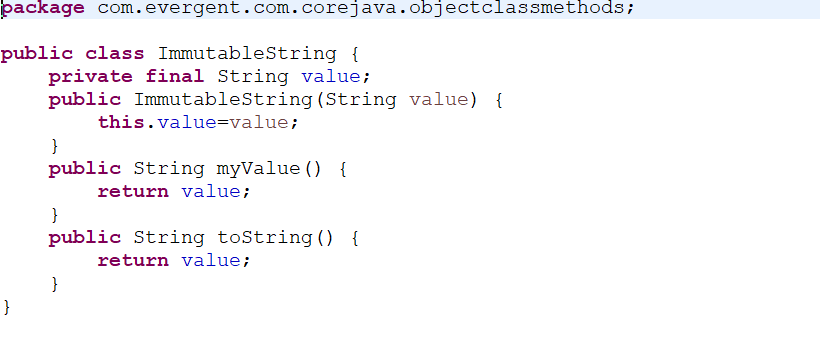
-hashCode();



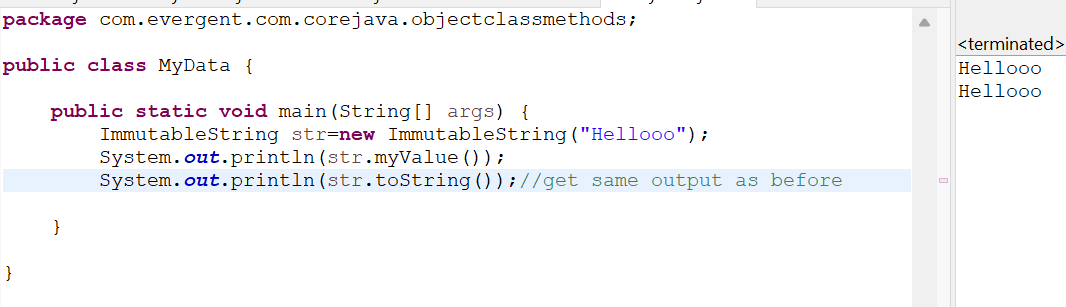
MyPerson class:



ImmutableString:



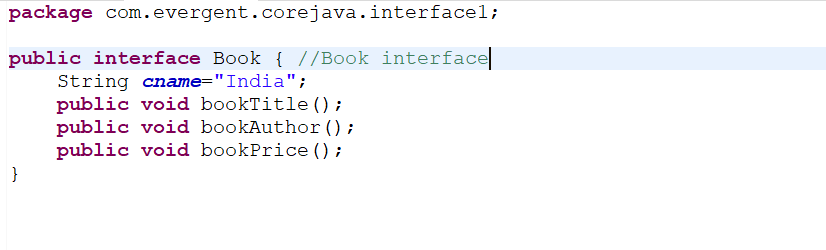
MyData class:



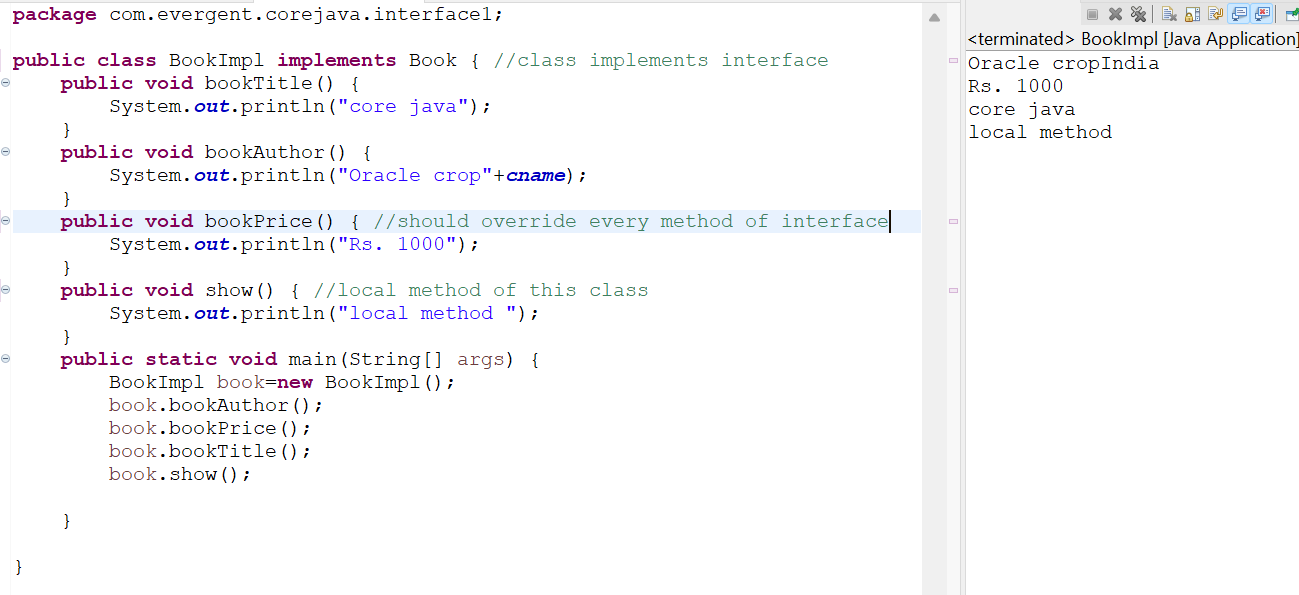
**Interfaces:**

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.

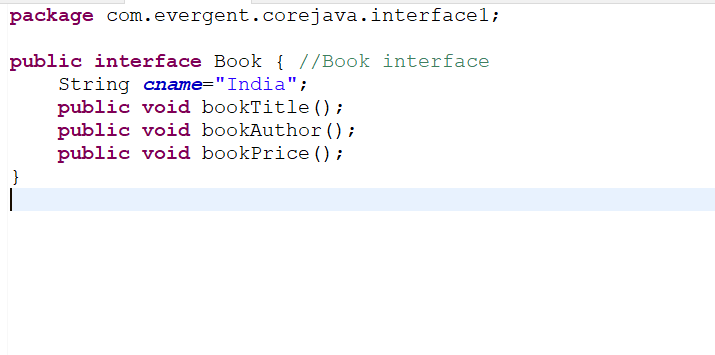
Program: Book interface

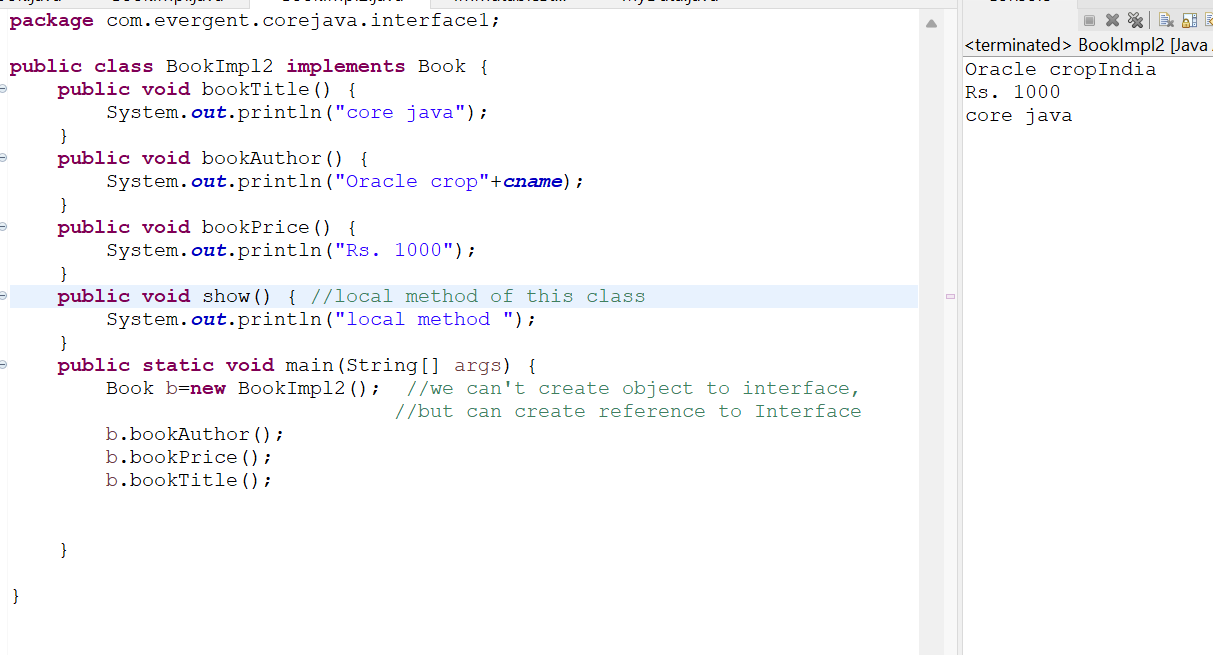


BookImpl implements Book:



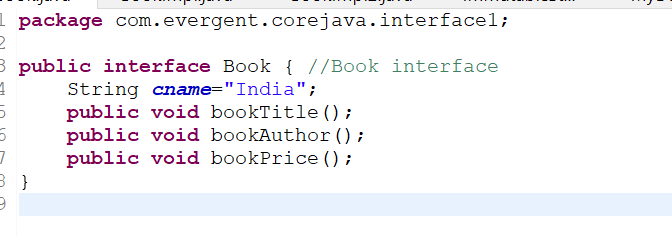
Program: Book Interface



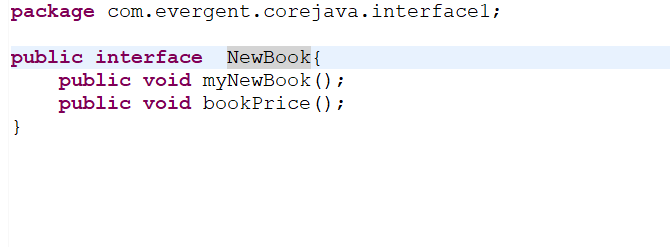


Program:

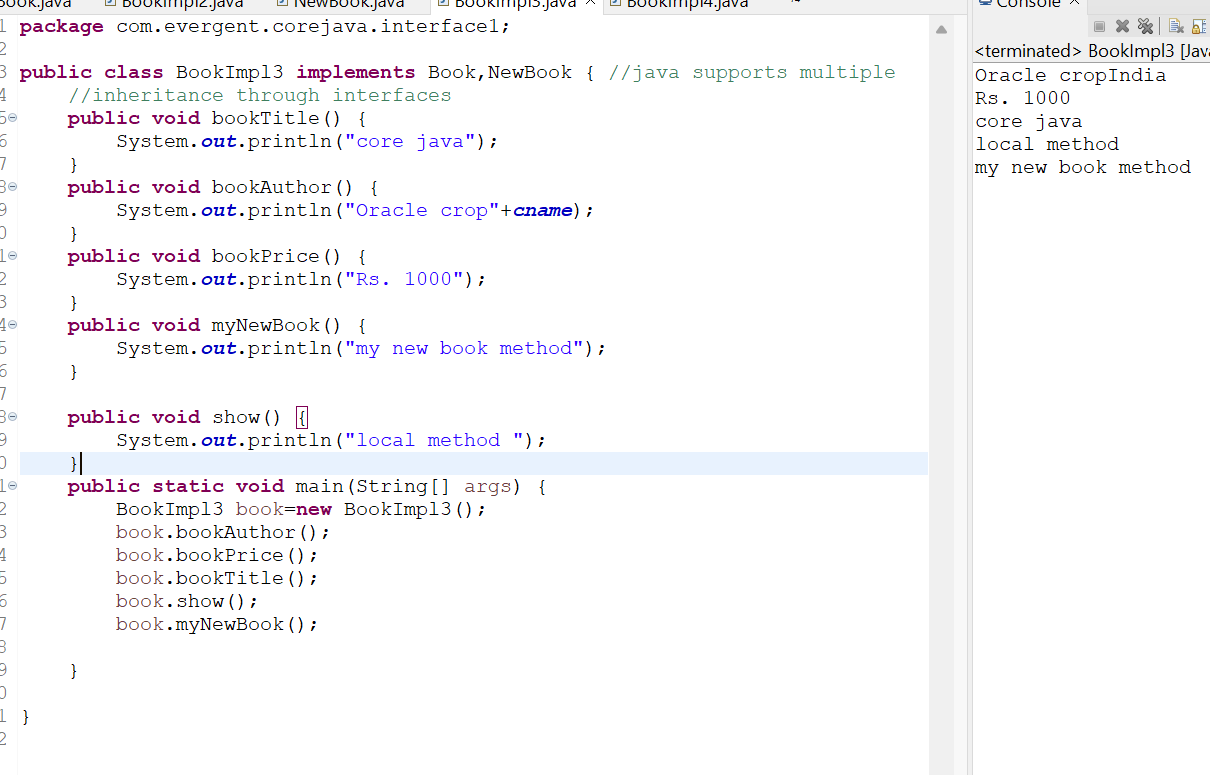
Book Interface:



NewBook Interface:

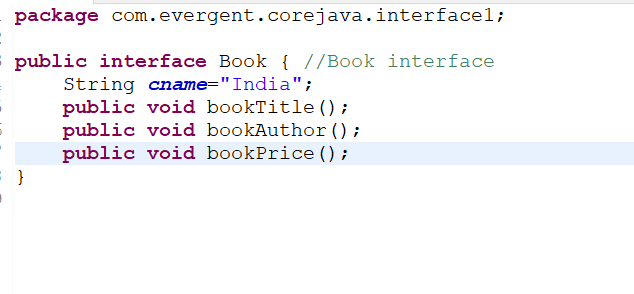


BookImpl3 implements Book, NewBook:

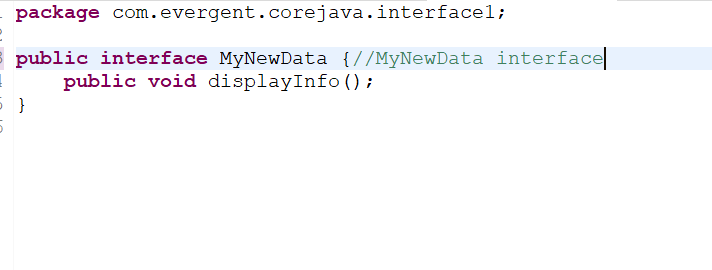


Program:

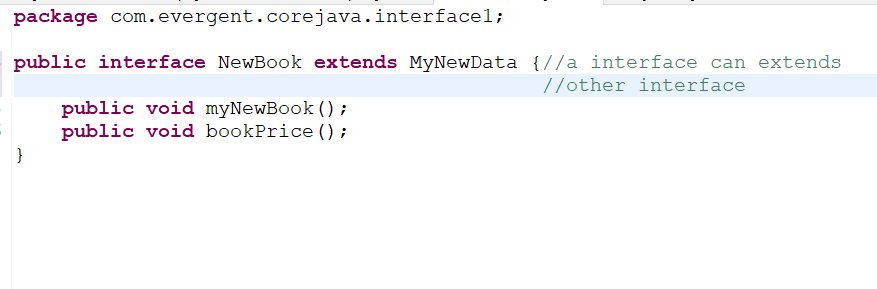
Book Interface:



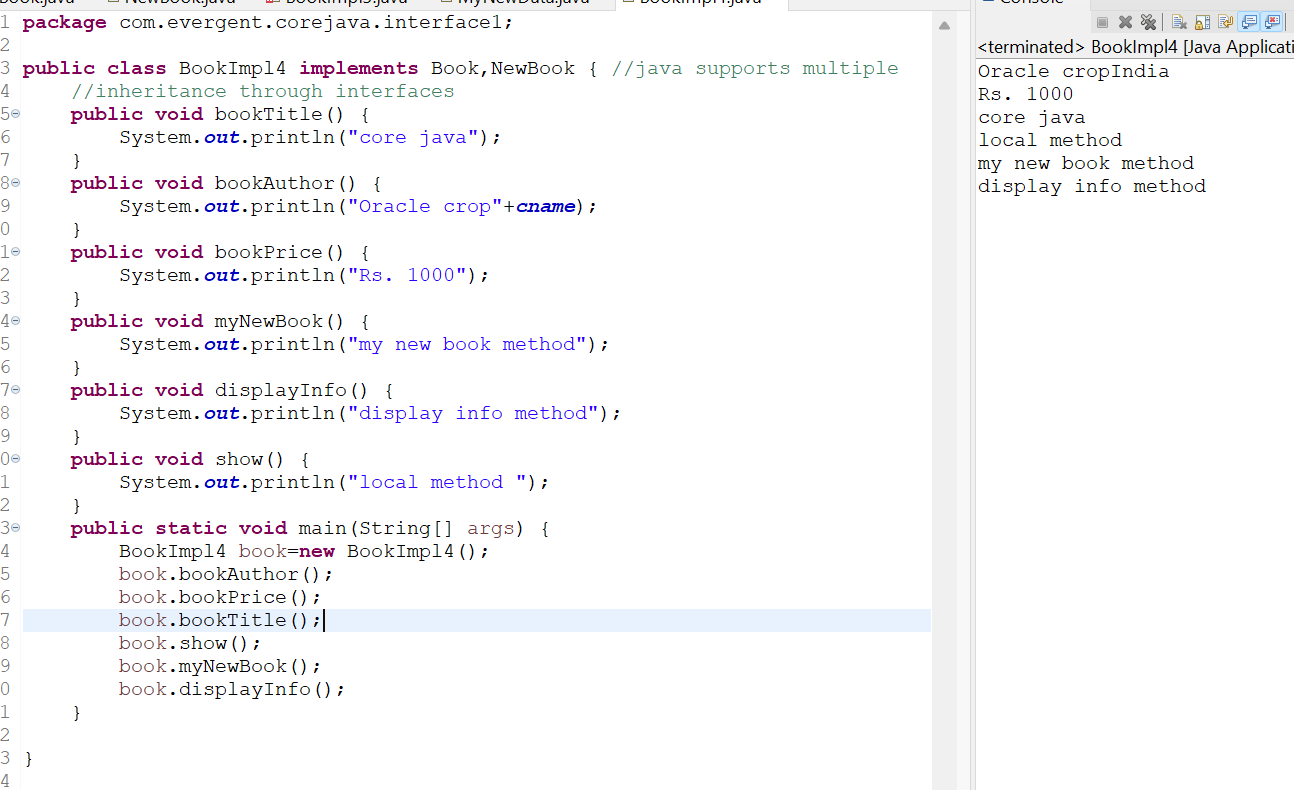
MyNewData interface:



NewBook interface extends MyNewData interface:



BookImpl4 implements Book,NewBook:

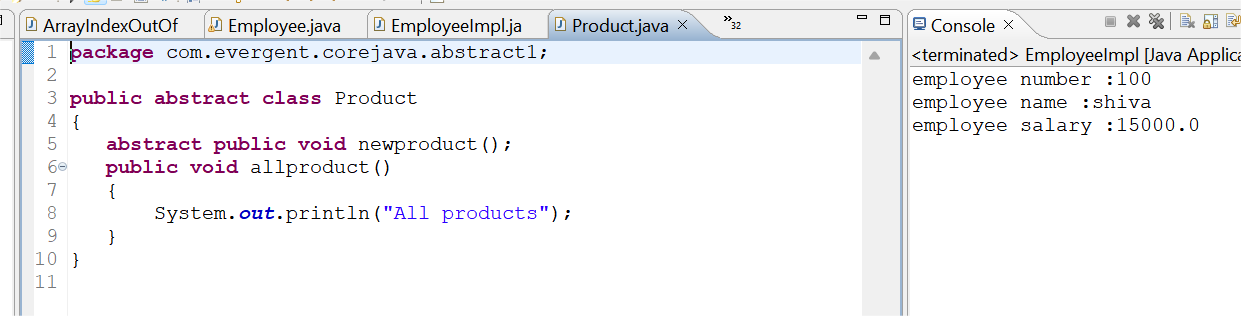


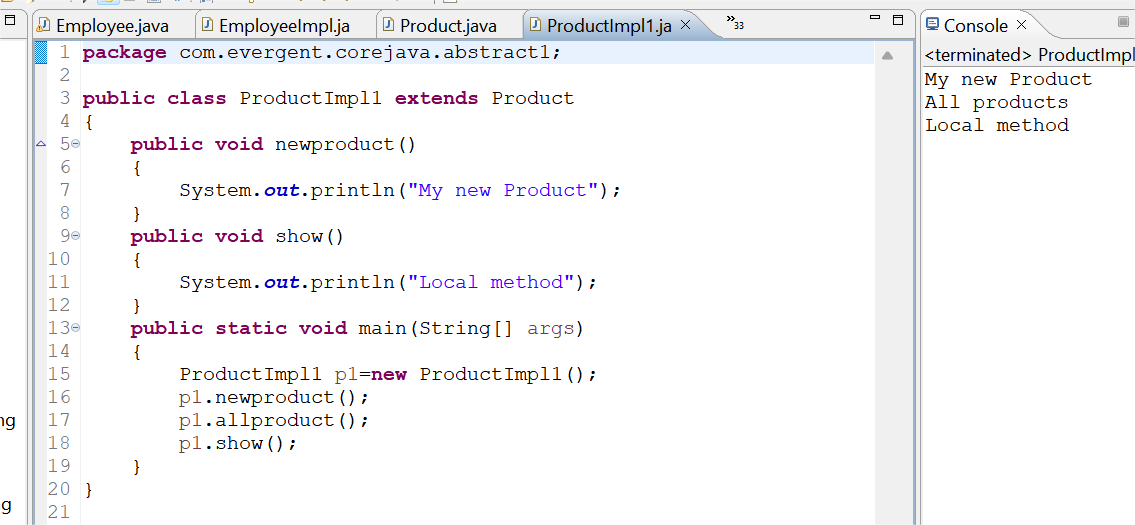
**Day 8: 14-Aug-2024:**

**Abstract class:**

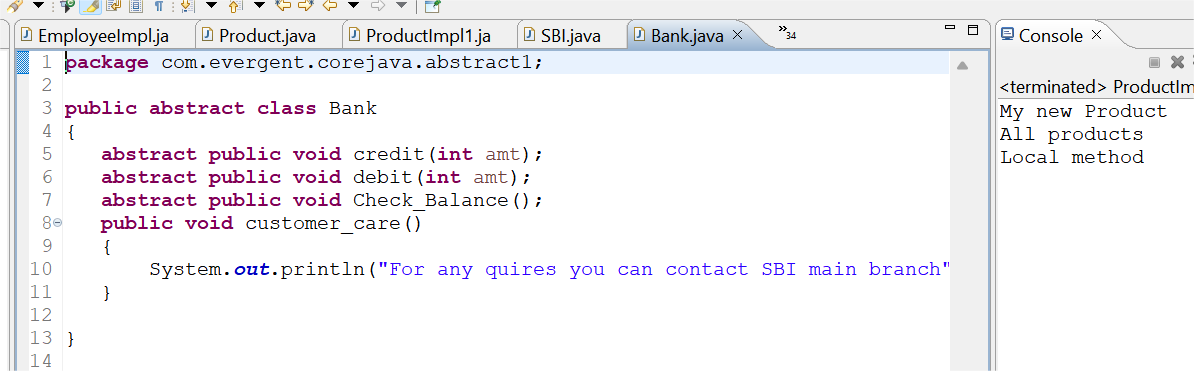
* Abstract is a keyword.
* Abstract class can have abstract methods and concrete methods.
* If any class having one abstract method that class should be declared as a abstract keyword, otherwise that class will be showing compile time error.
* If any class extends abstract class that class should override all abstract methods, otherwise the class will be showing compile time error.
* We cannot create object to abstract class but we can create reference to abstract class.
* We have to declare abstract methods explicitly.
* We can create constructor to abstract class.
* We can access abstract class constructor through the sub-class object creation.

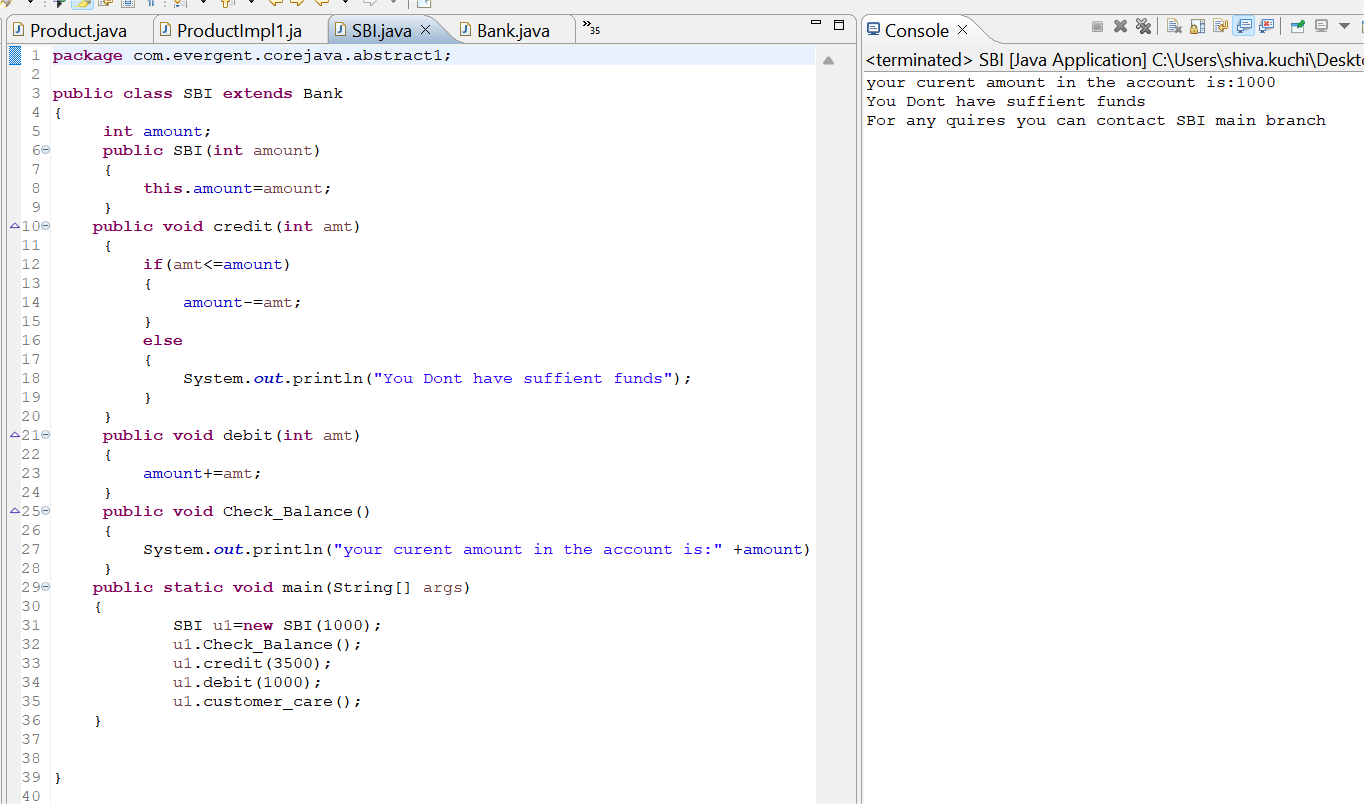
Program 1:





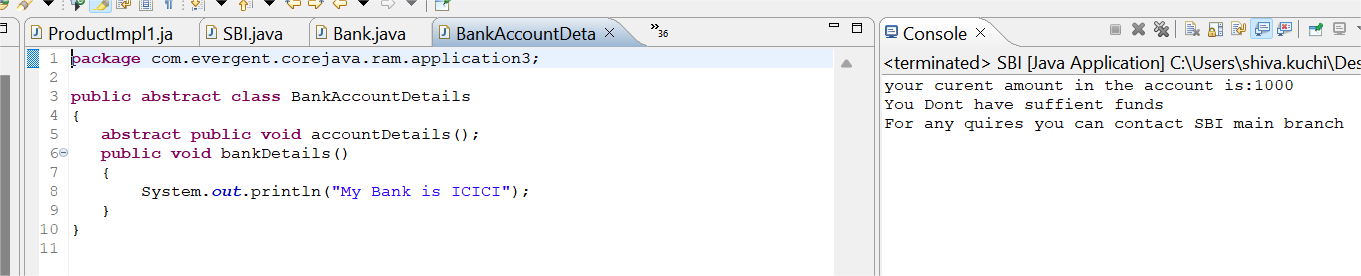
Own Program on abstraction

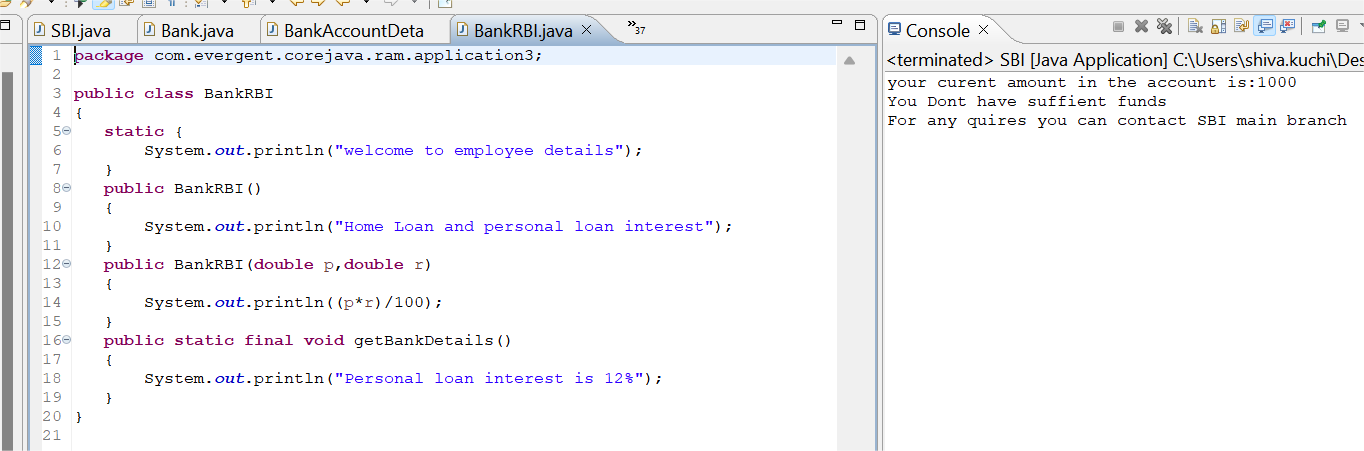


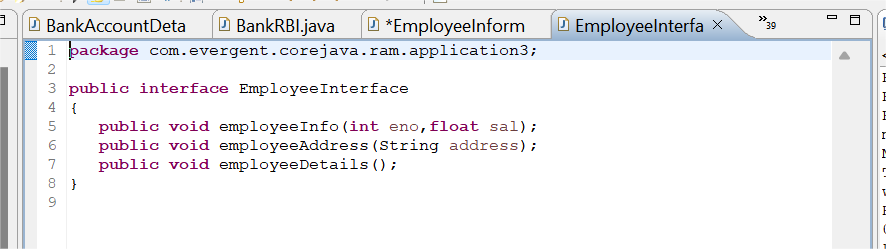


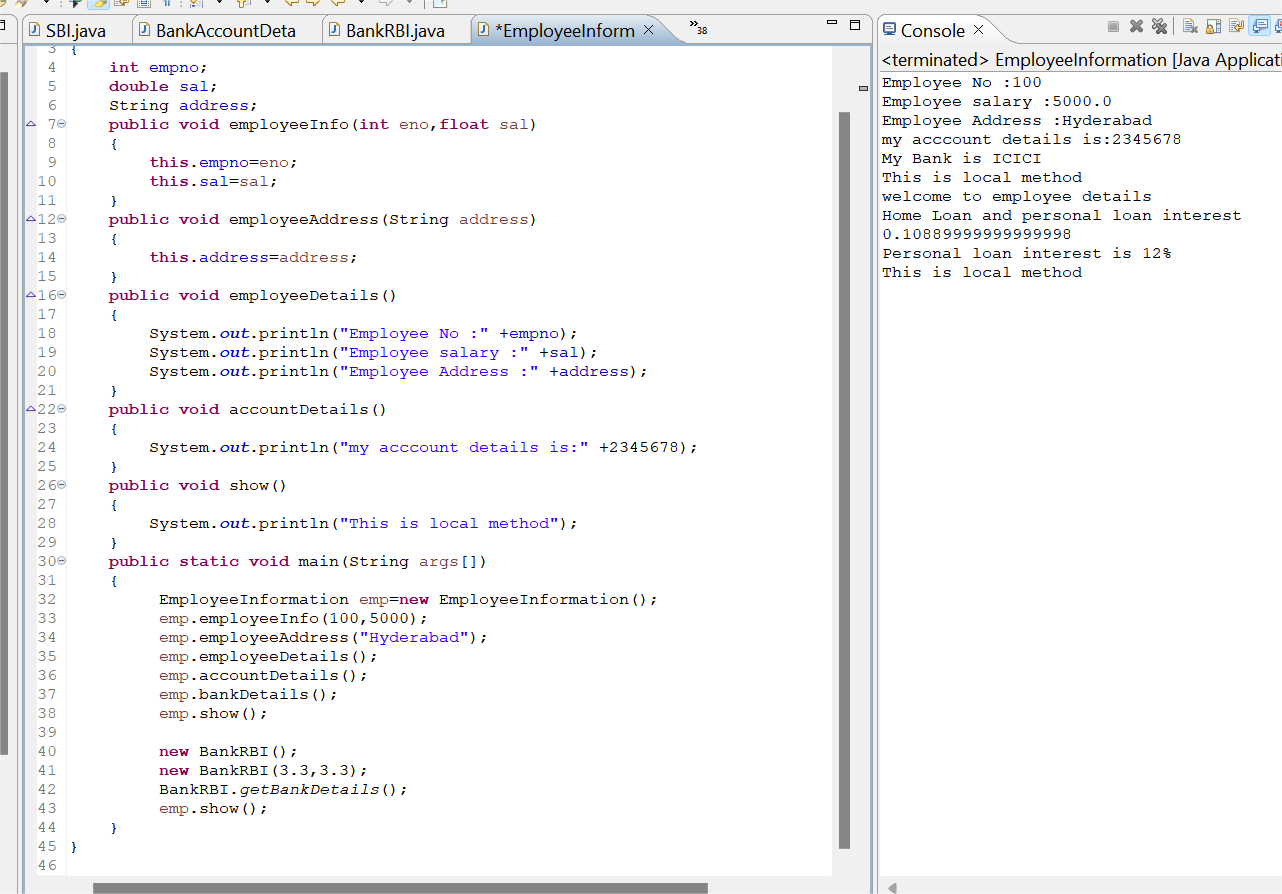
**Day 9: 16-Aug-2024:**

1. Java Concept Test
2. Application design and code

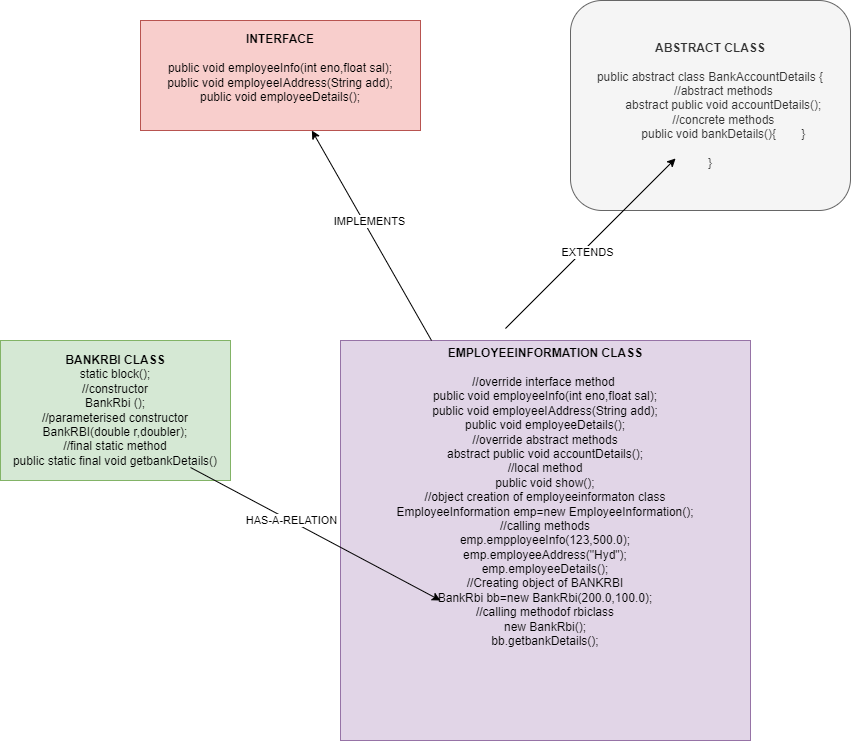
****

****

****

****

Flow Diagram

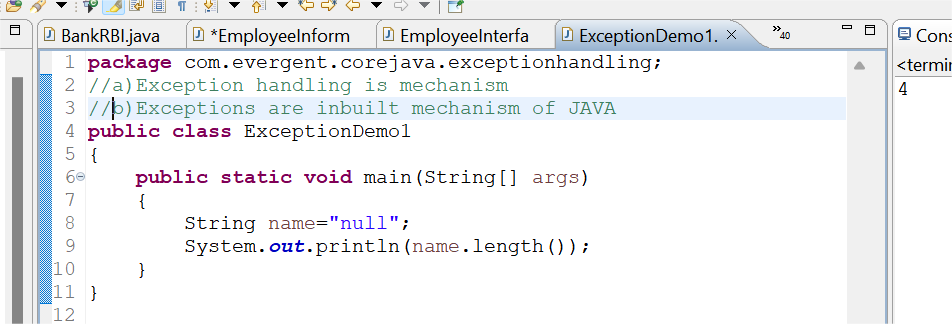


**Day 10: 19-Aug-2024 and Day 11: 20-Aug-2024:**

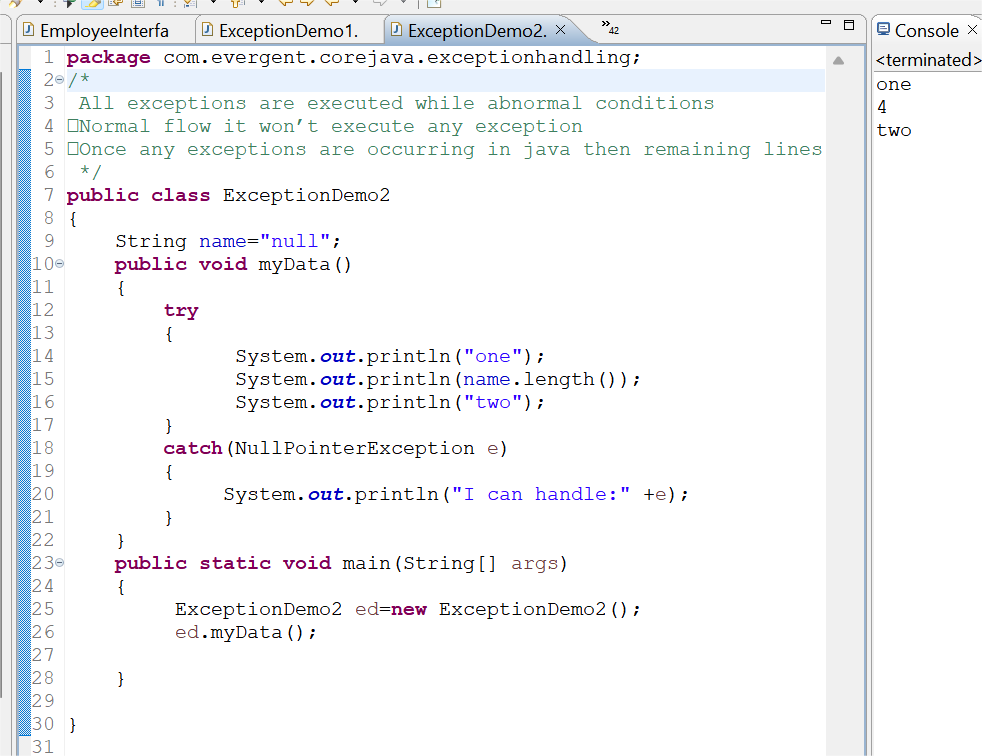
1. **Exception Handling:**

* Exception handling is mechanism
* Exceptions are inbuilt mechanism of JAVA
* All exceptions are executed while abnormal conditions
* Normal flow it won’t execute any exception
* Once any exceptions are occurring in java then remaining lines of code are unreachable.
* Java.lang.Throwable is super class for exception and error
* There are two types of exceptions in JAVA:
  + 1. Checked exception
    2. Unchecked exception
* All checked exceptions are compile time exceptions
* All unchecked exceptions are run time exceptions
* There are 5 keywords in exception handling:
  + 1. try
    2. catch()
    3. finally()
    4. throws
    5. throw
* Try is for business logic
* Catch is for handling exceptions
* Finally block, is executed if exception occurs or not
* Throws an exception will be executed method by method
* Throw is for run time exceptions & will call predefined exceptions
* Try followed by either catch or finally block
* We should follow exception hierarchical
* We can create our own (user-defined) exceptions
* Our own exceptions extends exception or run time exception
* All exception classes are in to java.lang package
* There is two exceptions in class, developer should be handle 1st exception then after 2nd exception will be handled.
* Errors cannot be controlled

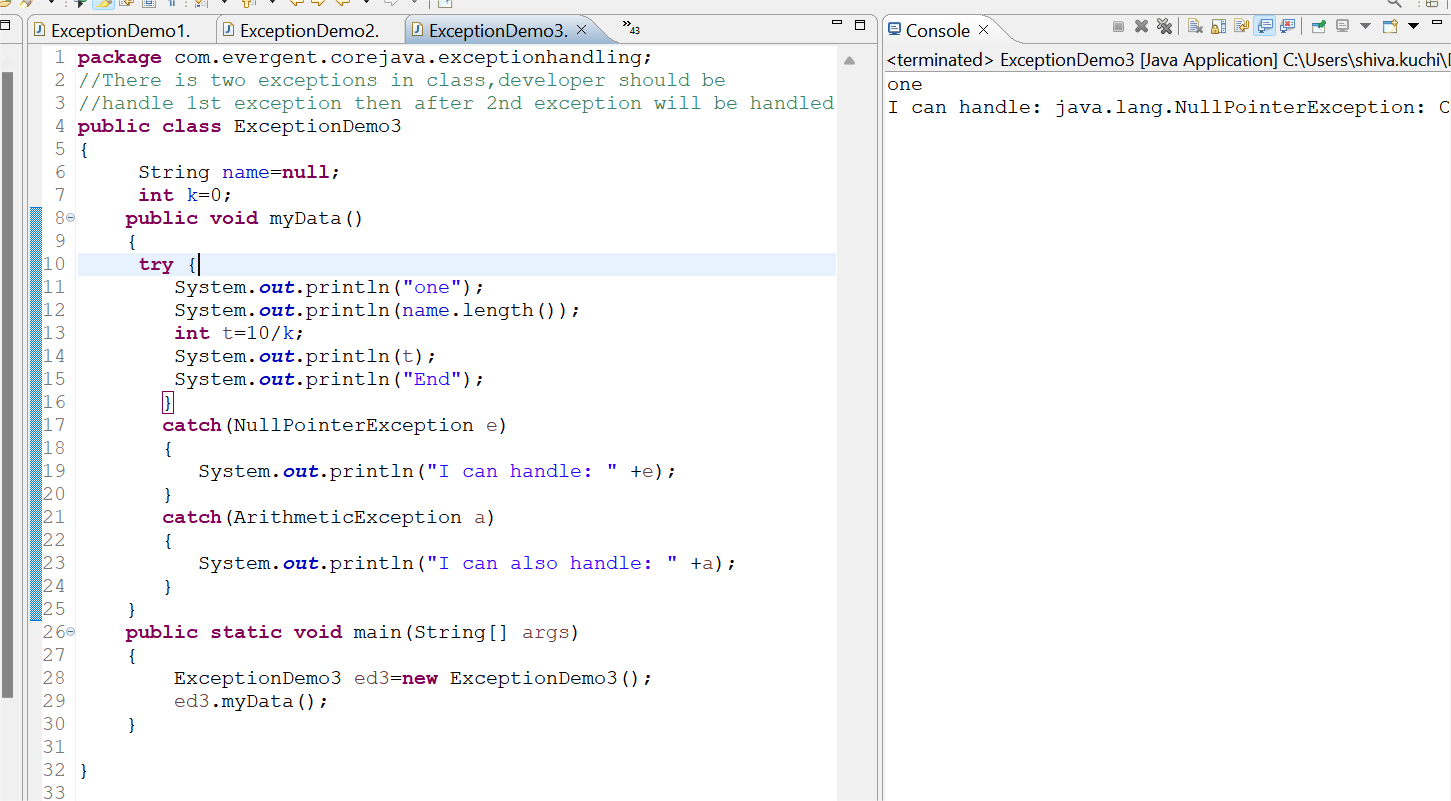
Program 1



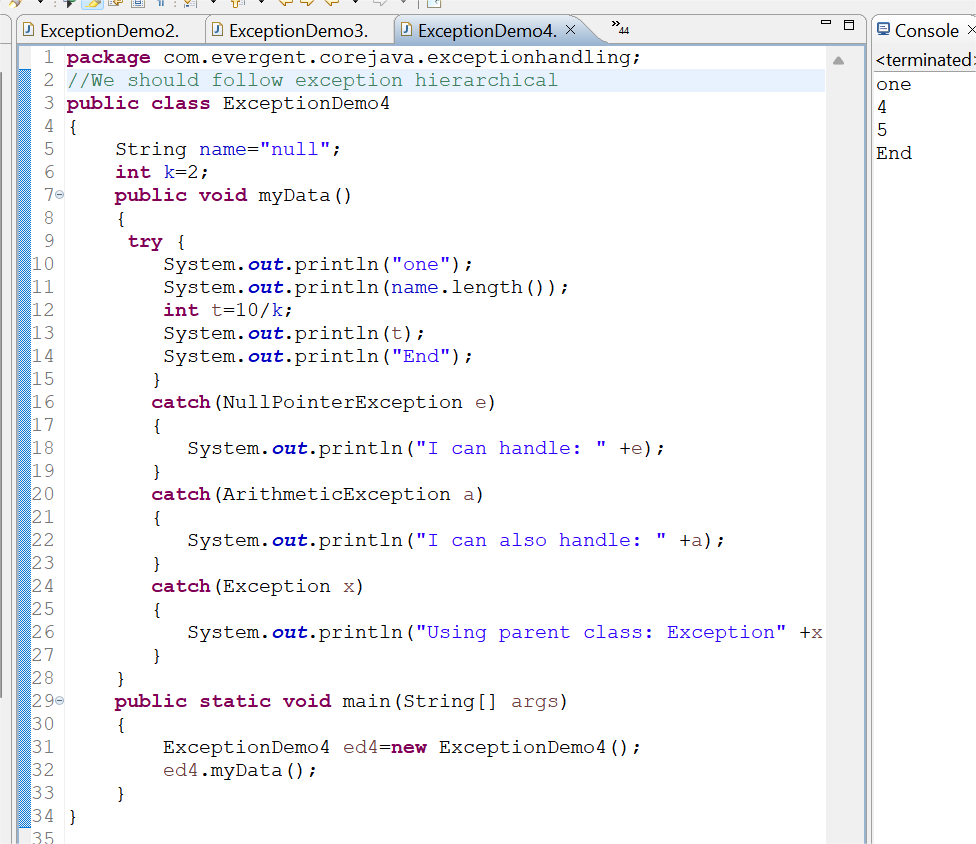
Program 2



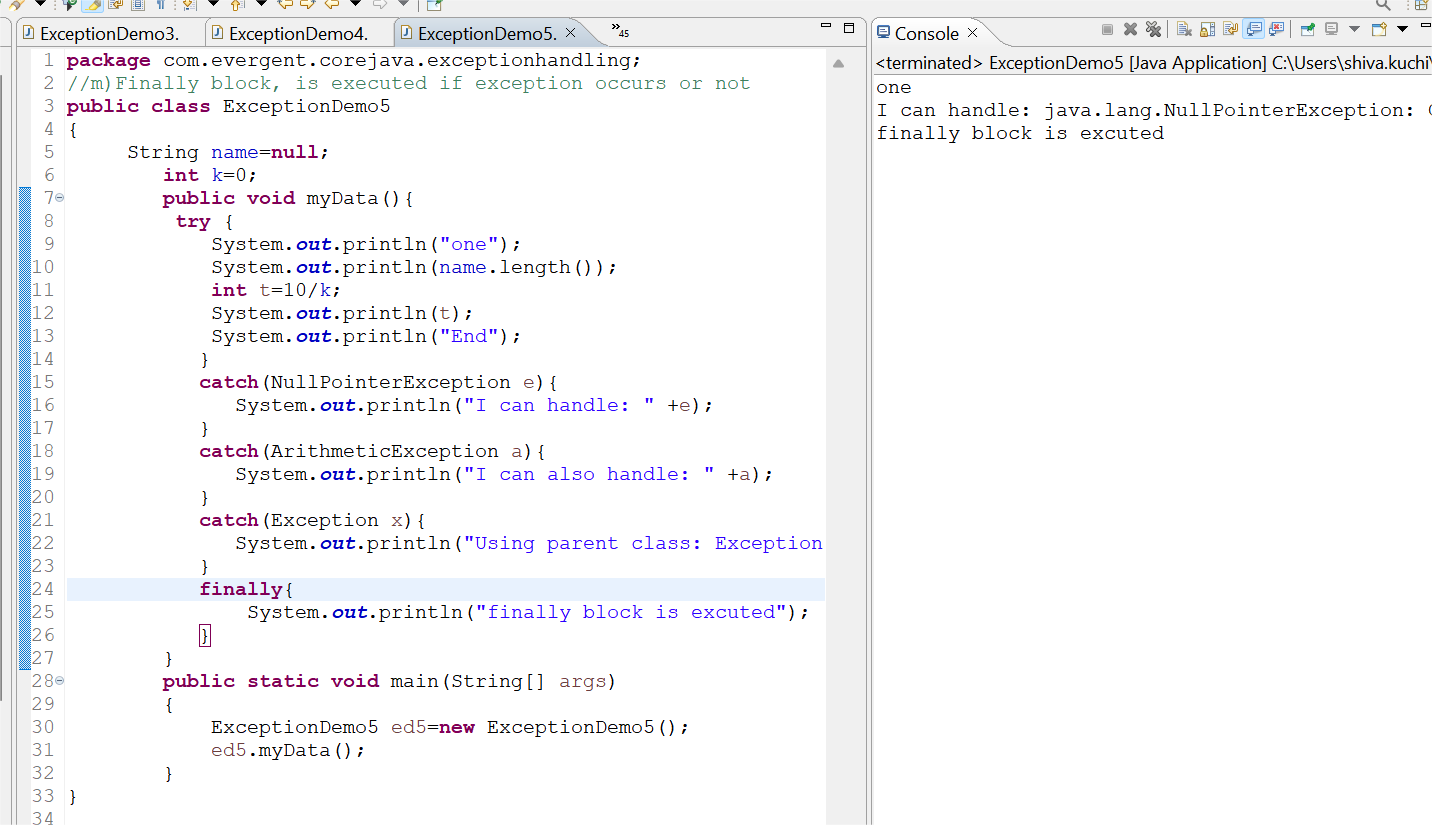
Program 3



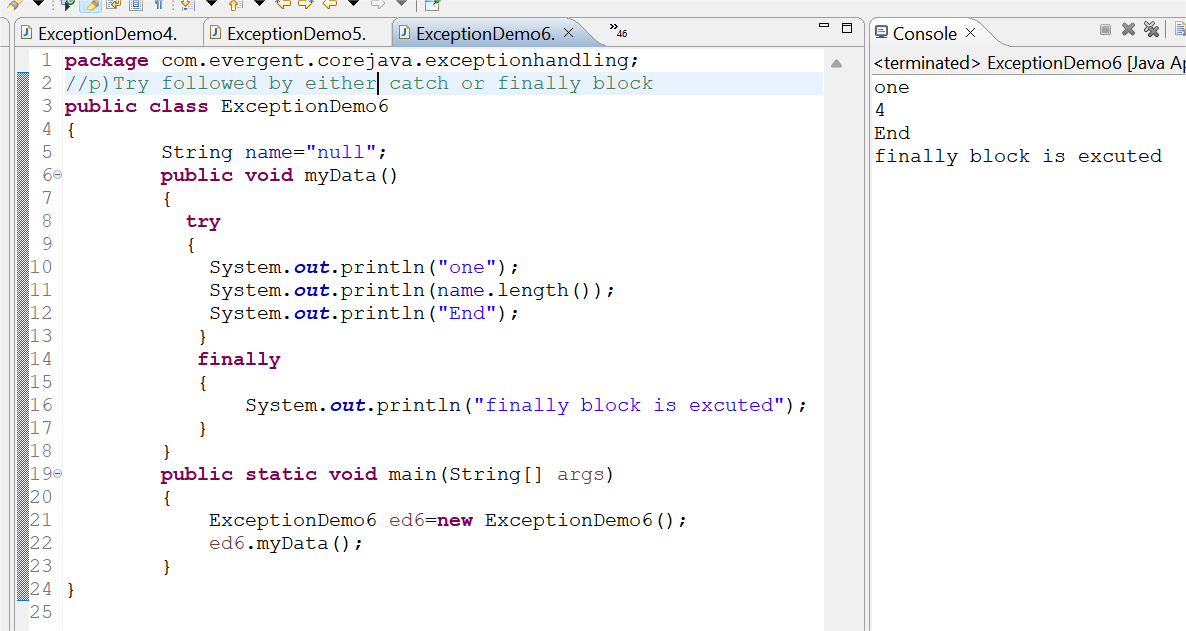
Program 4



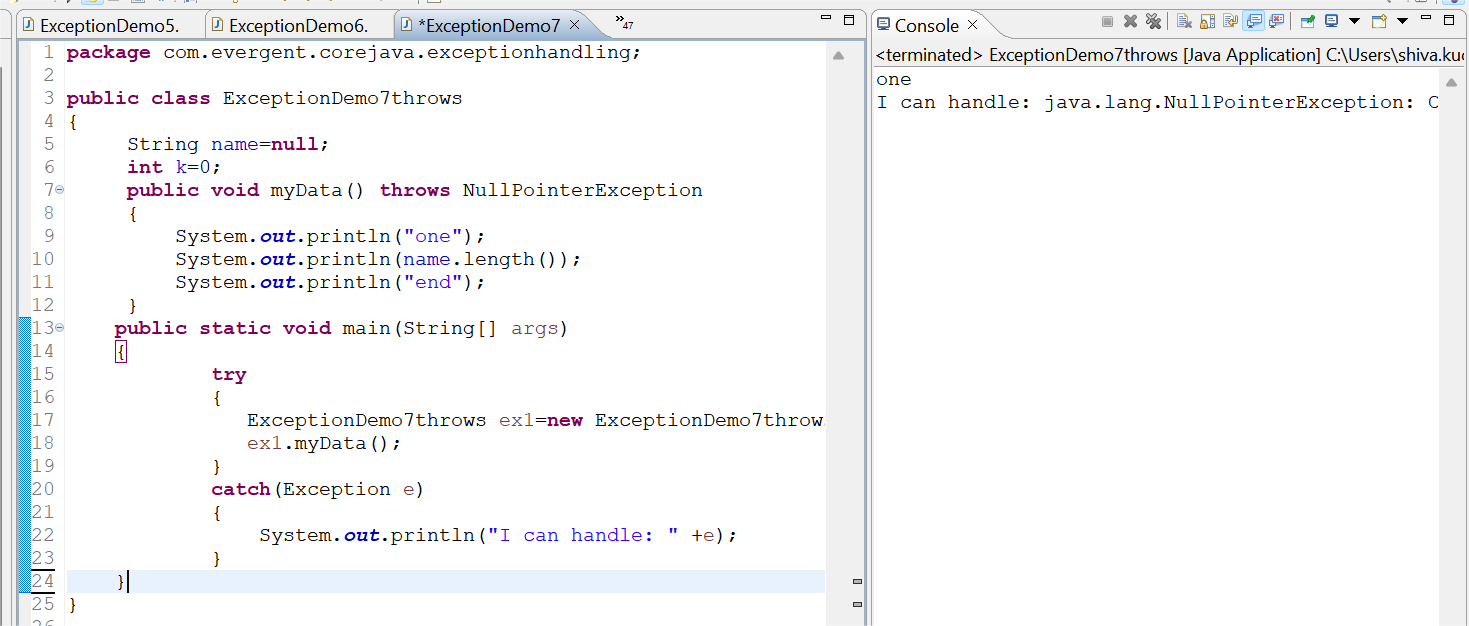
Program 5



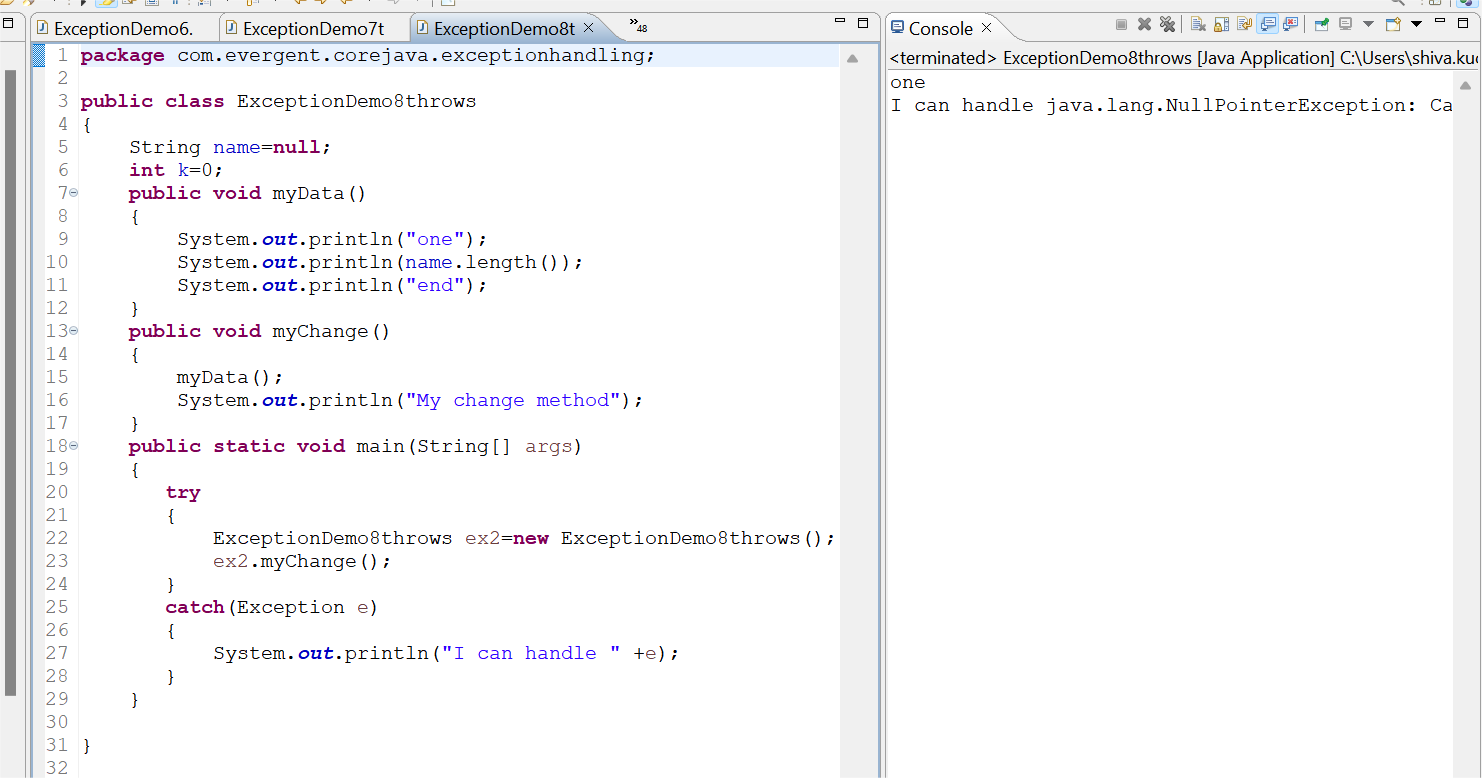
Program 6



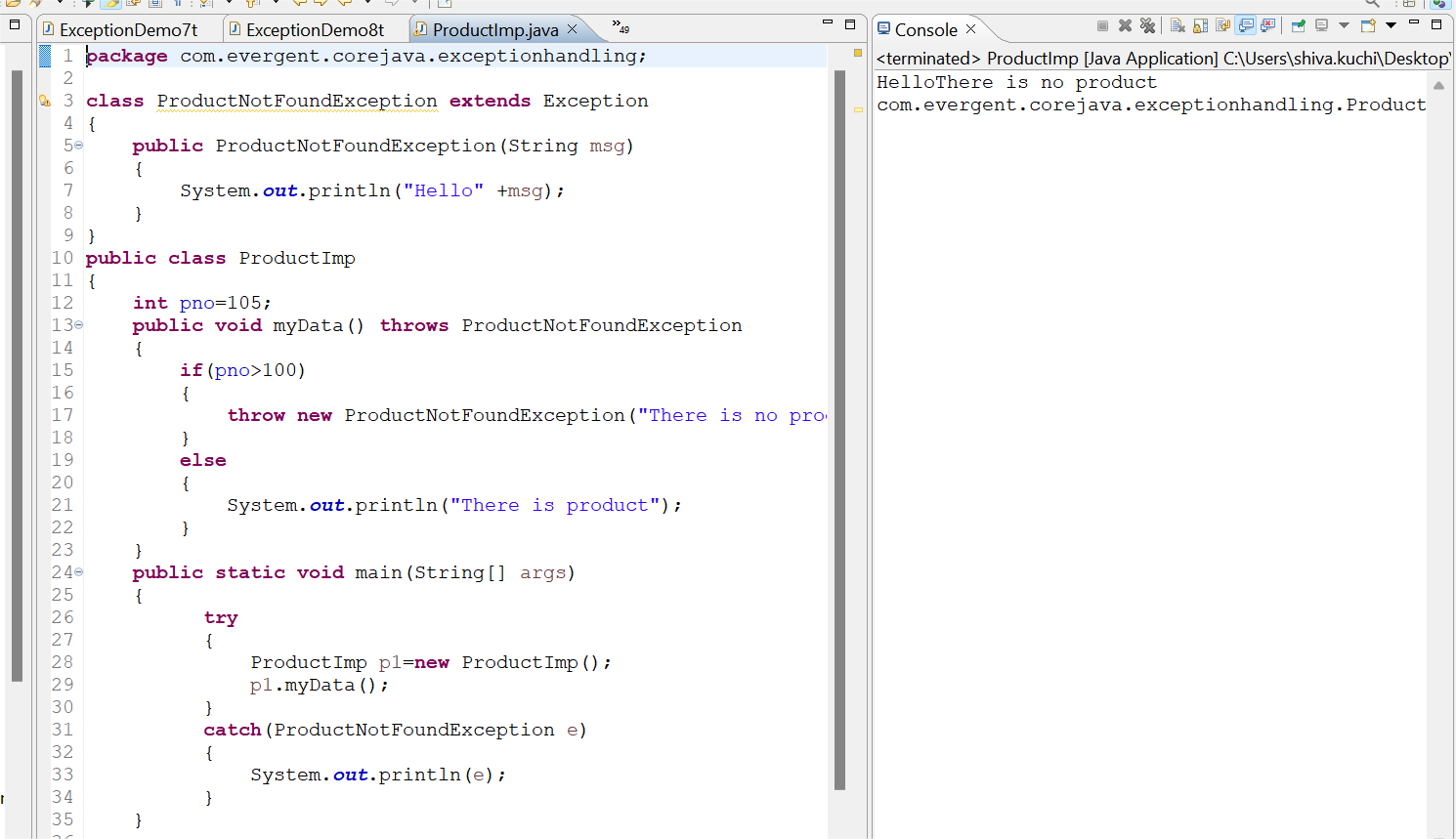
Program 7



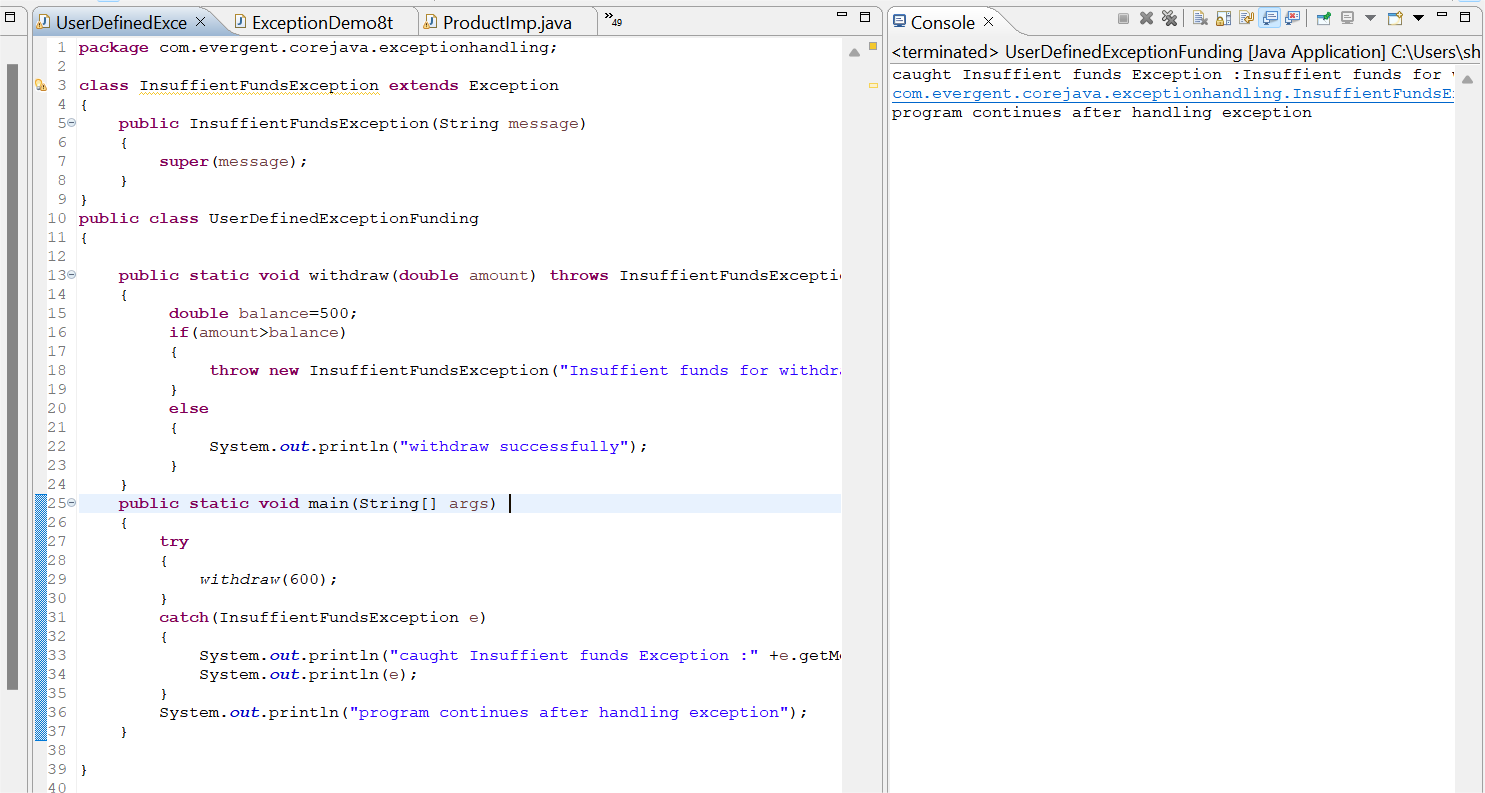
Program 8



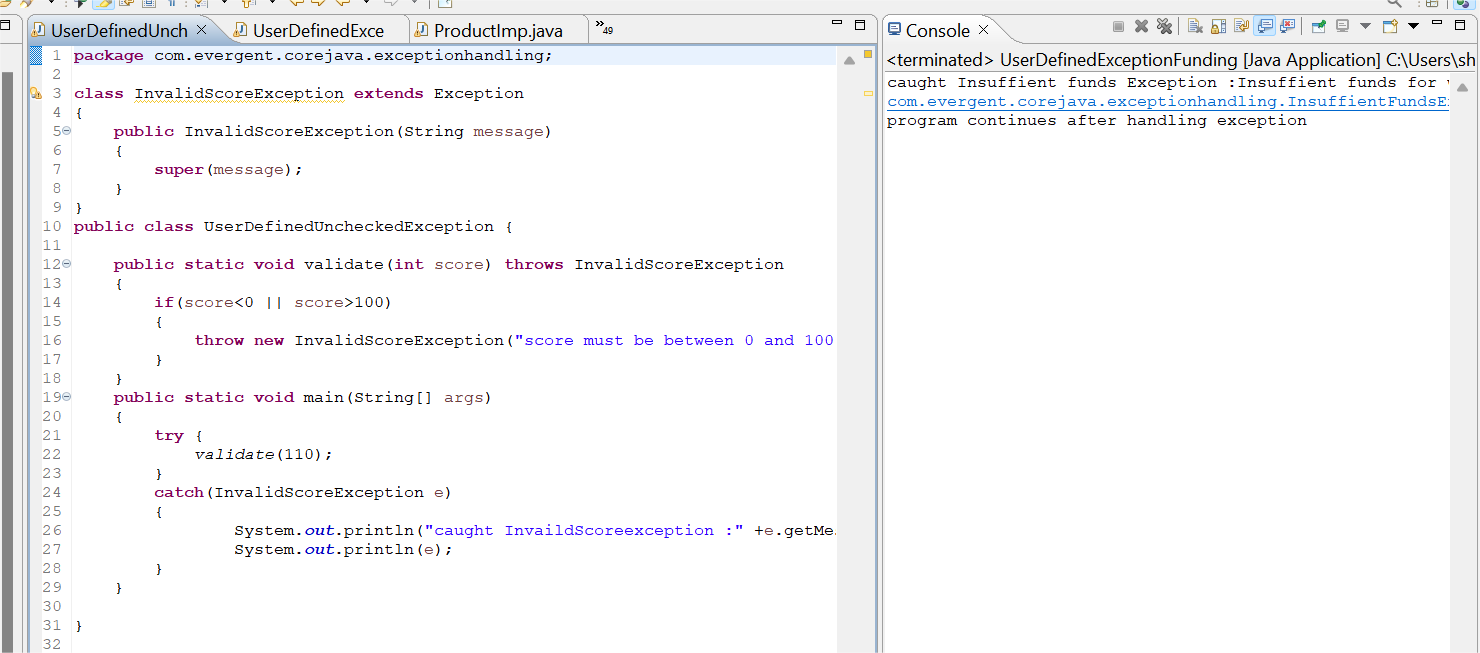
Program 9



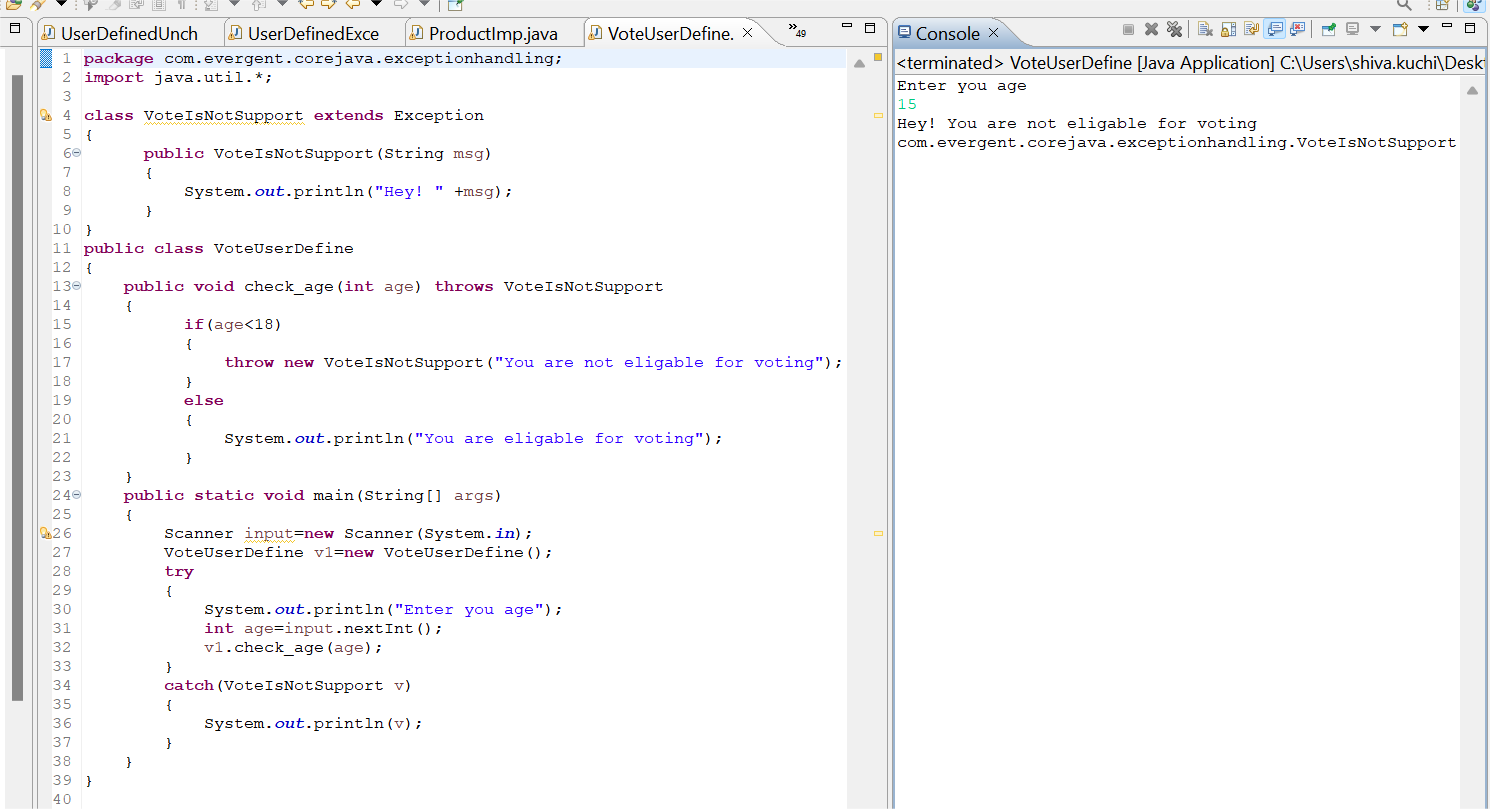
Program 10



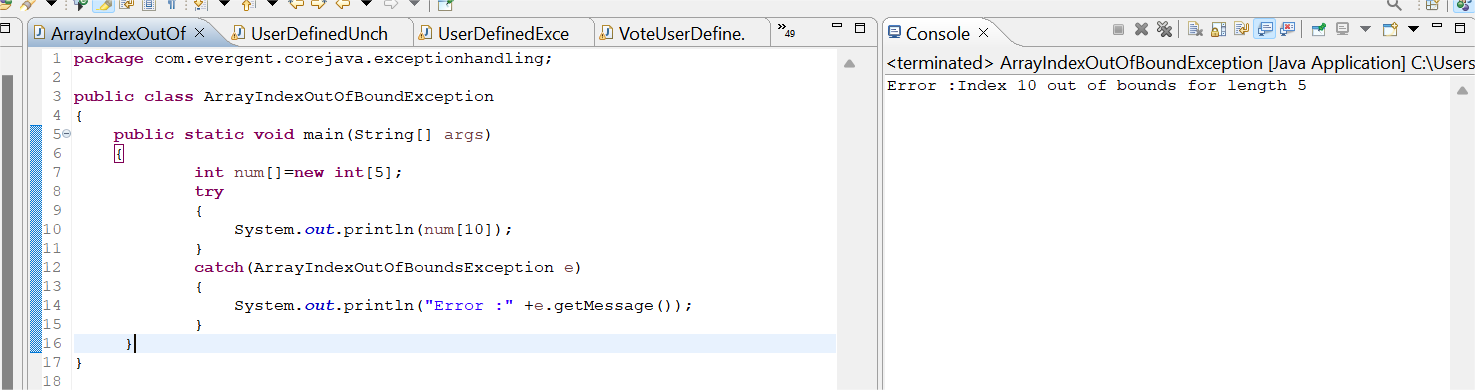
Program 11



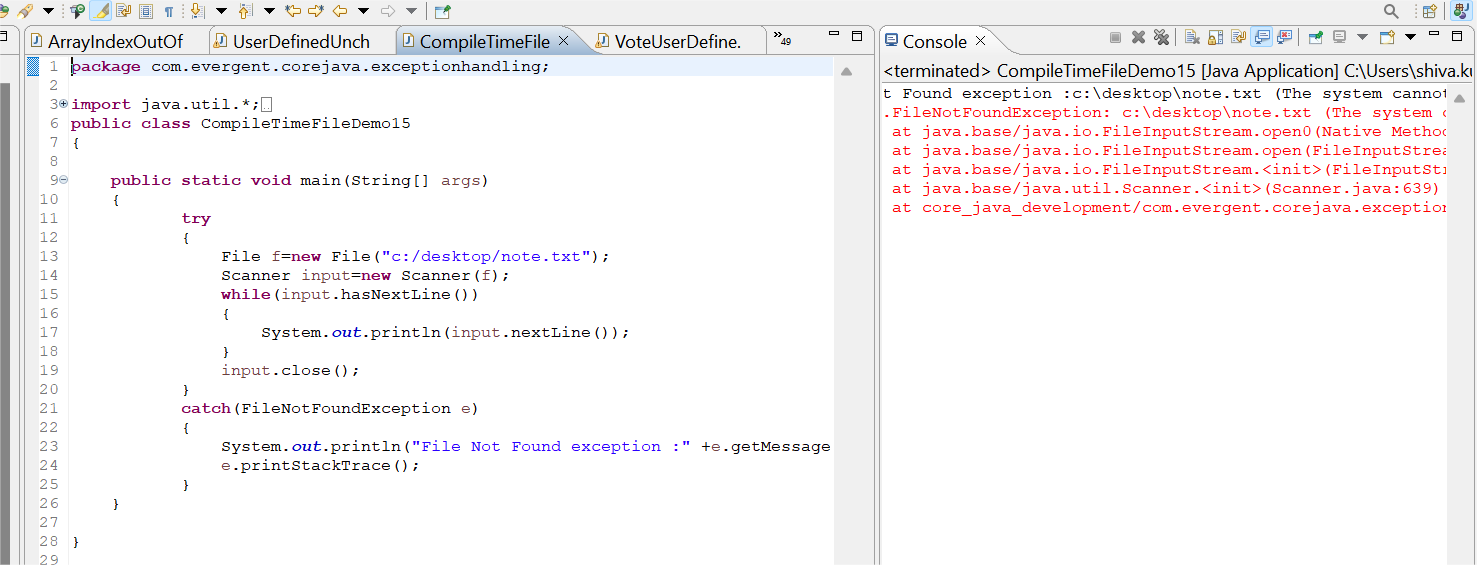
Program 12



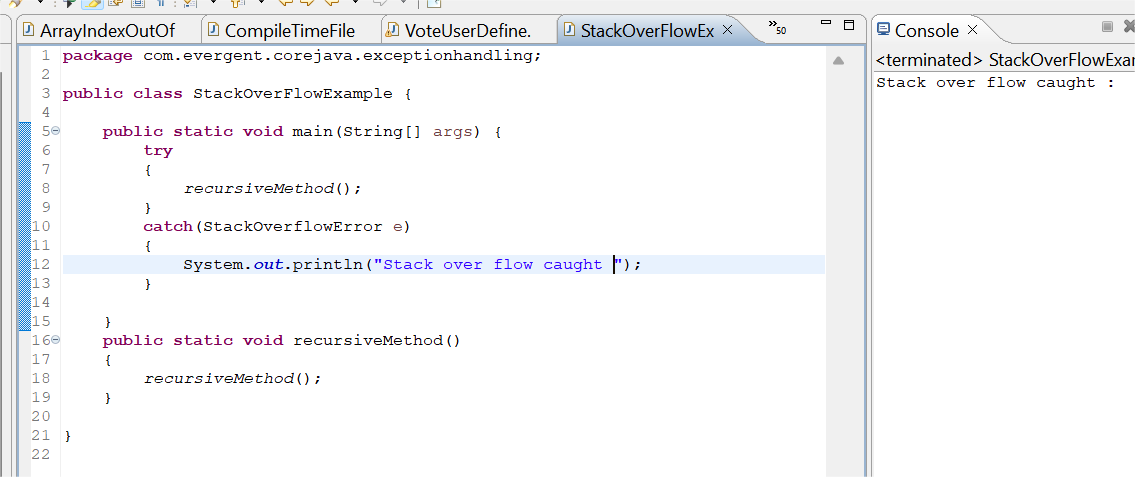
Program 13



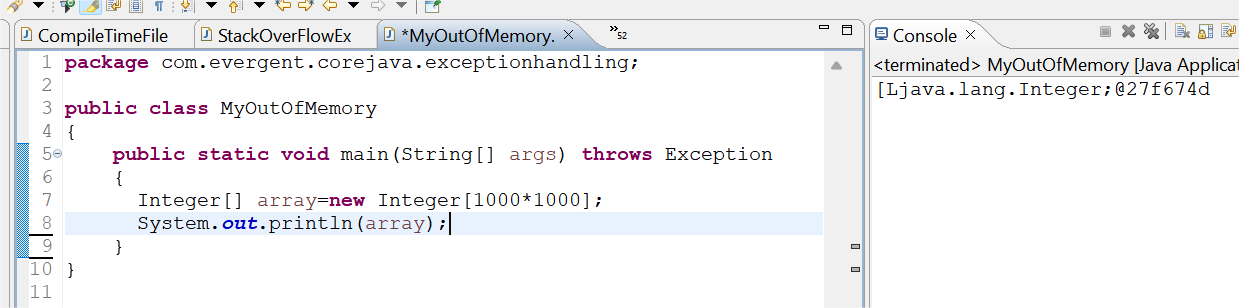
Program 14



Program 15



Program 16



Program 17

