**Core JAVA training - index**

**Date:05/08/2024**

1. Language And Applications
2. JAVA Features

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

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

**Date:06/08/2024**

**Mrng(11:00 am)**

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

**AfterNoon(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

MethodFlow

**AfterNoon(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

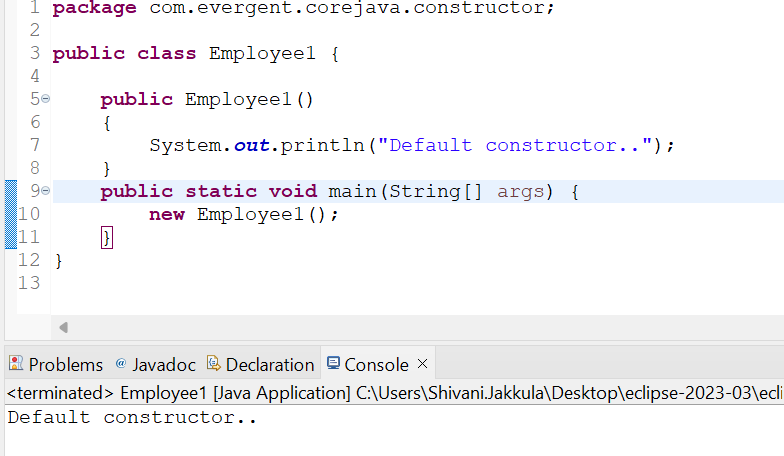
vi. Every class needs atleast 1 deafult constructor

vii. this, super This

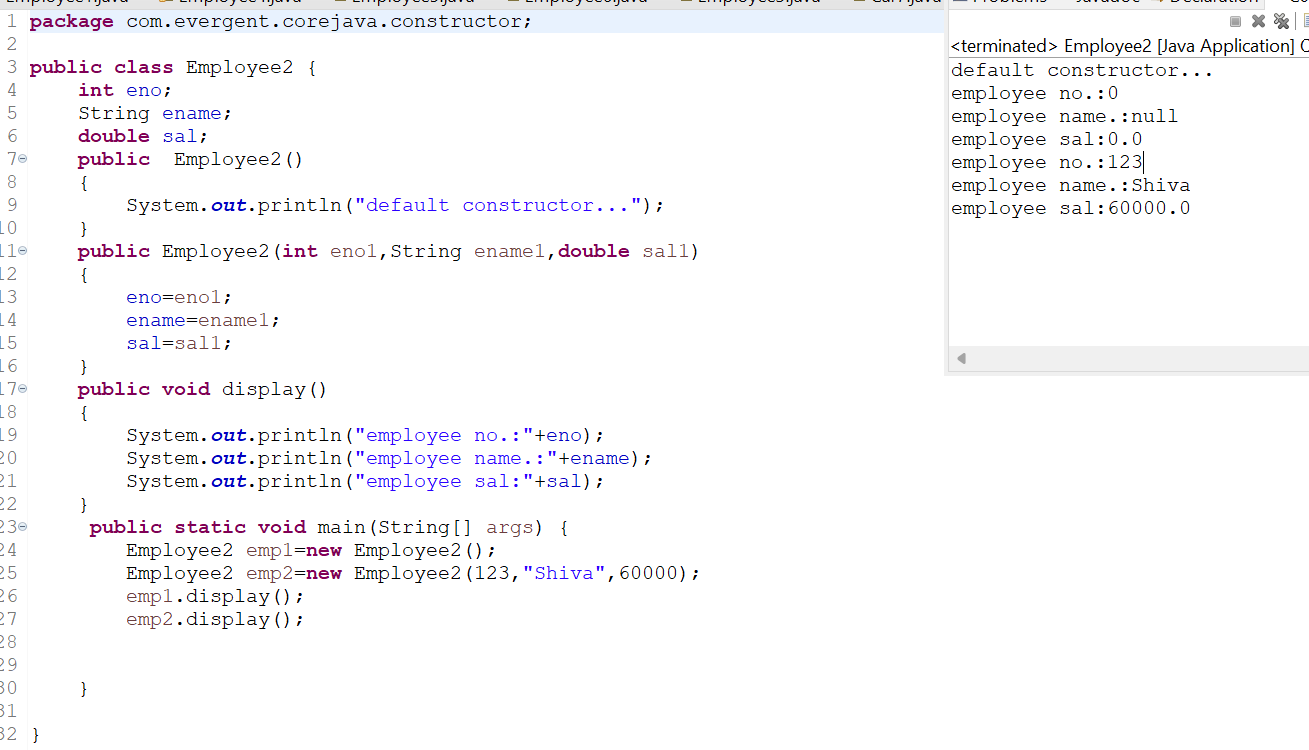
--> this is a keyword always refers to instance variables

viii. Always constructor are overloaded

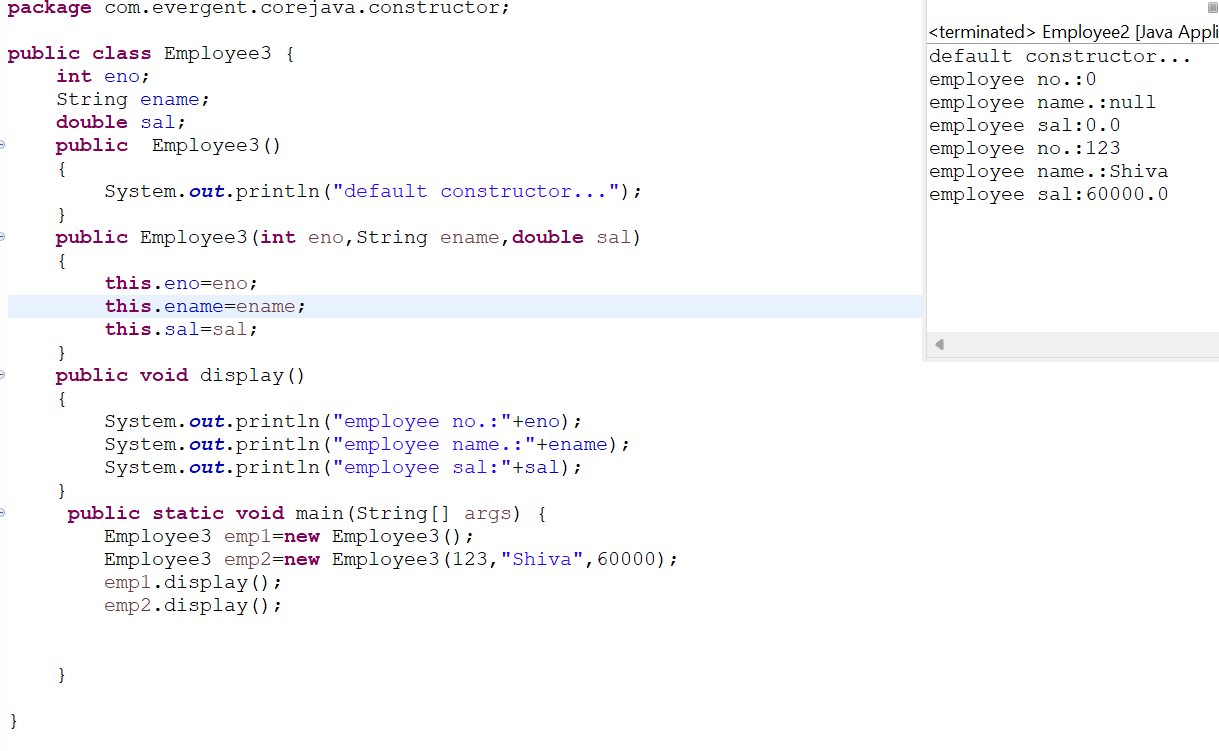
**Program1:**

****

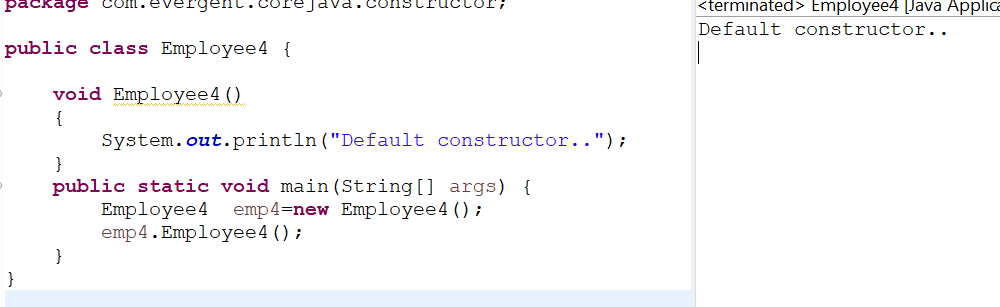
**Program2:**

****

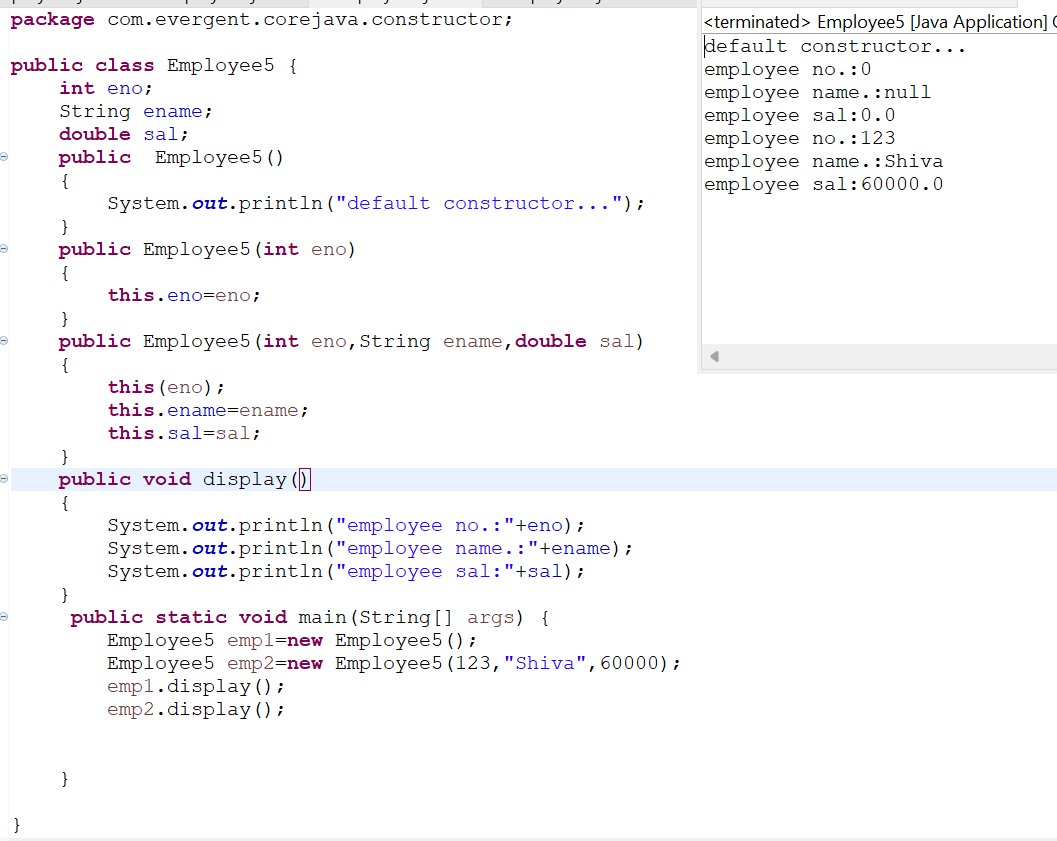
**Program3:**

****

**Program4:**

****

**Program5:**

****

**Program 6:**

****

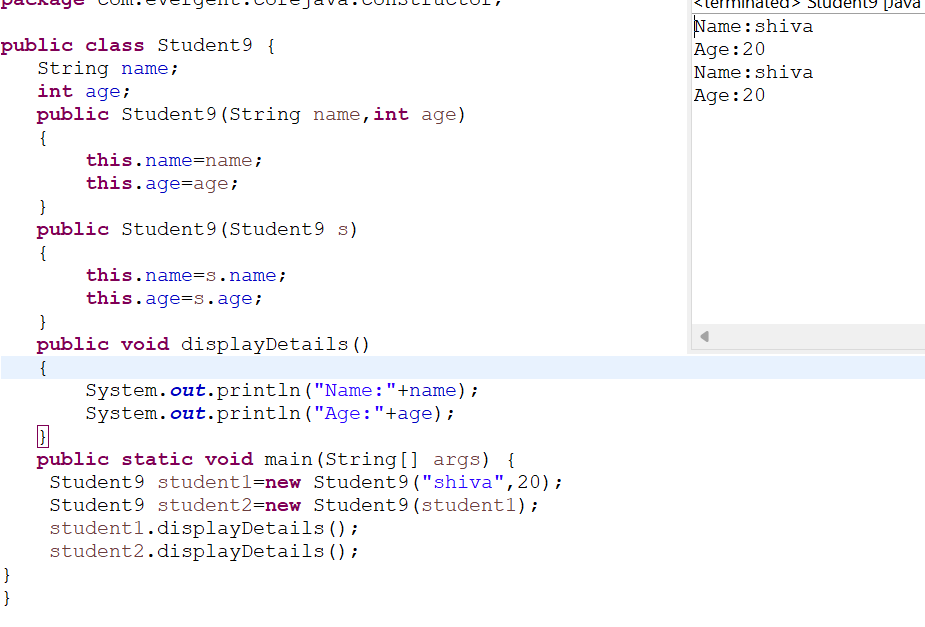
**Program 7:**

****

**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.

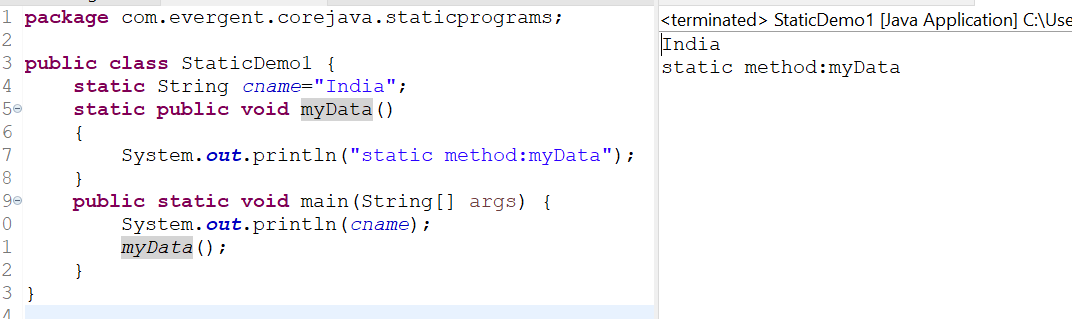
d. Static methods can access static methods and static variables only.

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

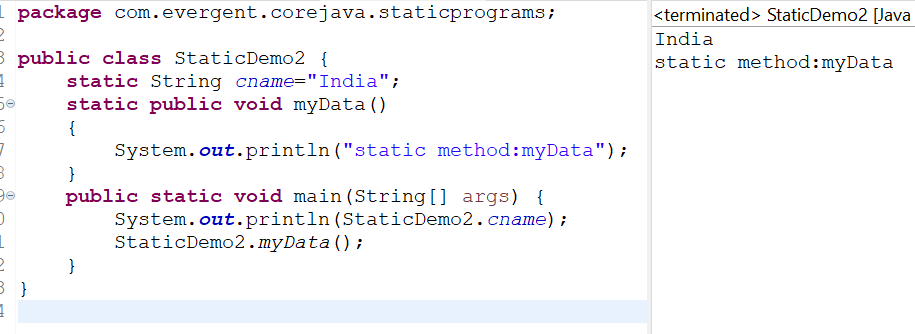
f. Non static methods can access static methods and static variables.

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

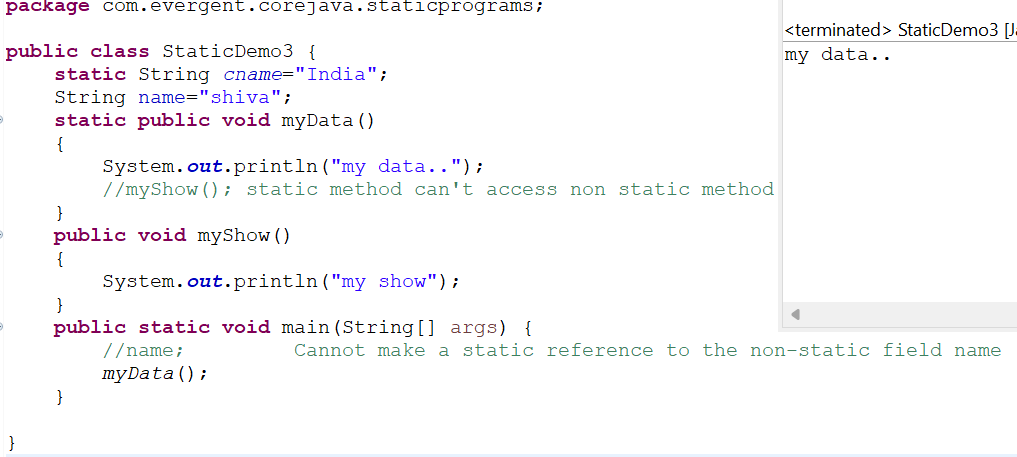
**Program1:**

****

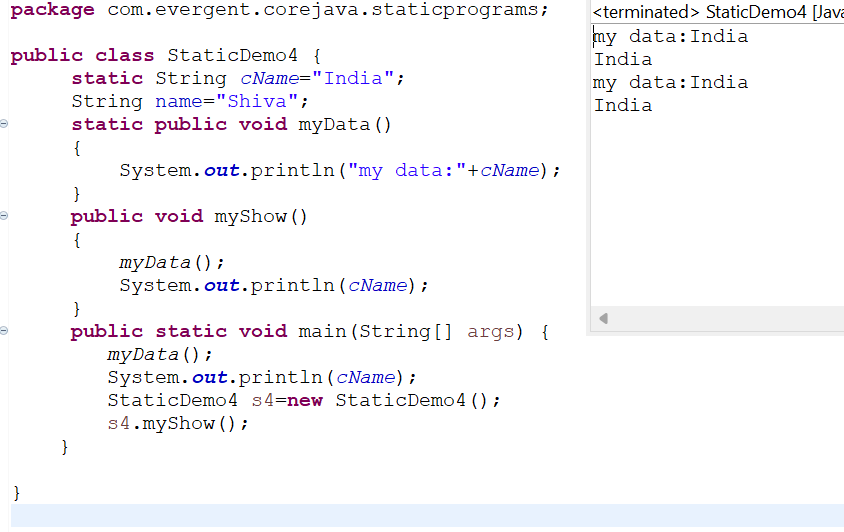
**Program2:**

****

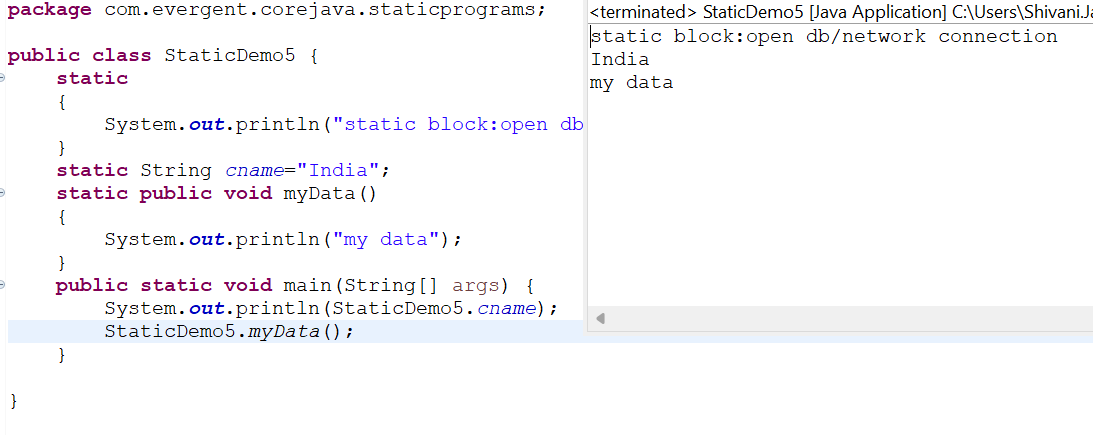
**Program3:**

****

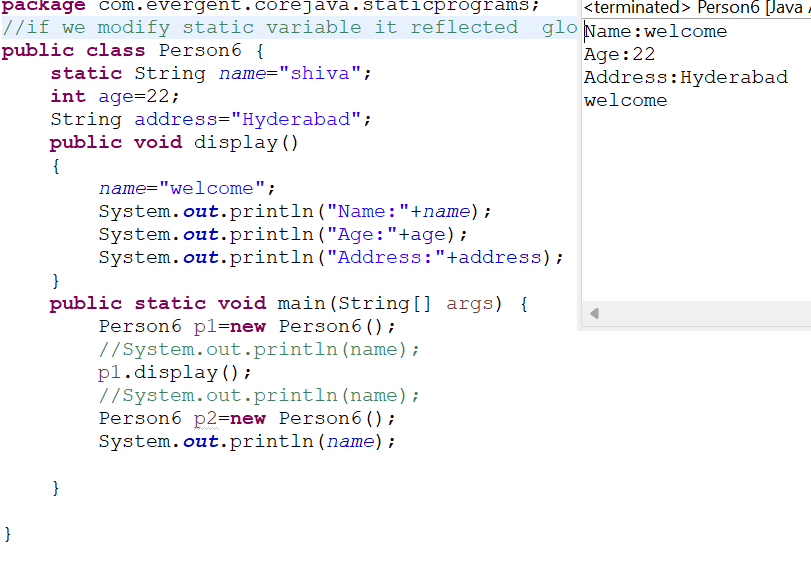
**Program4:**

****

**Program5:**

****

**Program6:**

****

**2. Final**

a. Final is a Keyword.

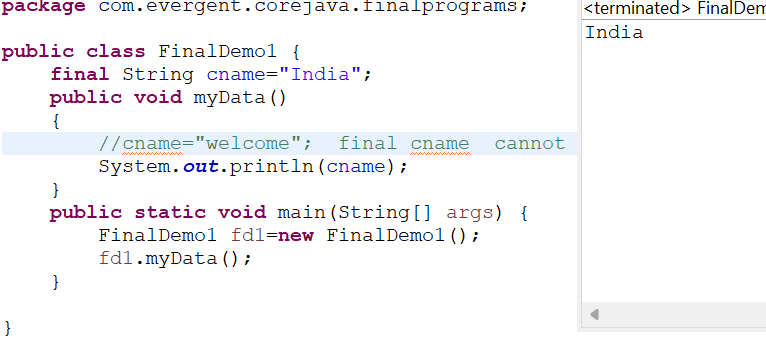
b. We can declare a variable, method, or a class as final.

c. Final variable cannot be modified.

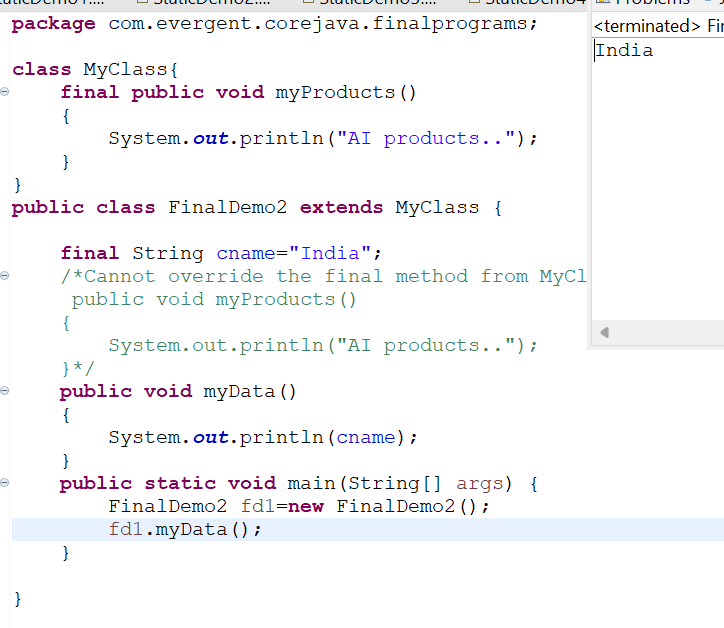
d. Final Method cannot be overrided.

e. Final class cannot be inherited.

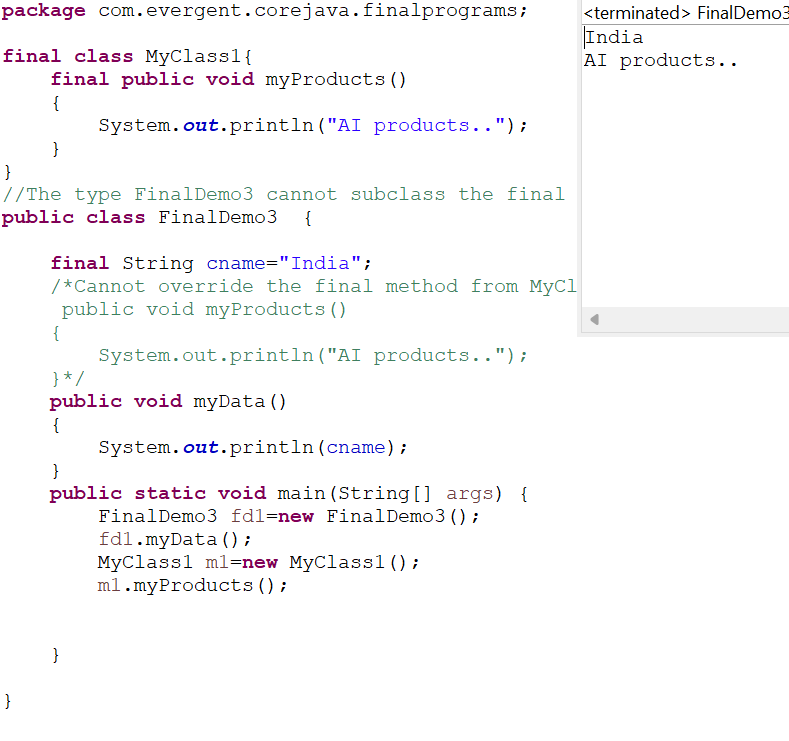
**Program1:**

****

**Program2:**

****

**Program3:**

****

**Date:12/08/2024 - Day6**

1. **String**
2. why String is immutable?
3. String is final class
4. String class having methods.
5. All string class methods are non-synchronized.

**METHODS:**

1. length();
2. toLowercase();
3. toUpperCase();

**2.**  **StringBuffer**

1. StringBuffer is final class
2. StringBuffer is mutable
3. StringBuffer having methods
4. All StringBuffer class methods are synchronized.

**METHODS:**

1. append();
2. insert();
3. replace();
4. delete();
5. reverse();
6. capacity();
7. length();

**3. StringBuilder**

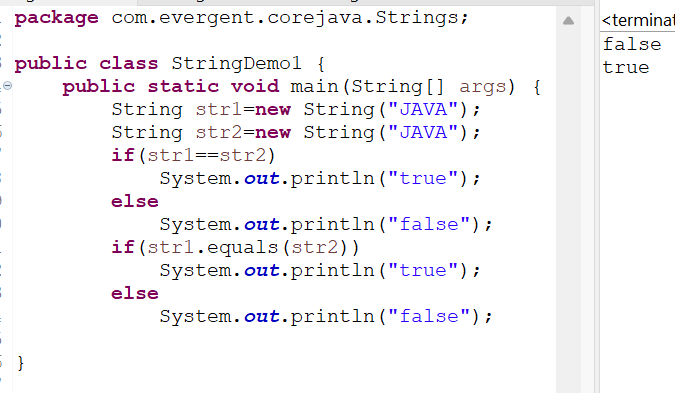
1. StringBuilder is final class
2. StringBuilder is mutable
3. StringBuilder having methods
4. All StringBuilder class methods are non-synchronized.

**METHODS:**

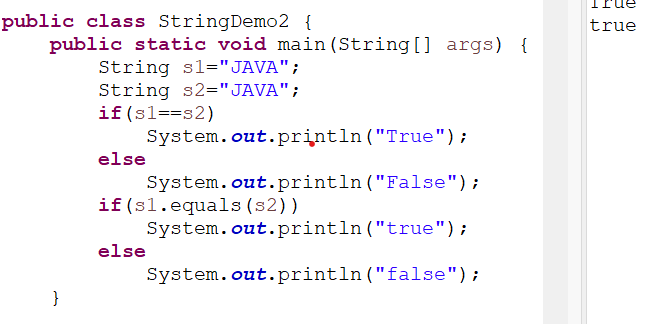
1. append();
2. insert();
3. replace();
4. delete();
5. reverse();

4. Difference between String,StringBuffer,StringBuilder

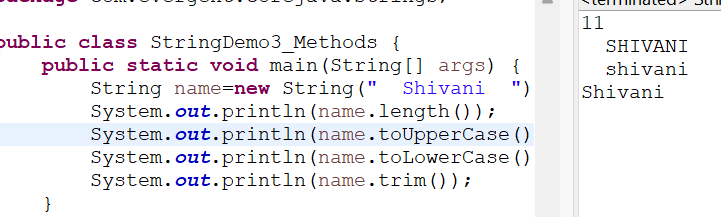
Program1:



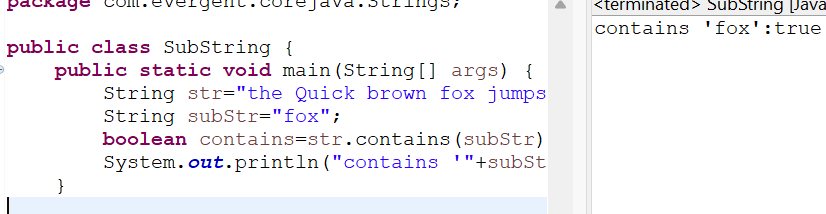
Program2:



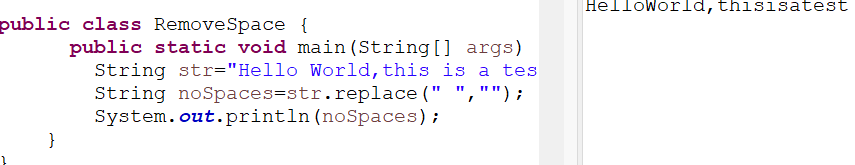
Program3:



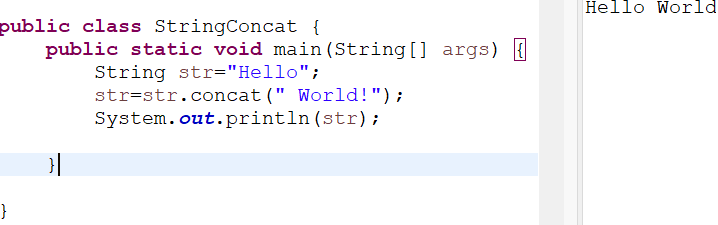
Program4:



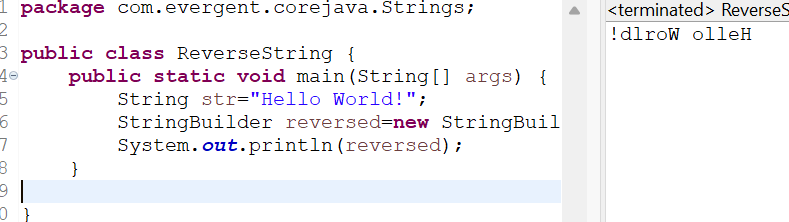
Program5:



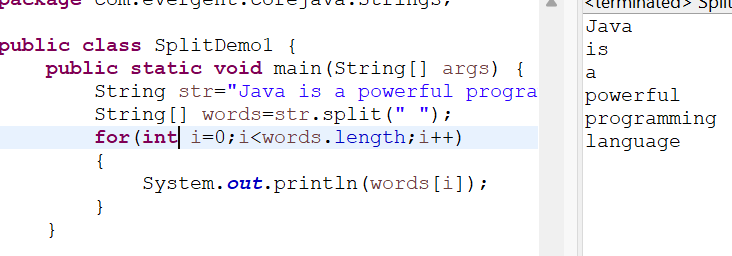
Program6:



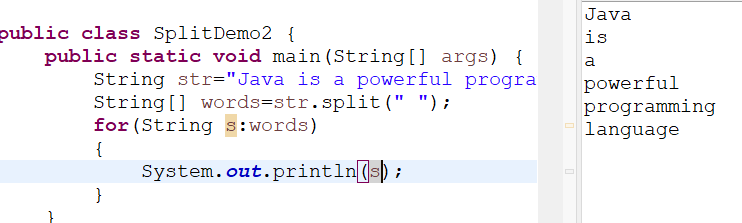
Program7:



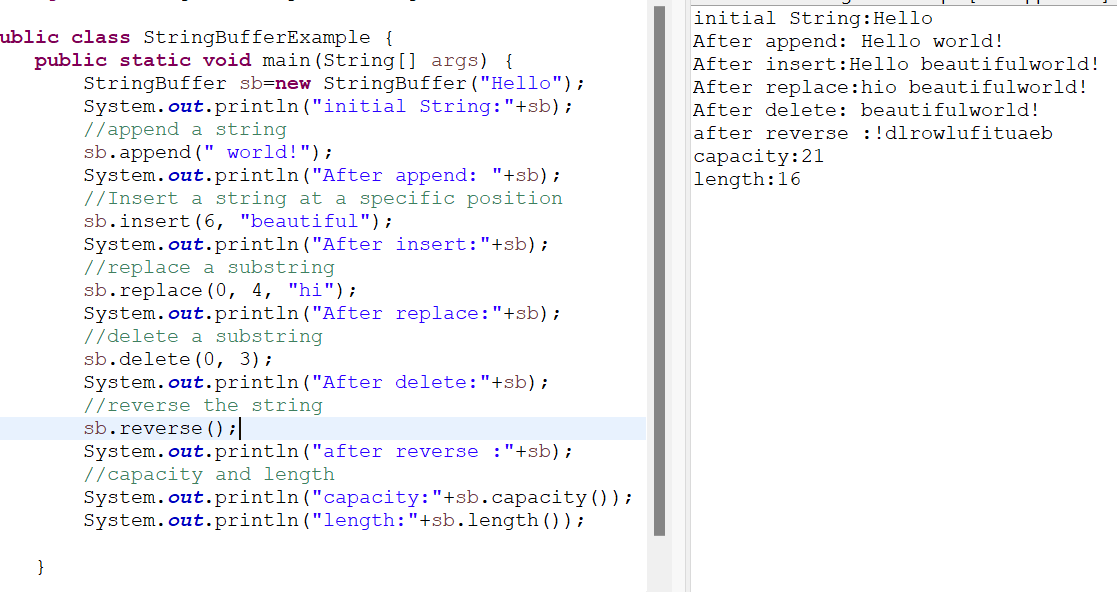
Program8:



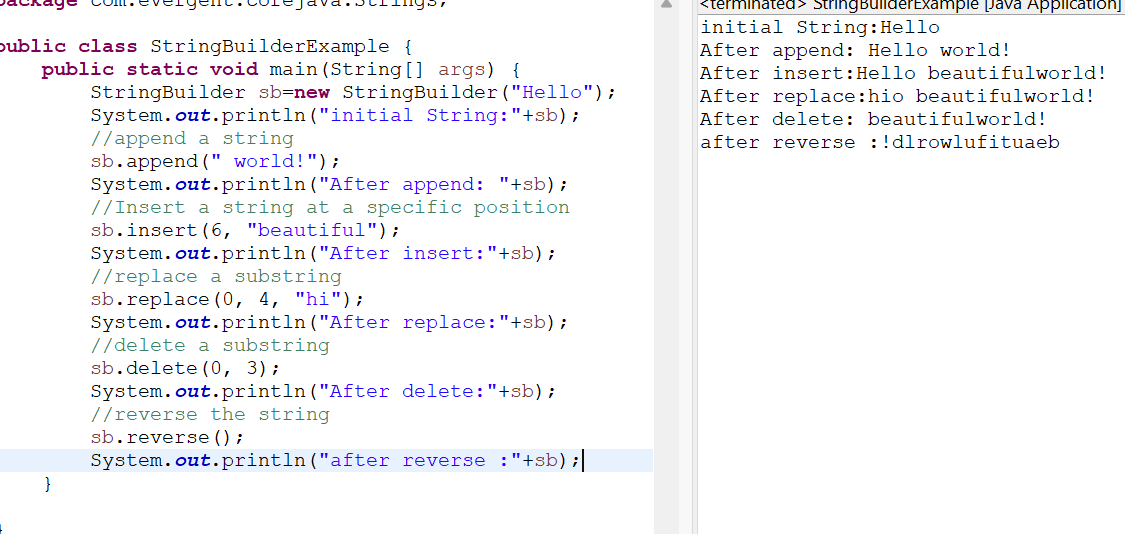
Program9:



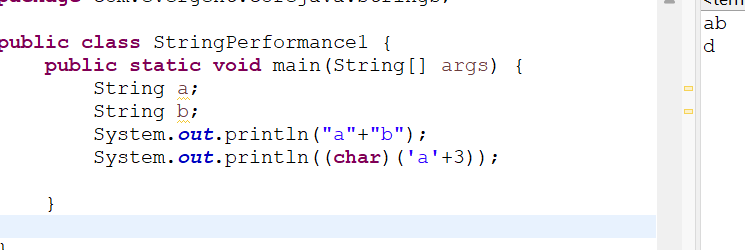
Program10:



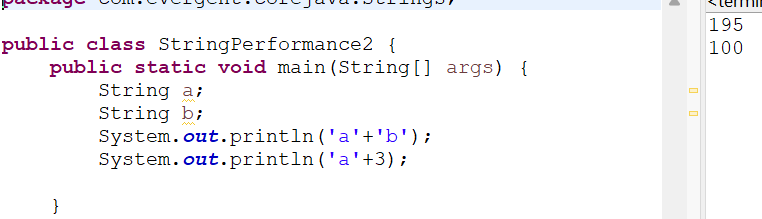
Program11:



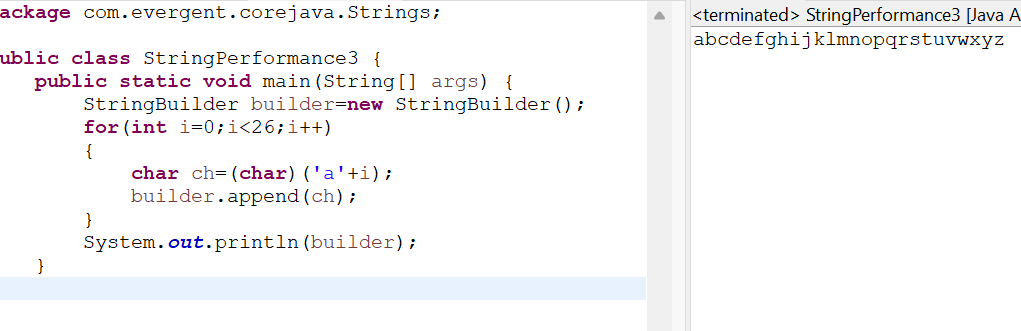
Program12:



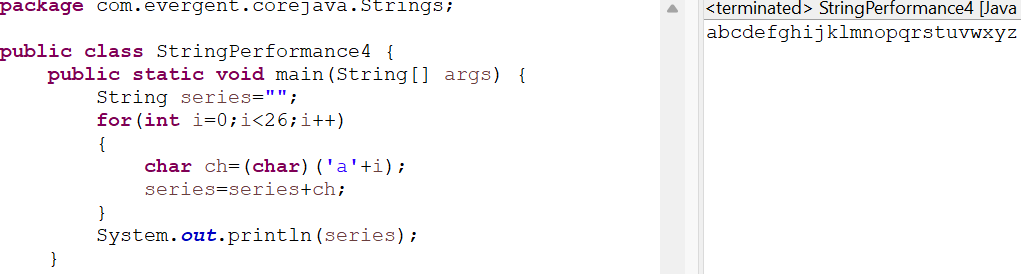
Program13:



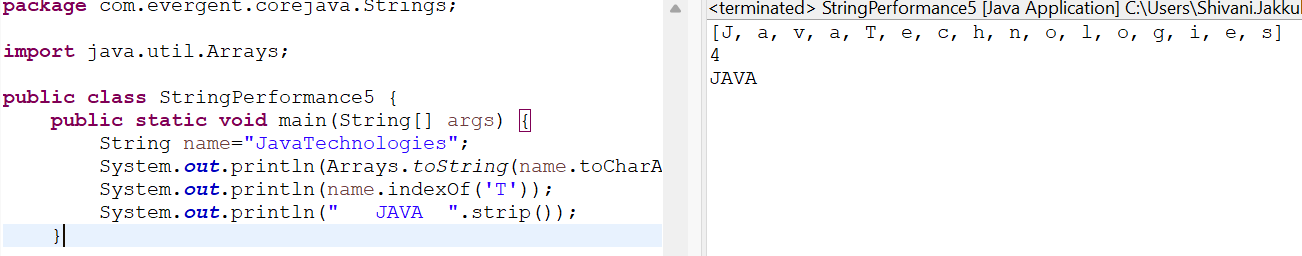
Program14:



Program15:



Program16:



**Date:13/08/2024 - Day7**

* **Can we make class as immutable?**

Yes,by declaring the class as final,as well as declaring the variable as private and final.

* Object class method - tostring()
* String immutable
* Interface

1 Interface is a keyword.

2 we can declare method signature only but not implementation.

3 By default all interface methods are abstract.

4 If any class implements interface the 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.

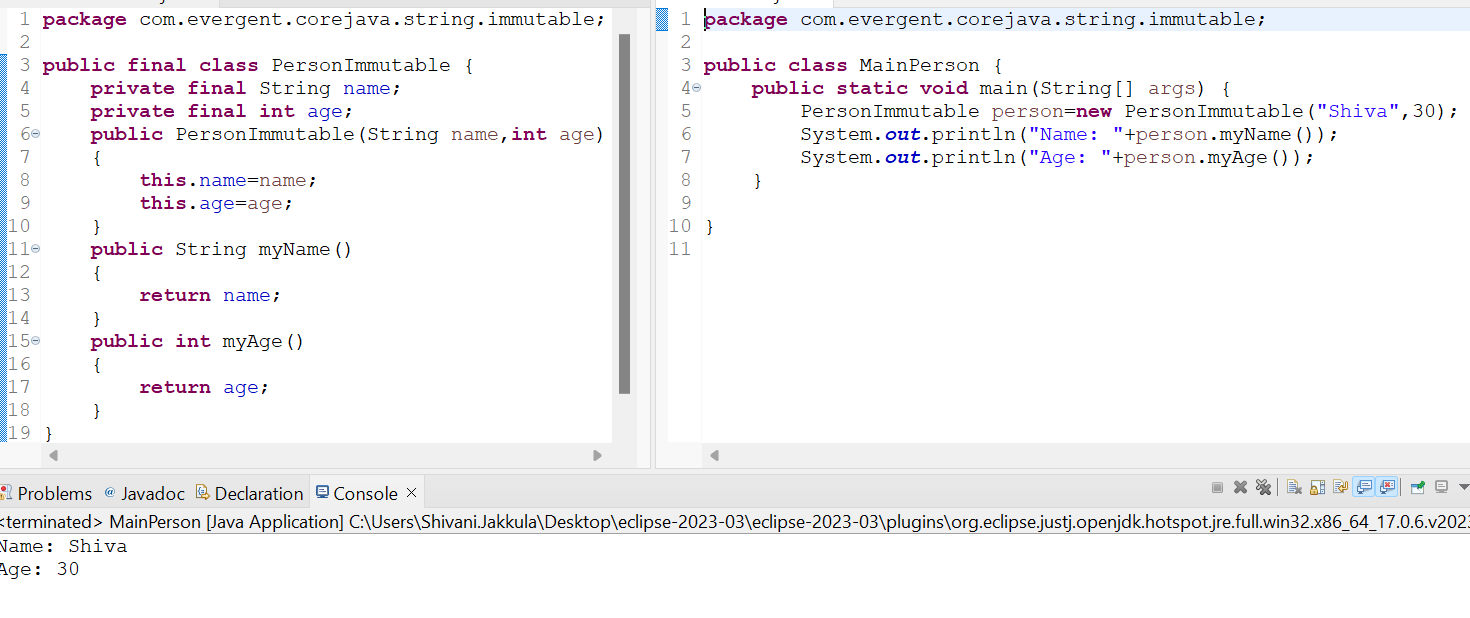
6. We can declare variables inside interface all are public static final.

7. Java will support multiple inheritance through interface.

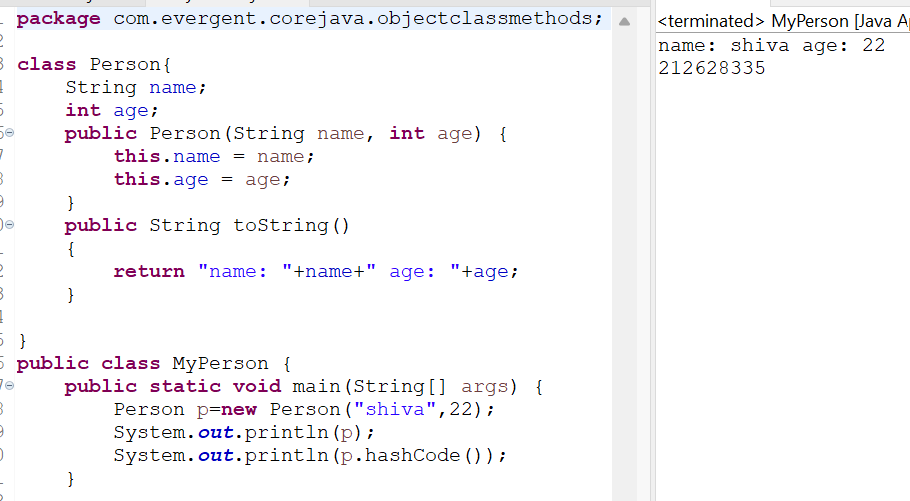
8. One class can implements more than one interface.

9. One interface can extend other interface.

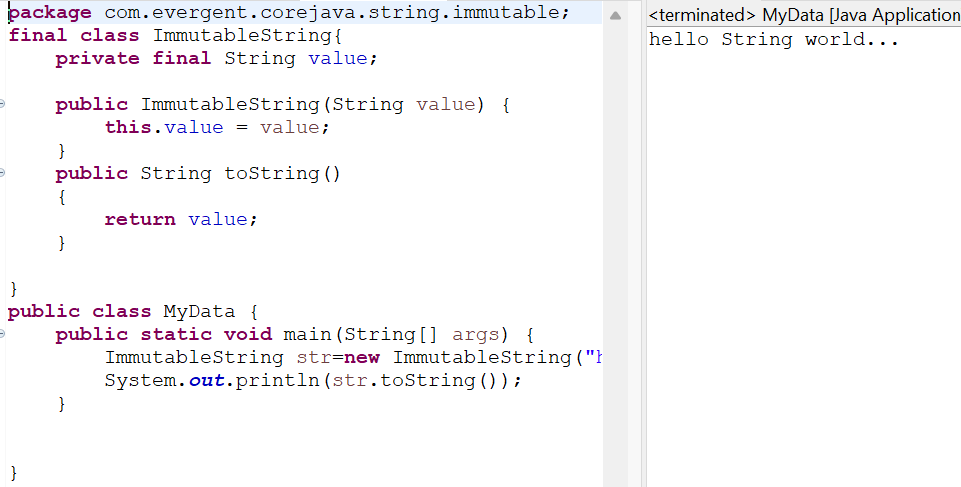
Program1: MainPerson, PersonImmutable



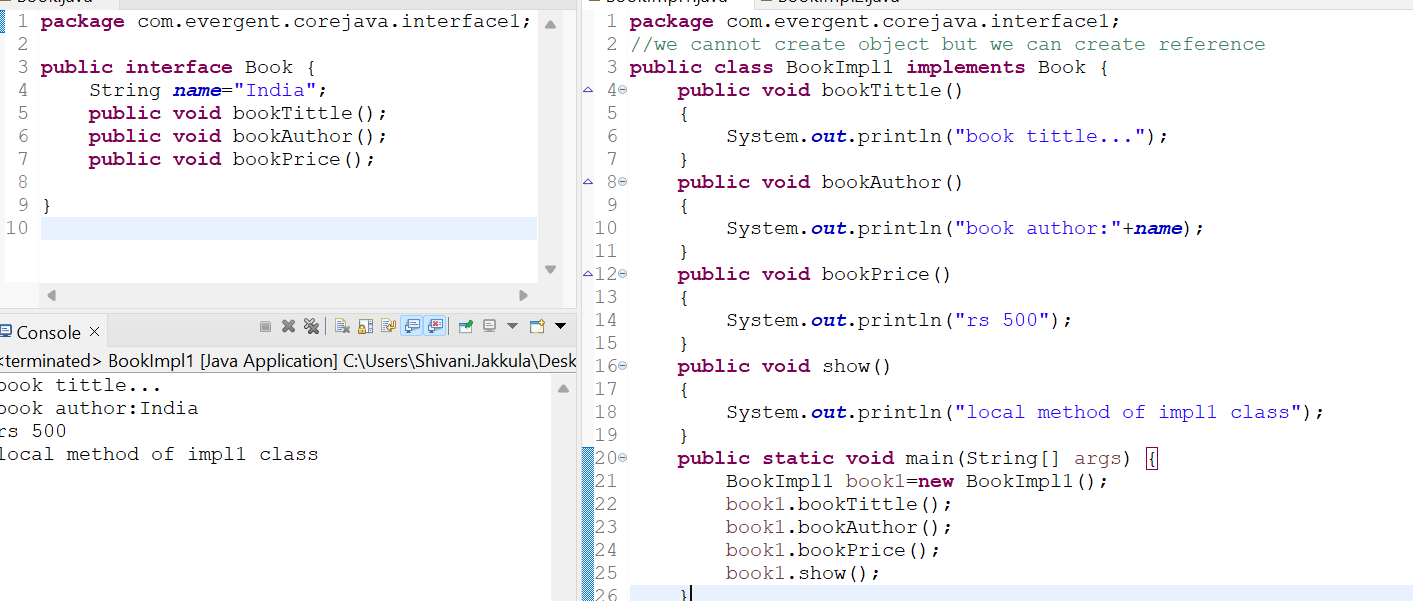
Program2:



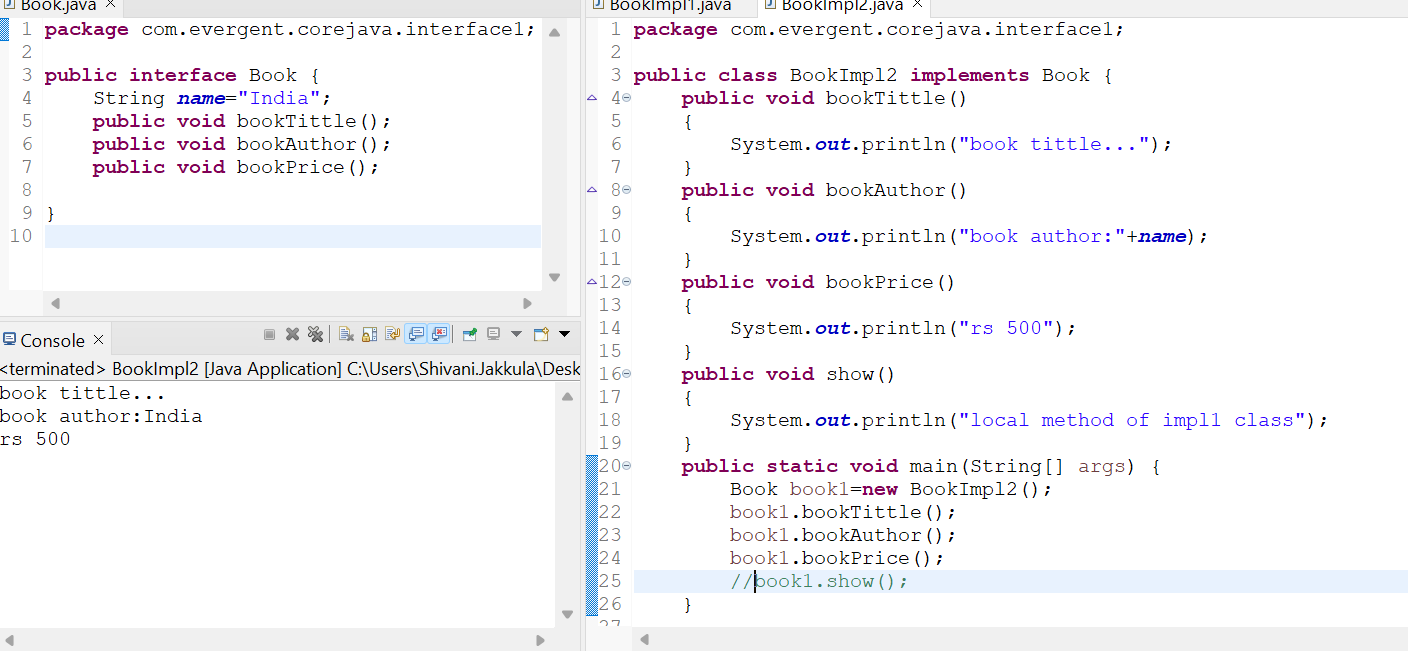
Program3:



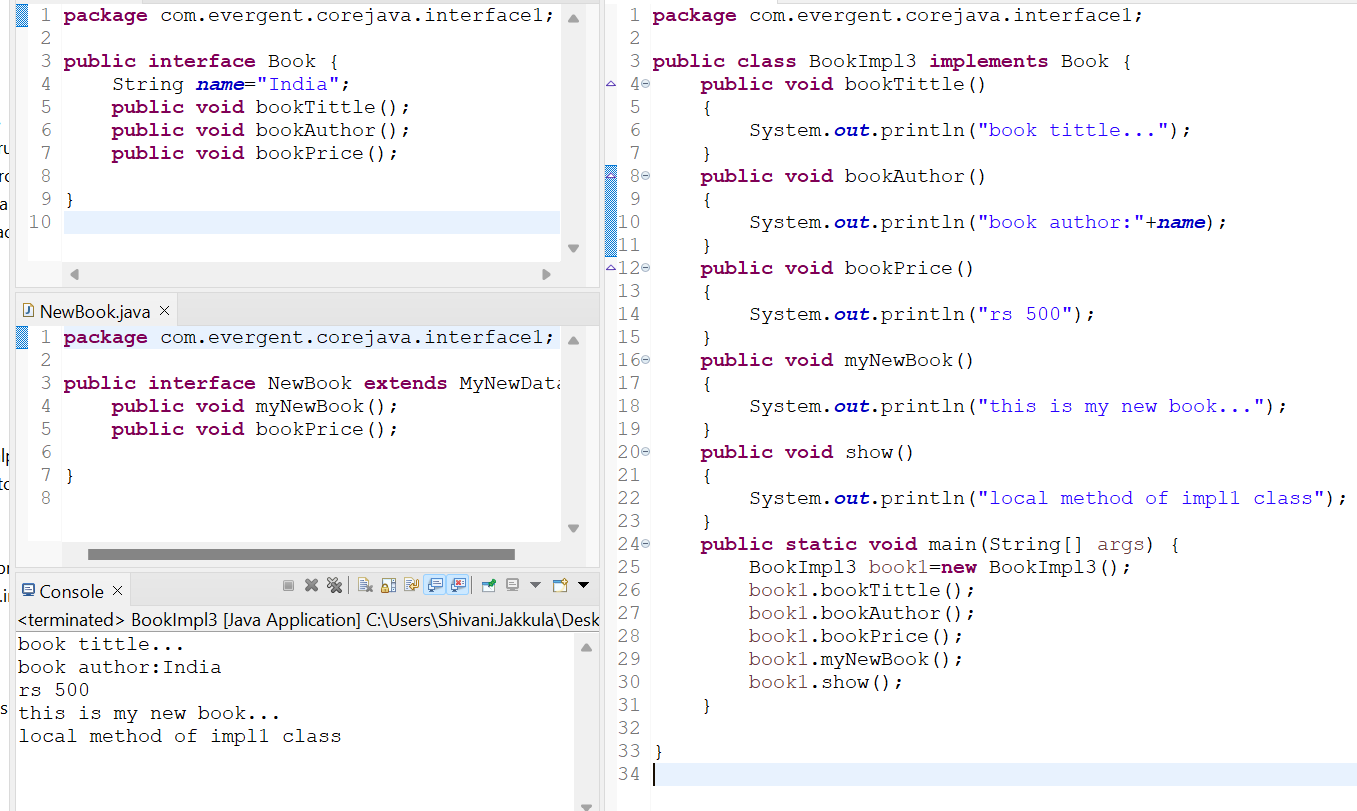
Program4:



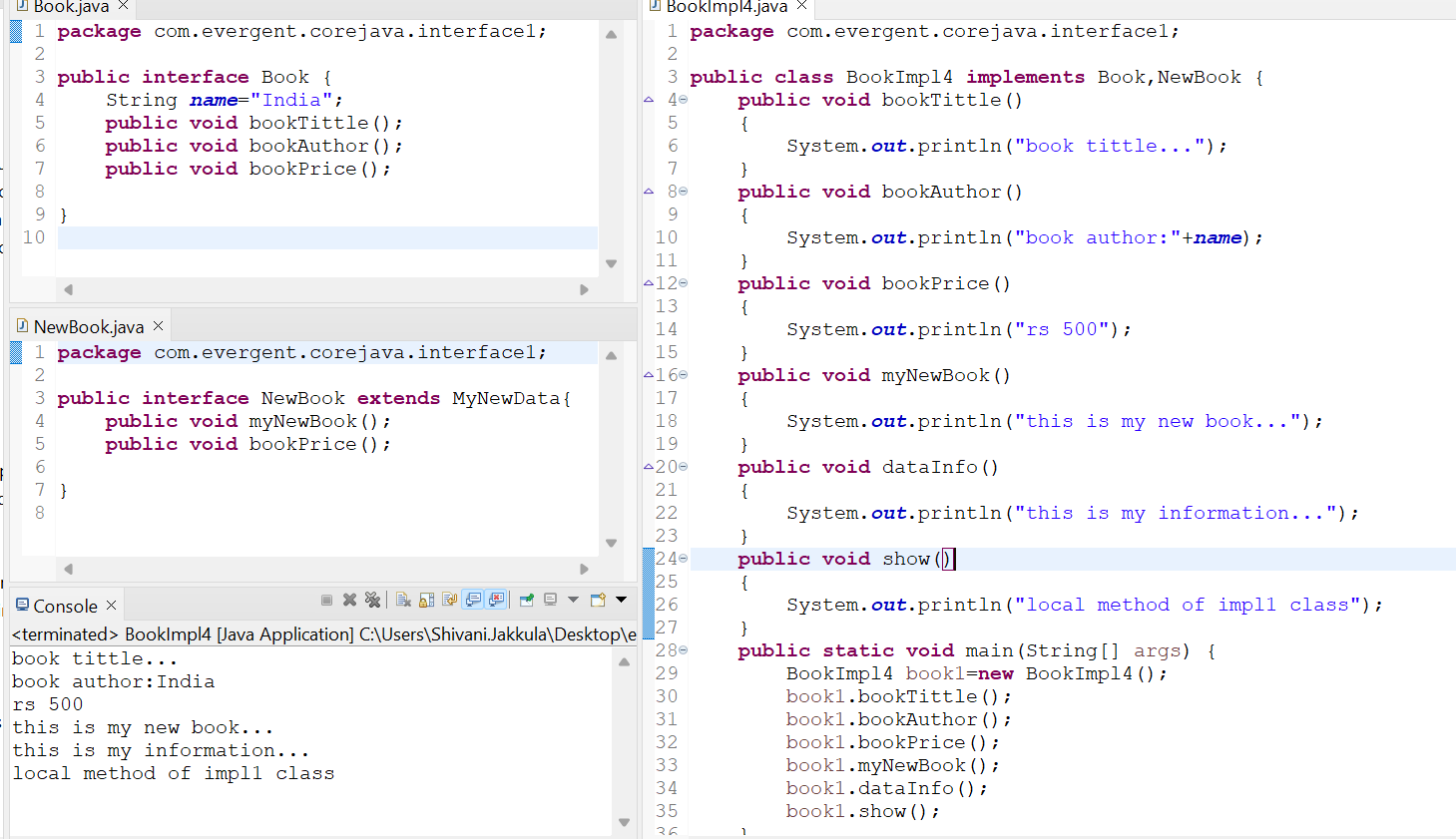
Program5:



Program6:



Program7:

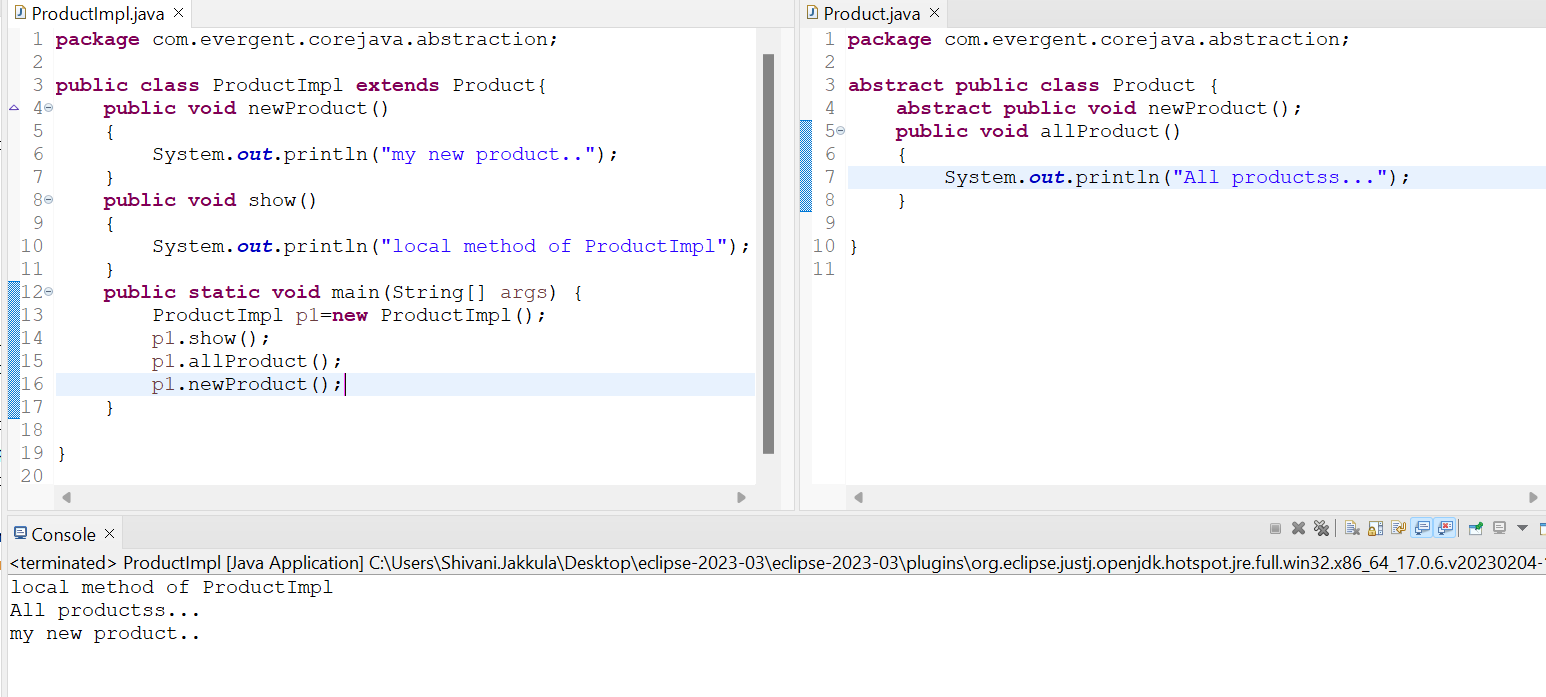


**Date:14/08/2024 - Day8**

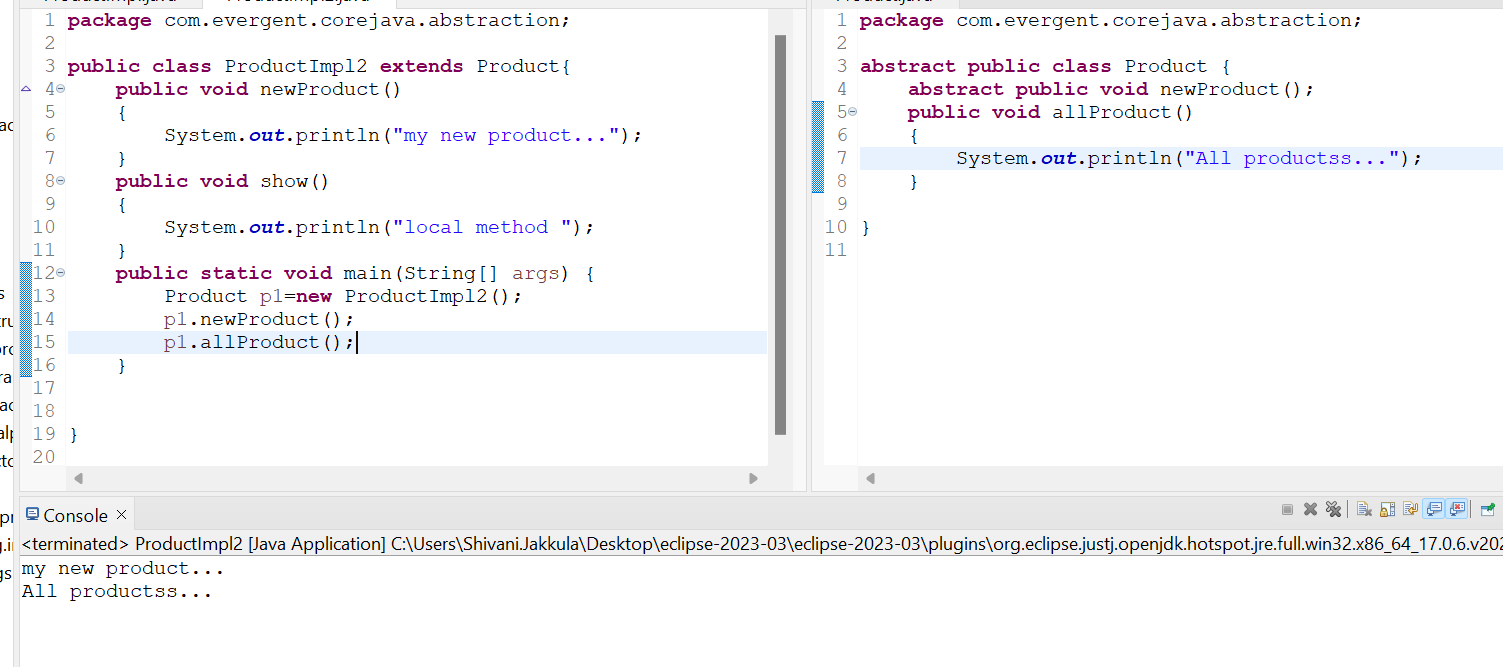
* **Abstract class**

1. Abstract is a keyword
2. Abstract class having abstract and concrete methods(implemented).
3. If any class having one abstract method that class should be declare as abstract class ,otherwise the abstract class should show compile time error.
4. If any class extends abstract class that class should be override all abstract methods otherwise the class will show compile time error.
5. We cannot create object to abstract class but we can create reference to abstract class.
6. Without any abstract methods we can also declare as abstract for the security & we can’t create object.

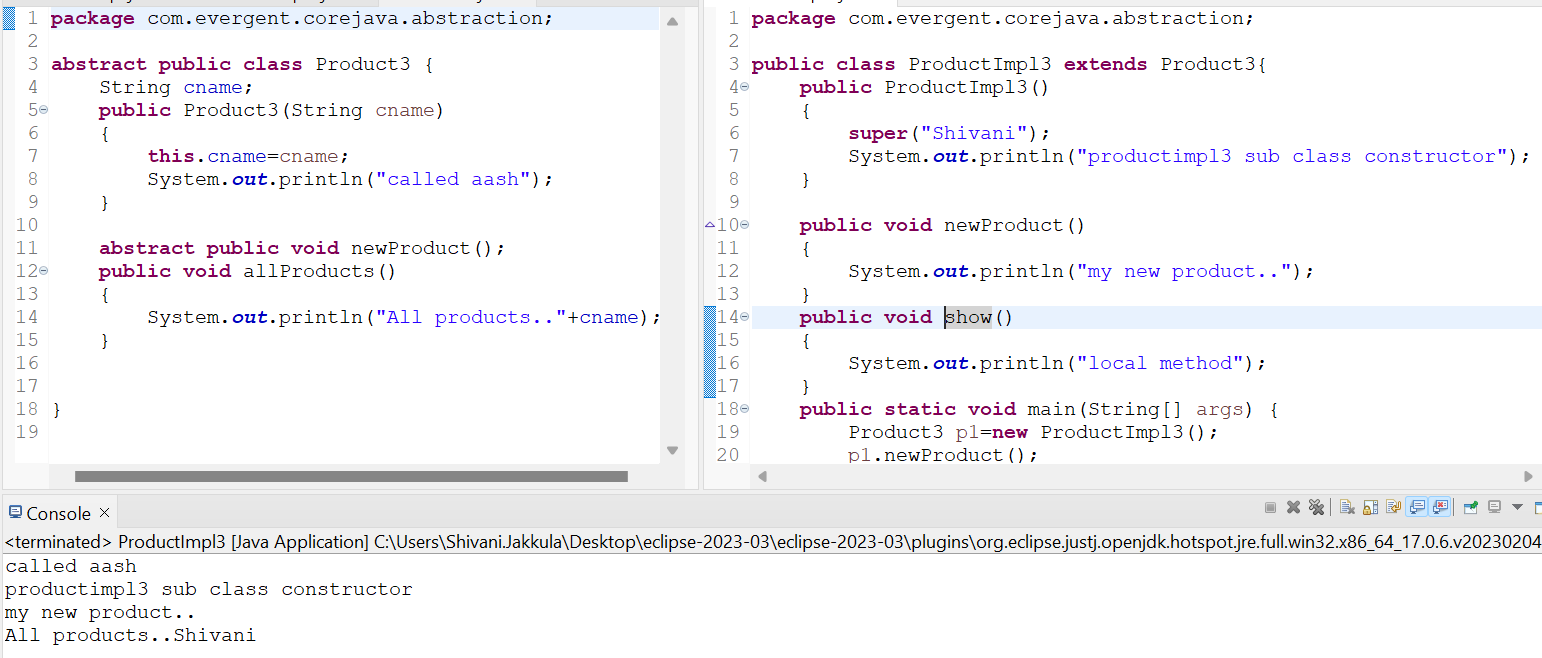
Program1:



Program2:



Program3:



**19/08/2024- Day 10**

1. Exception Handling is mechanism.
2. Exceptions are inbuilt 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 the supper class for exception and errors.
7. There are two types of exceptions in Java
8. Checked exceptions.
9. Unchecked exceptions.

8. All checked exceptions are compile time exceptions.

9. All unchecked exceptions are runtime 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 a block , if exceptions is occurred or not finally block will be execute.

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 exceptions hierarchical.

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

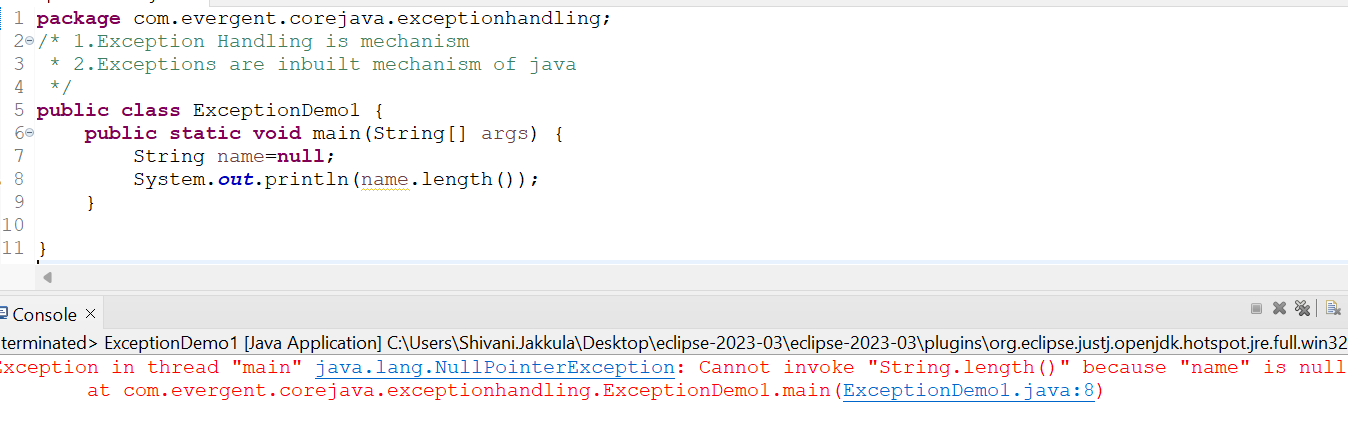
19. Our own exceptions extends exceptions or runtime exceptions.

20. All exceptions classes are into java.lang package.

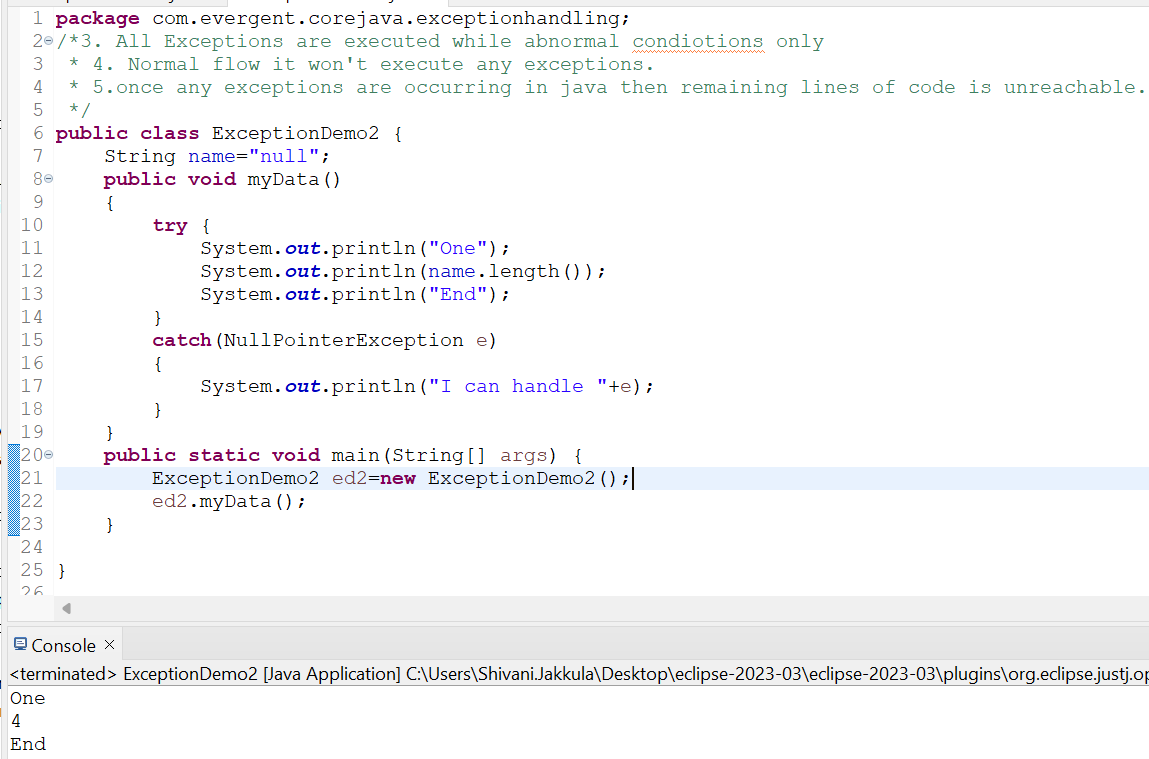
21. There is two exceptions in class, Developer should be handle one after one.

22. Developer cannot handled error.

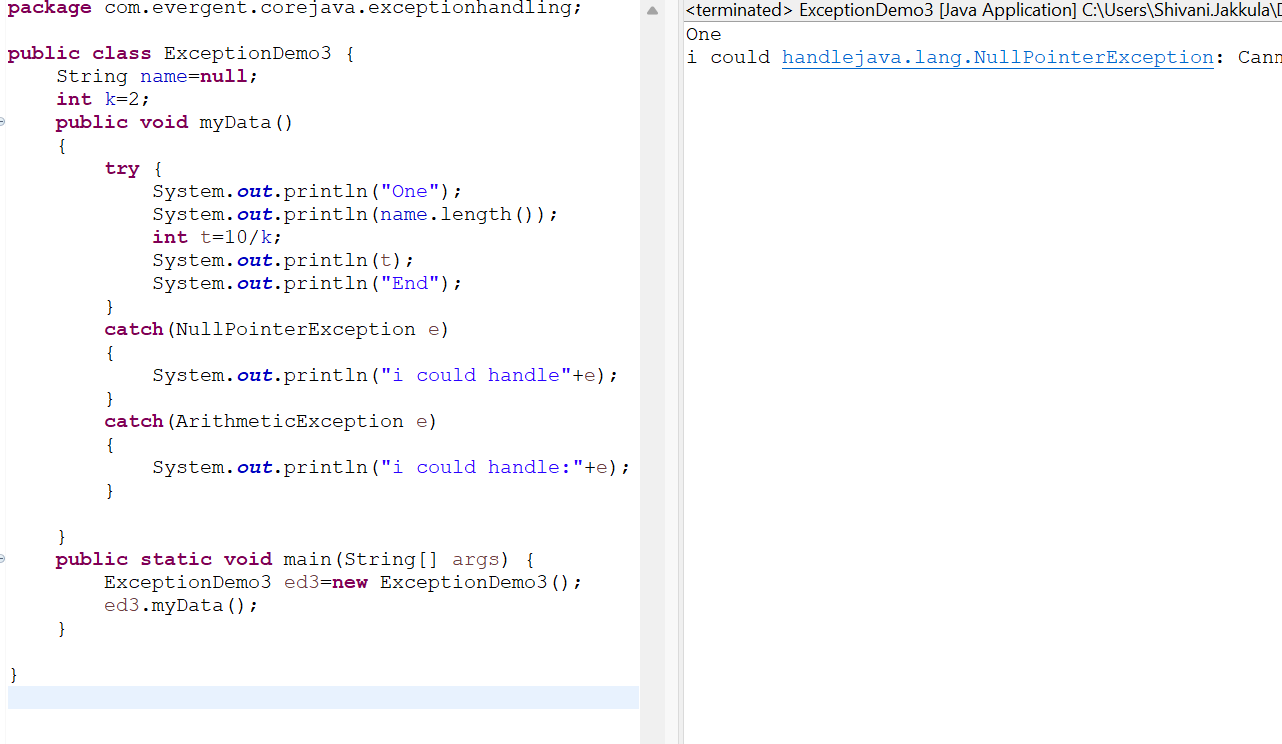
Program 1:



Program 2:



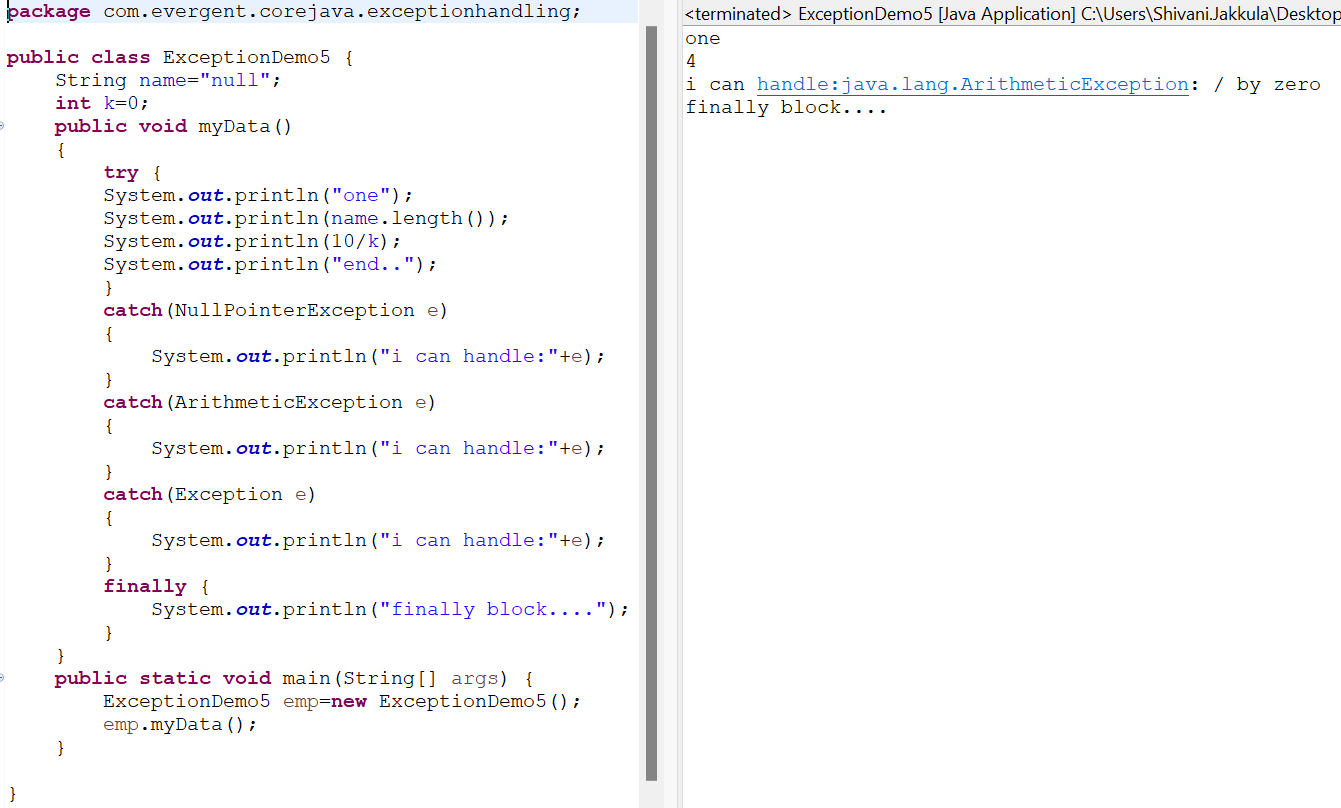
Program 3:



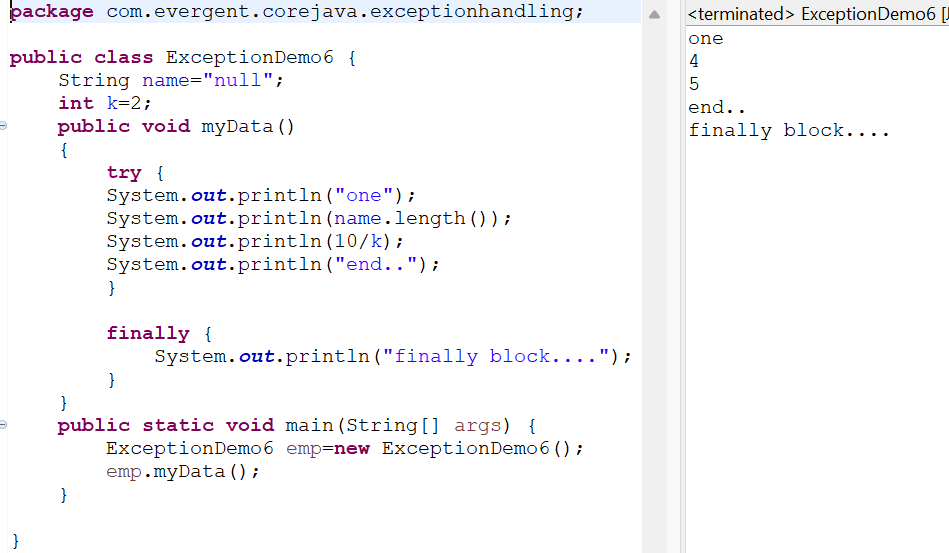
Program 4:



Program5:

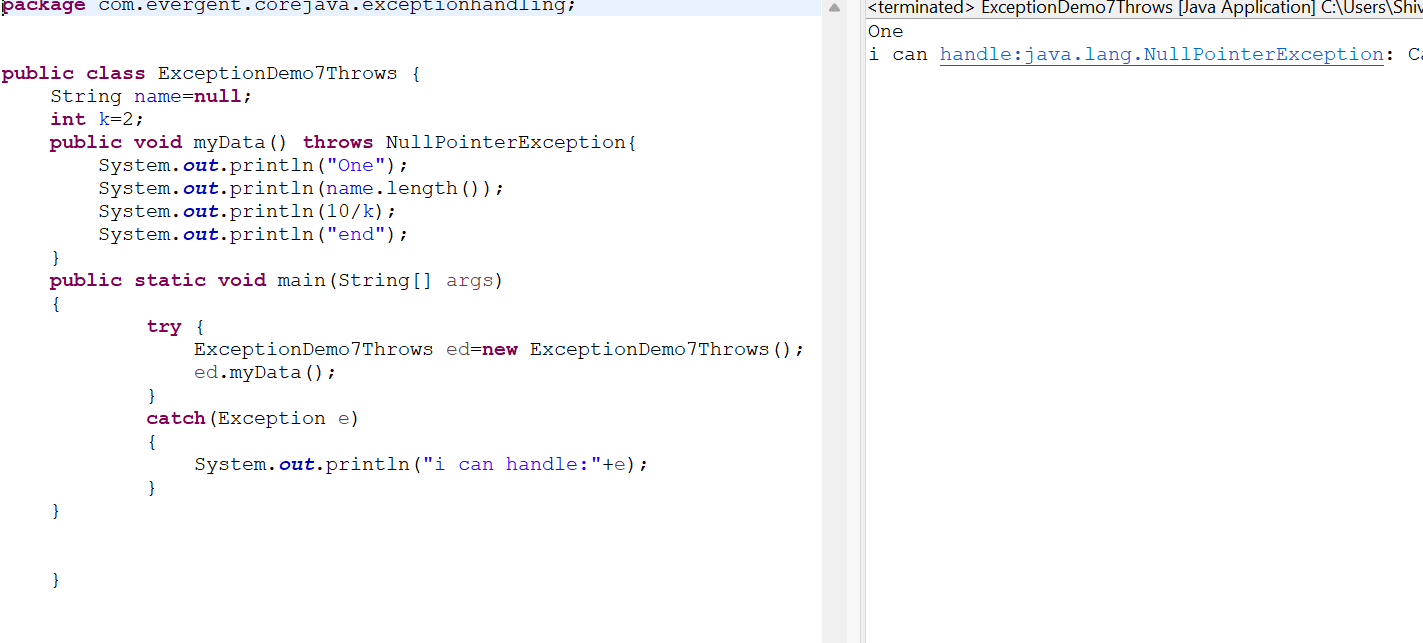


Program 6:



**20/08/2024- Day 11**

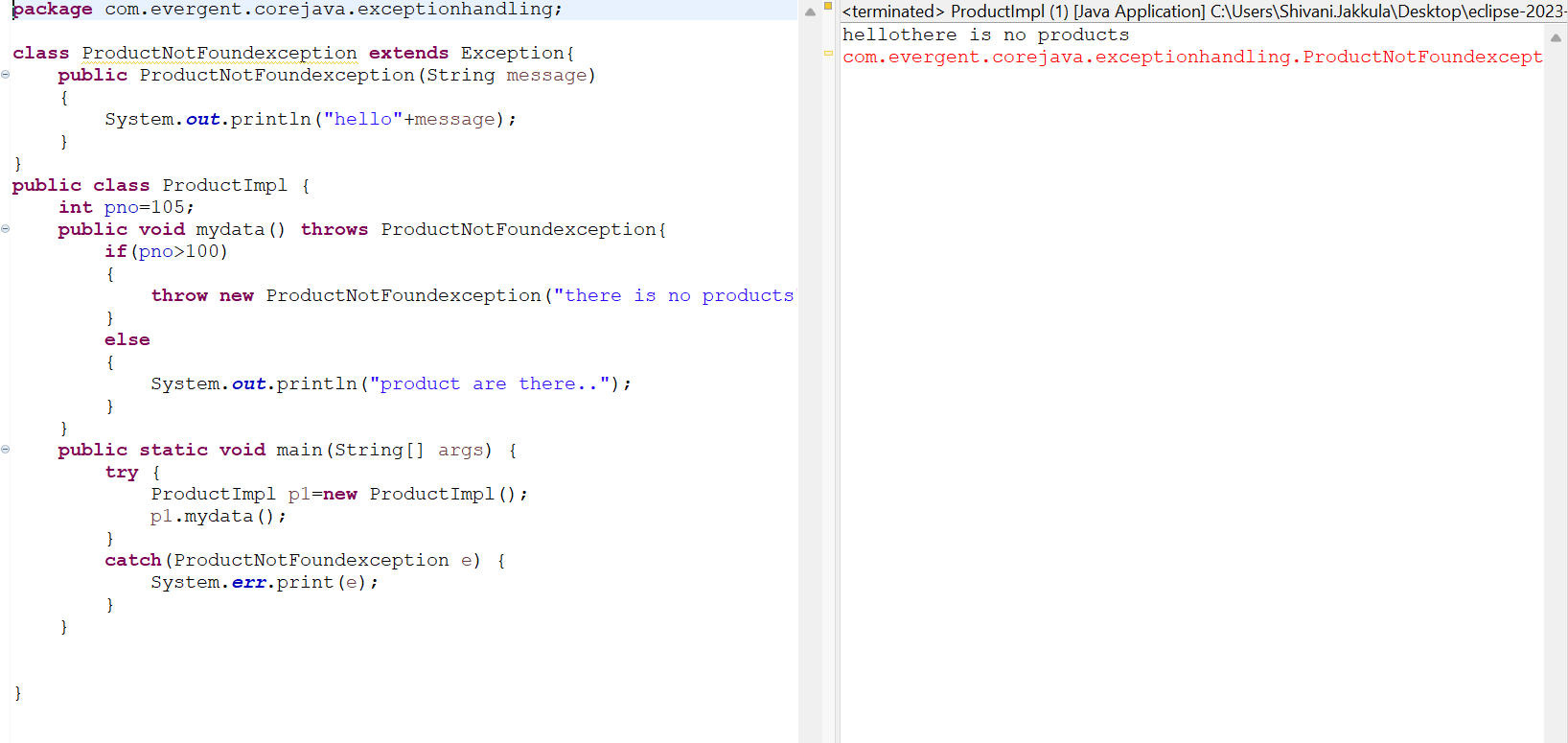
**Program 7:**

****

**Program 8:**

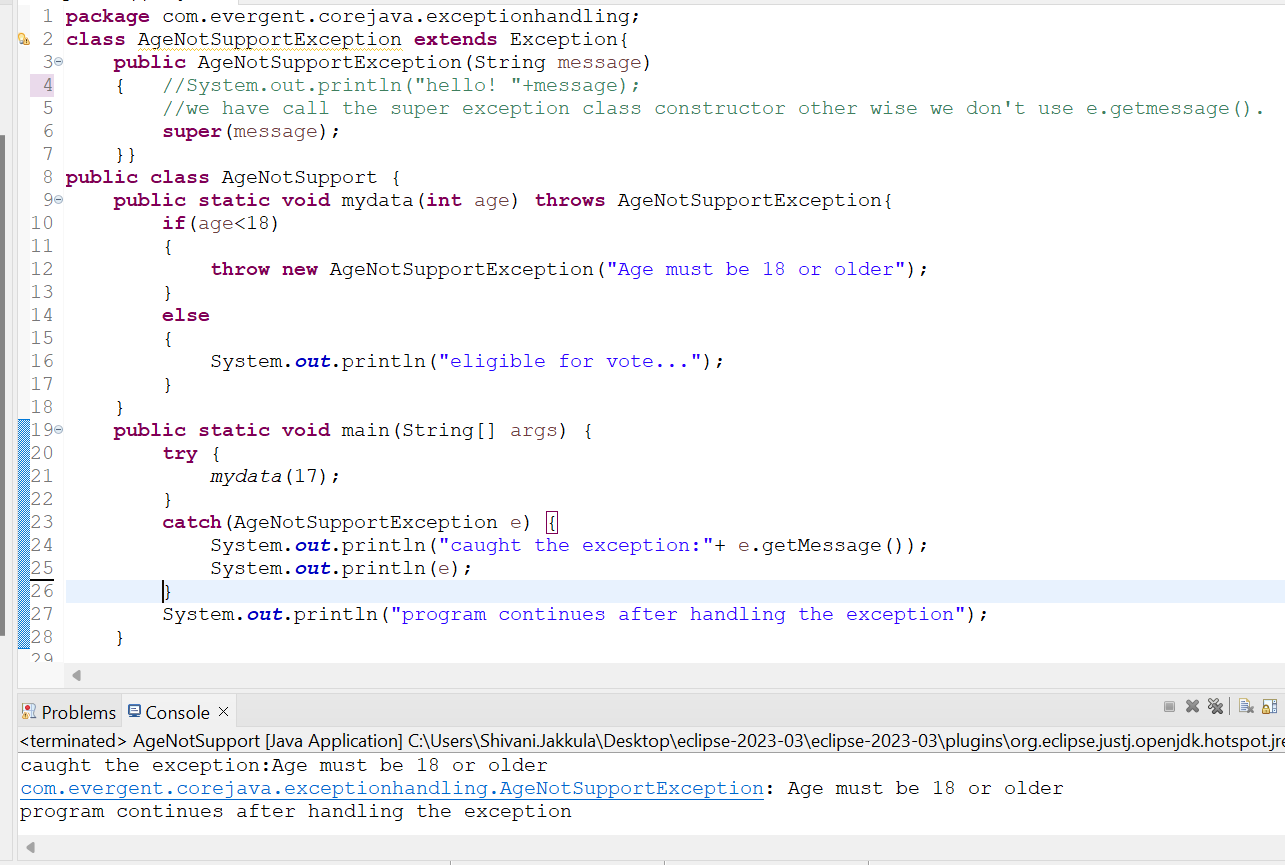
****

**Program 9:**

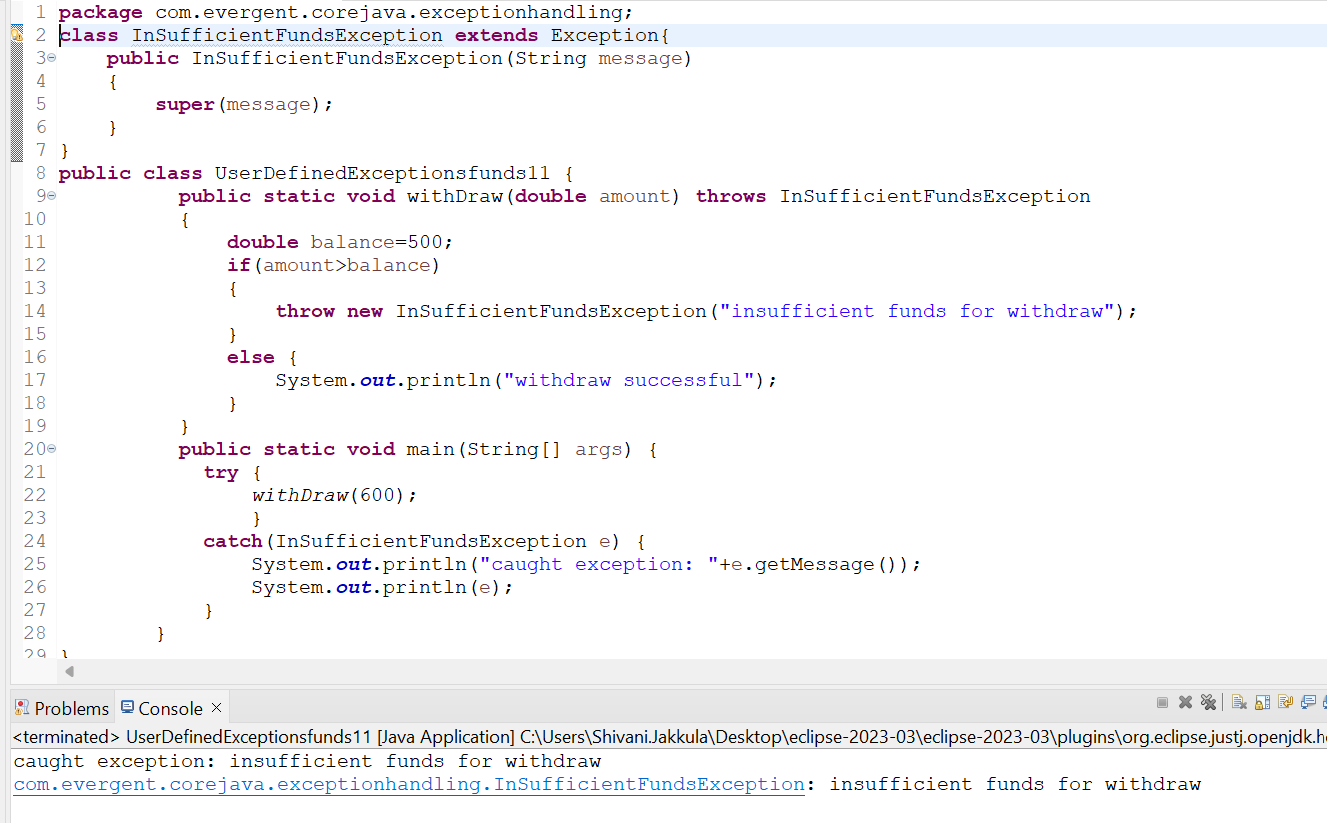
****

**21/08/2024- Day 12**

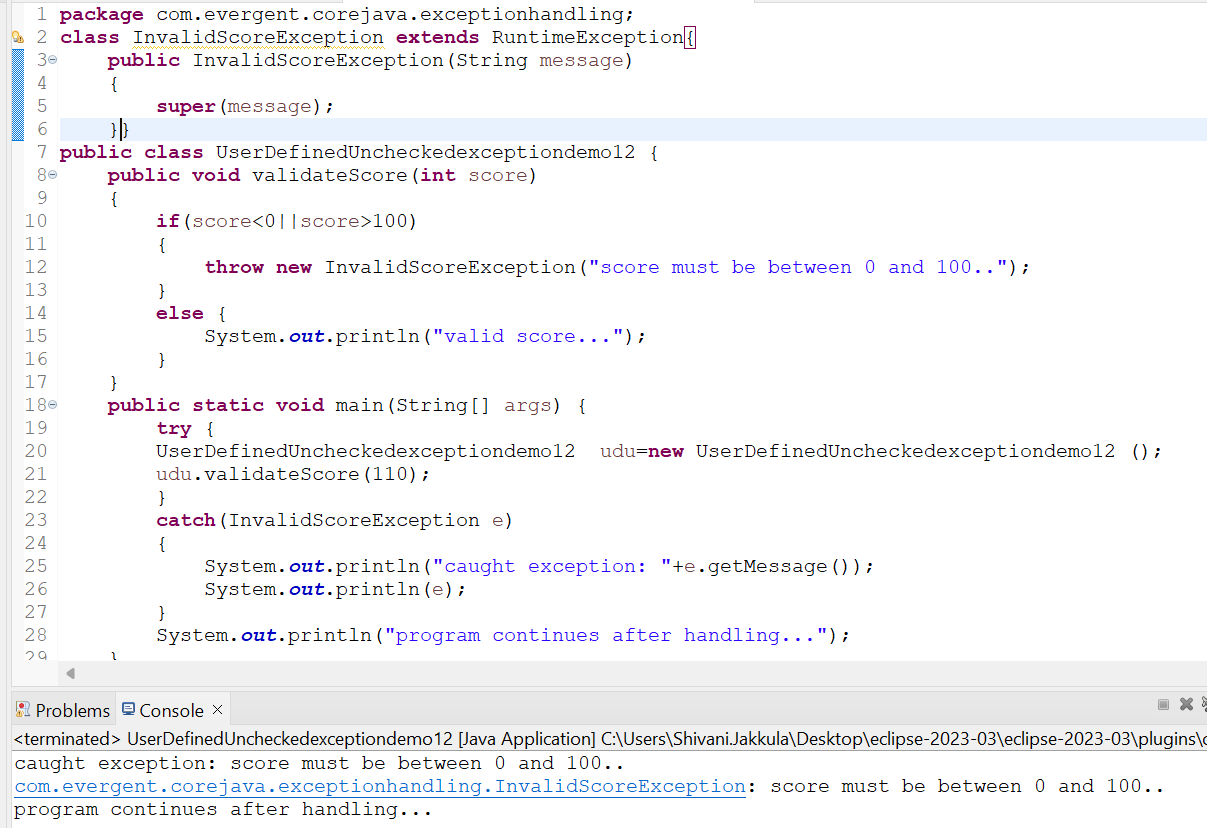
**Program 10:**

****

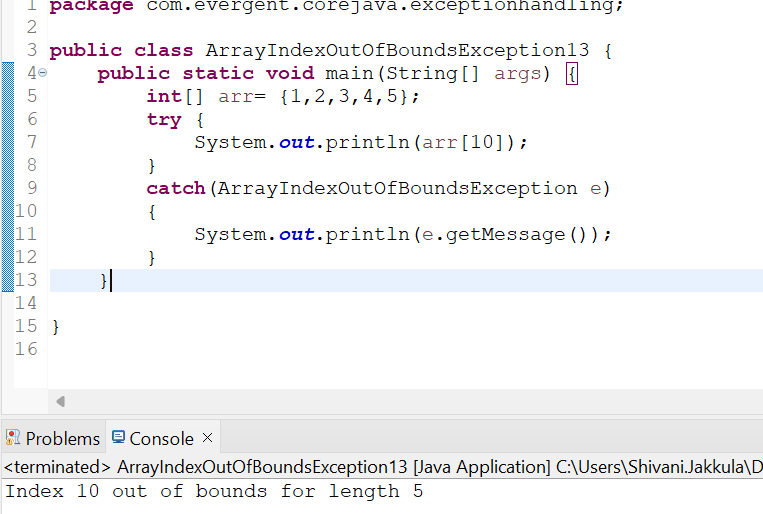
**Program 11:**

****

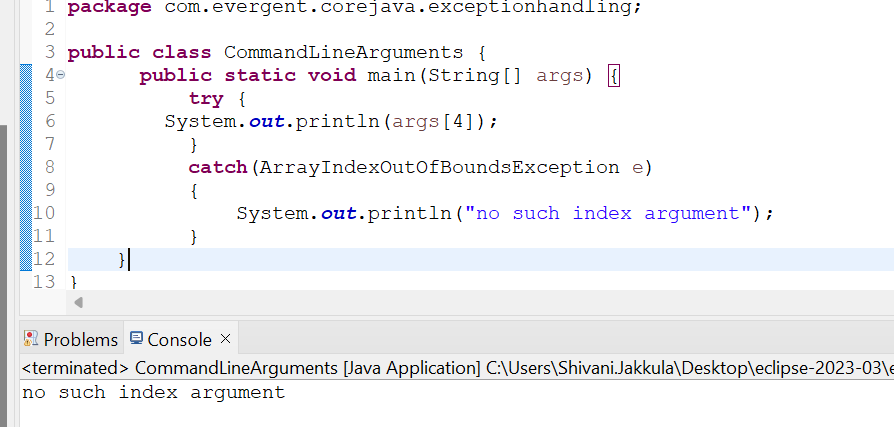
**Program 12:**

****

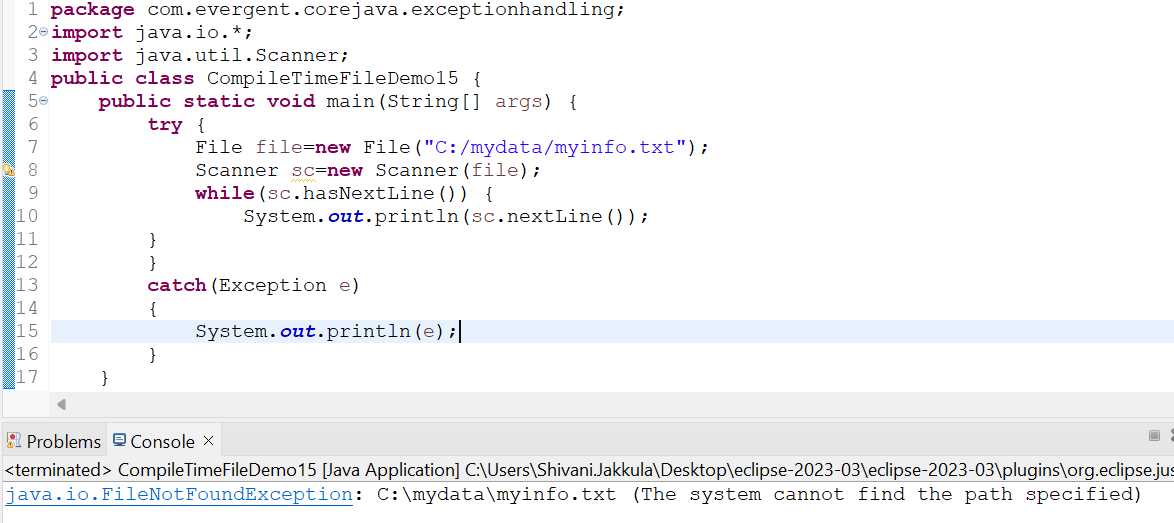
**Program 13:**

****

**Program 14:**

****

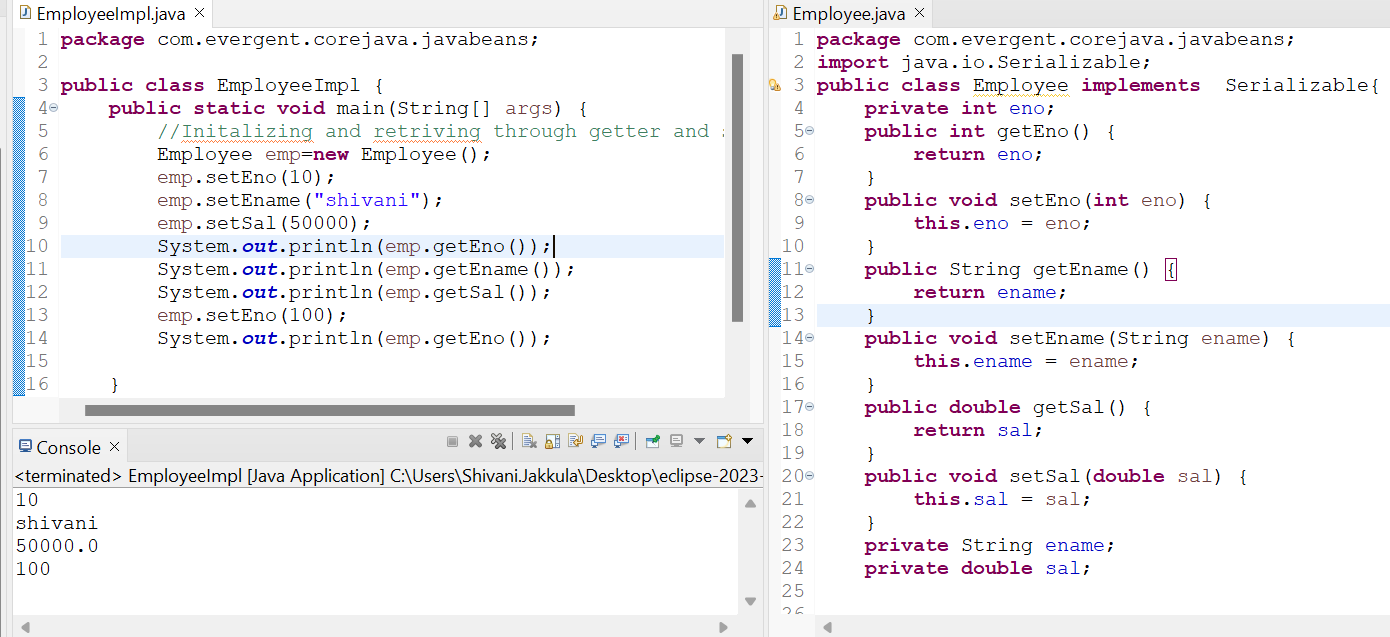
**Program 15:**

****

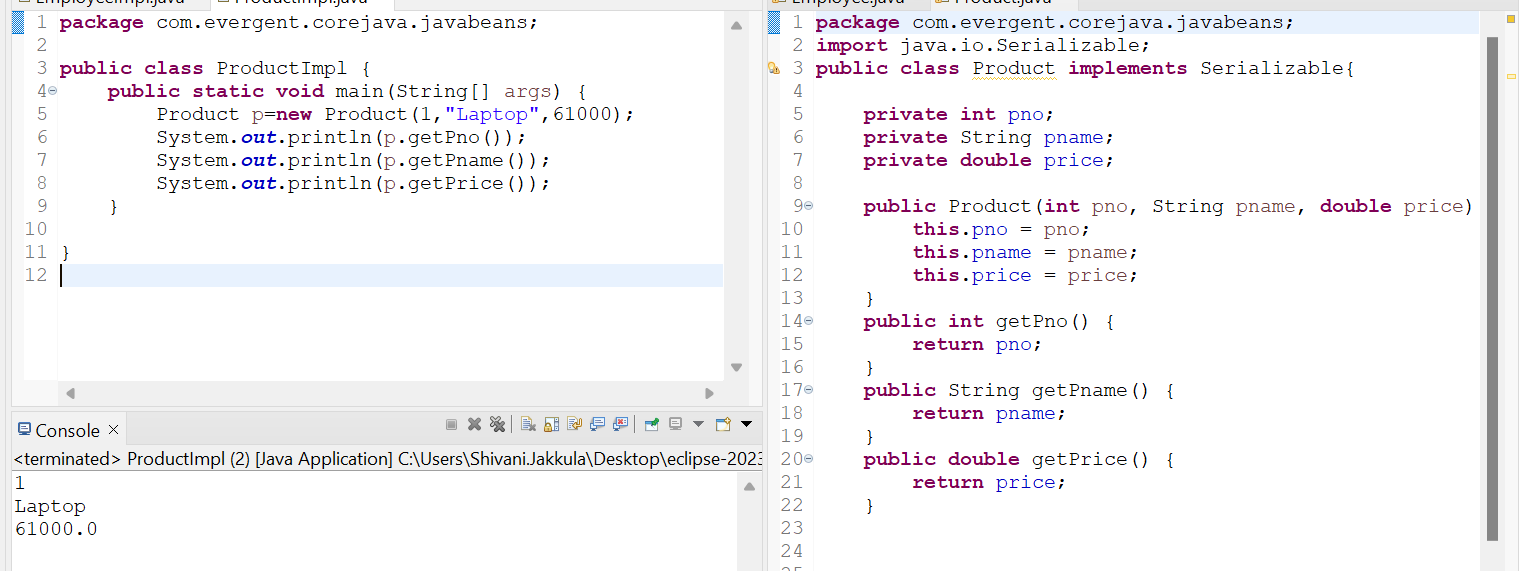
**JAVABEANS:**

* Javabeans is a mechanism.
* Javabeans is lightweight.
* All attributes are private and get/set methods are public implements java.io.Searializable interface.
* We can achieve tightly encapsulation through javabeans.

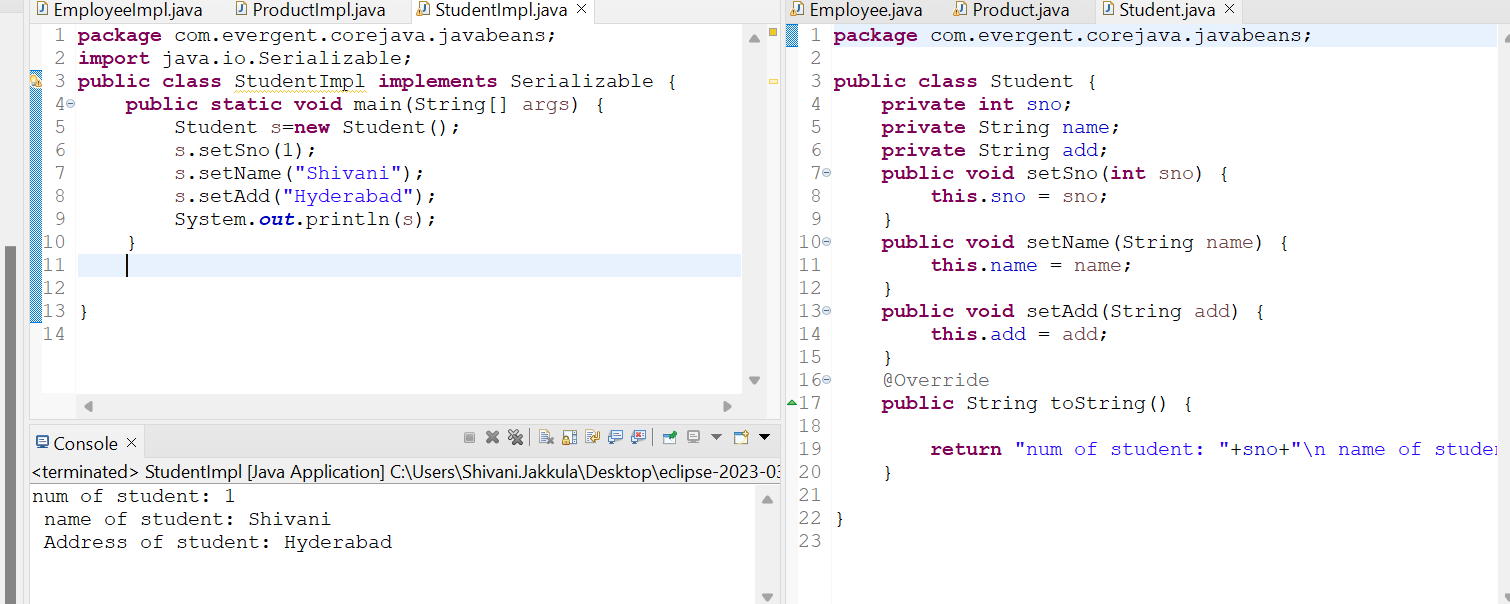
**Program1:**

****

**Program2:**

****

**Program 3:**

****