**Core Java Index**

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

1. Language and Applications
2. Java Features
3. Why Java is Platform Independent
4. Oops
5. Exception Handling
6. Multi-Threading
7. Web applications
8. Open source
9. Security
10. Supports Networking
11. Memory Management
12. JDK, JRE, JVM
13. Conditional Statements ,Operators
14. Basic Java Programming
15. Largest of Two numbers
16. Largest of Three numbers
17. Swapping of two numbers with and without temp variable
18. Packages
19. Practice program: Largest of 5 numbers

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

1. Nested loops- Pattern Programs
2. Arrays - 1D example programs
3. 2D Arrays - example programs
4. Logical Programmings
5. Switch cases
6. Java.Lang package & object class methods
7. Scanner
8. Enumeration
9. Event management Applications

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

1. Oops
2. Encapsulation
3. Calculations
4. PersonClass
5. MethodFlow
6. System.Out.Println()
7. Inheritance
8. Polymorphism
9. MethodOverloading
10. MethodOverriding
11. Abstract
12. IS-A Relation (inheritance )
13. HAS-A Relation(Object creation)

**Day -4 (08-08-2024)**

**Constructors**

-> Class name and Constructor name should be Same

-> They are 2 types of Classes

1. Default constructors
2. Parameterized constructors

-> We can Access constructors while creation of objects

-> Constructors are mainly for initializing.

-> Constructors doesn’t have any returntype not even void also.if we declare as a void the compiler will consider it as a method not as a constructors.

-> Every class needs atleast one Constructors.

->Always constructors are overloaded.

->this and super keyword.

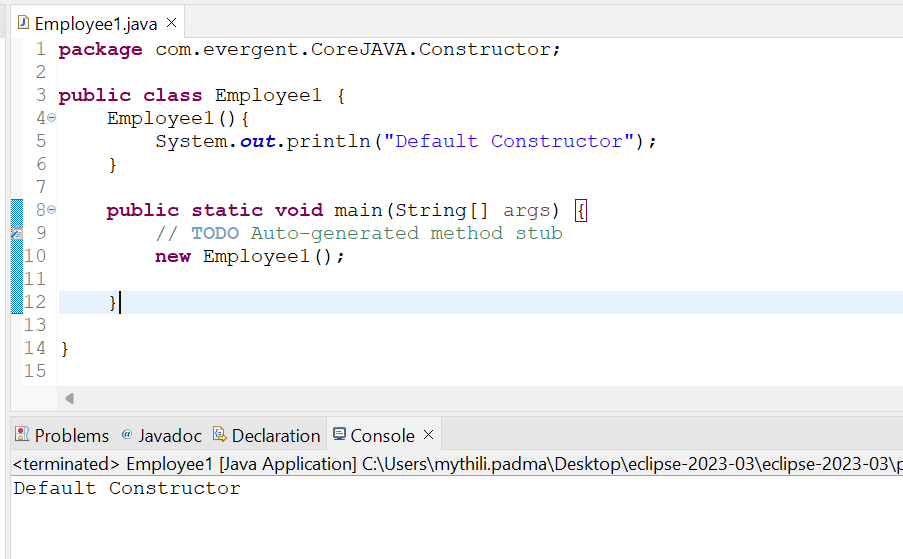
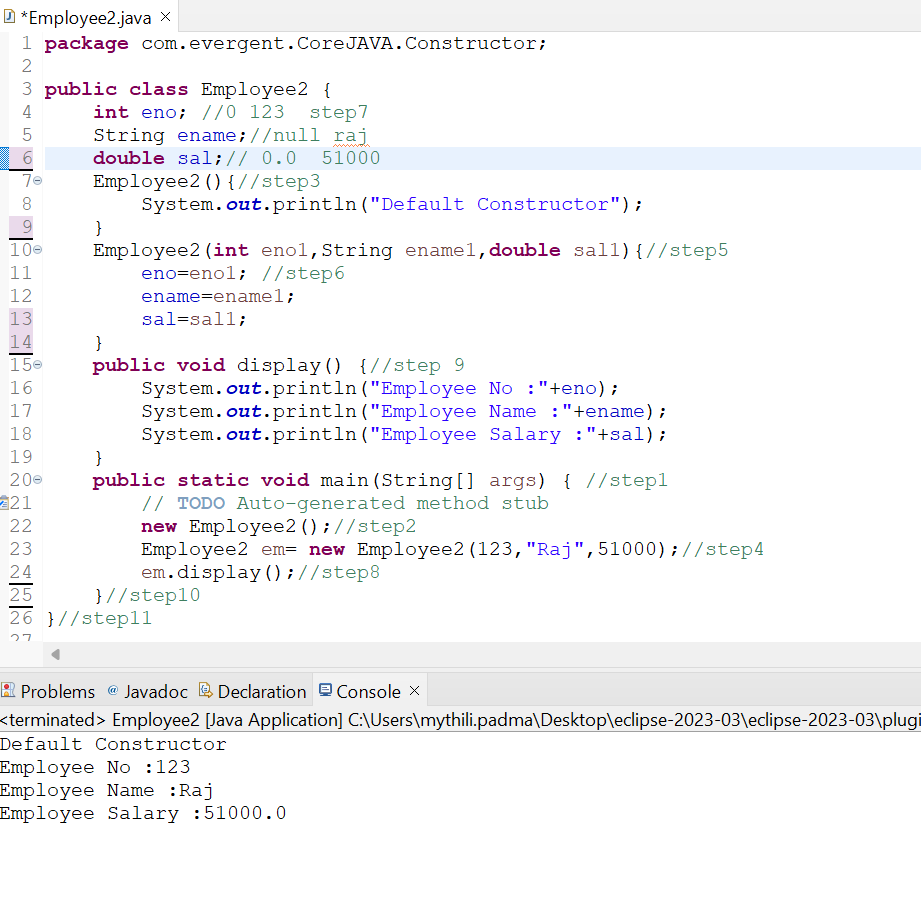
this---- keyword is always pointing to instance variable.

this can be use to call one constructor to another constructor (within a class)

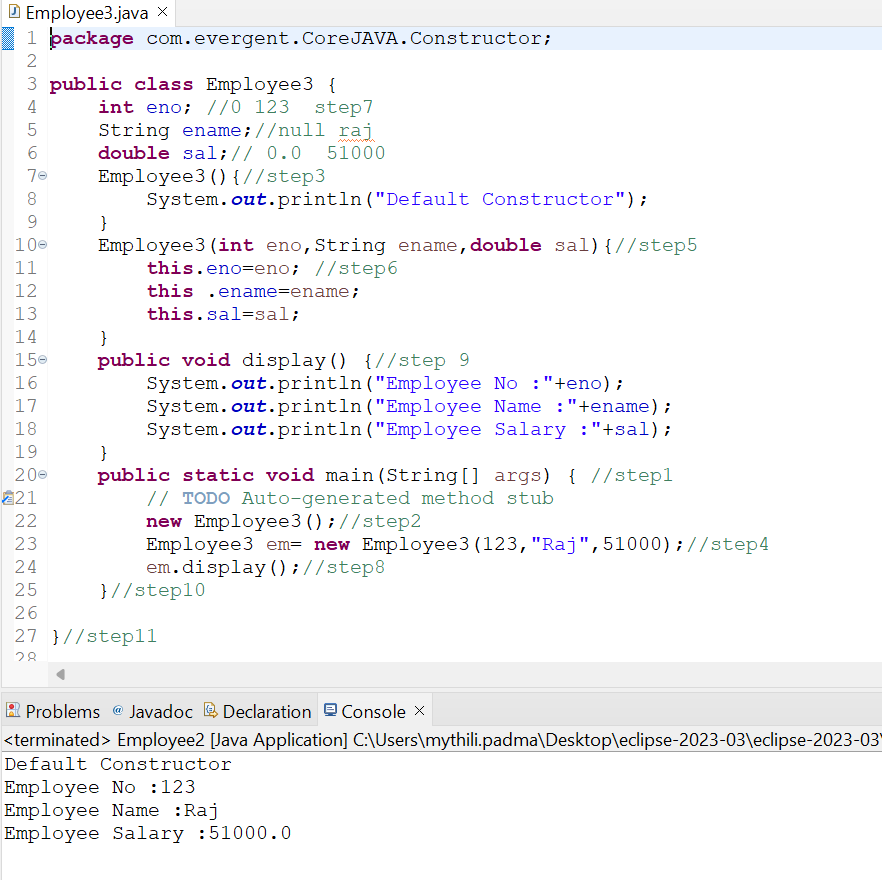
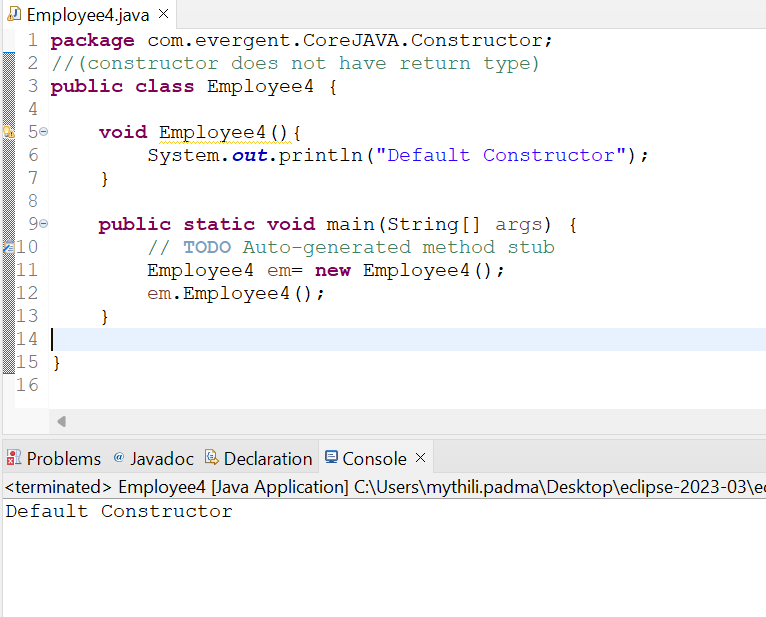
Super----- Keyword always it will call super class constructor

-> Copy Constructor ----- Object reference is copied & calls the constructor

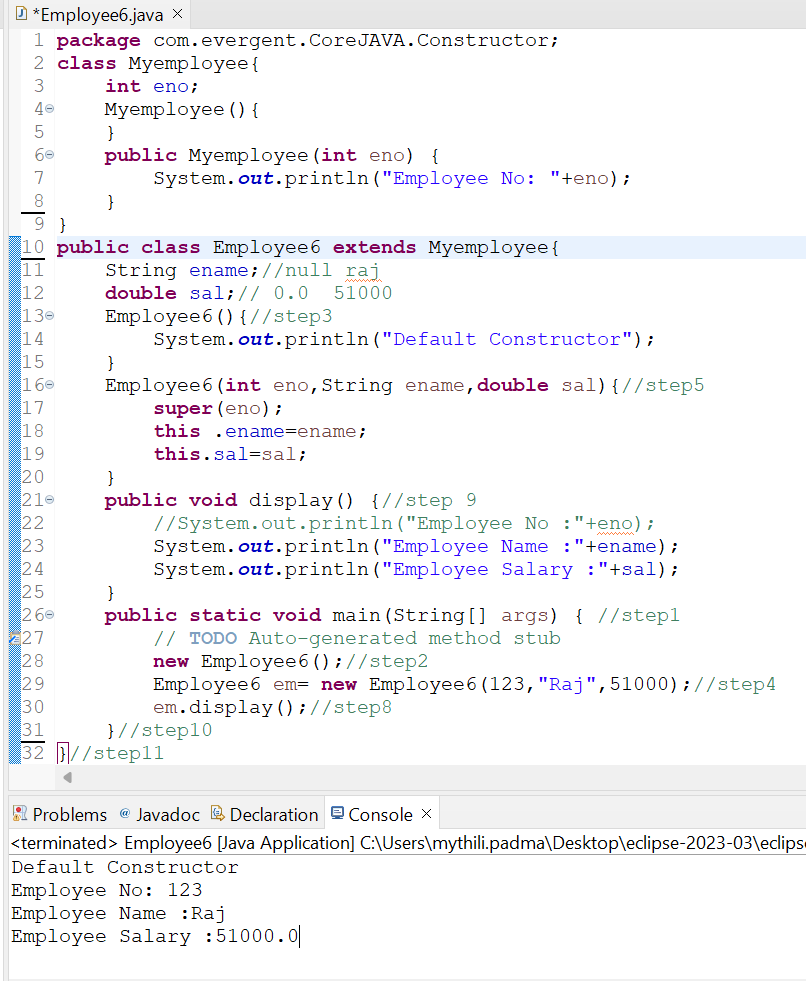
**Program 1: Program 2:**

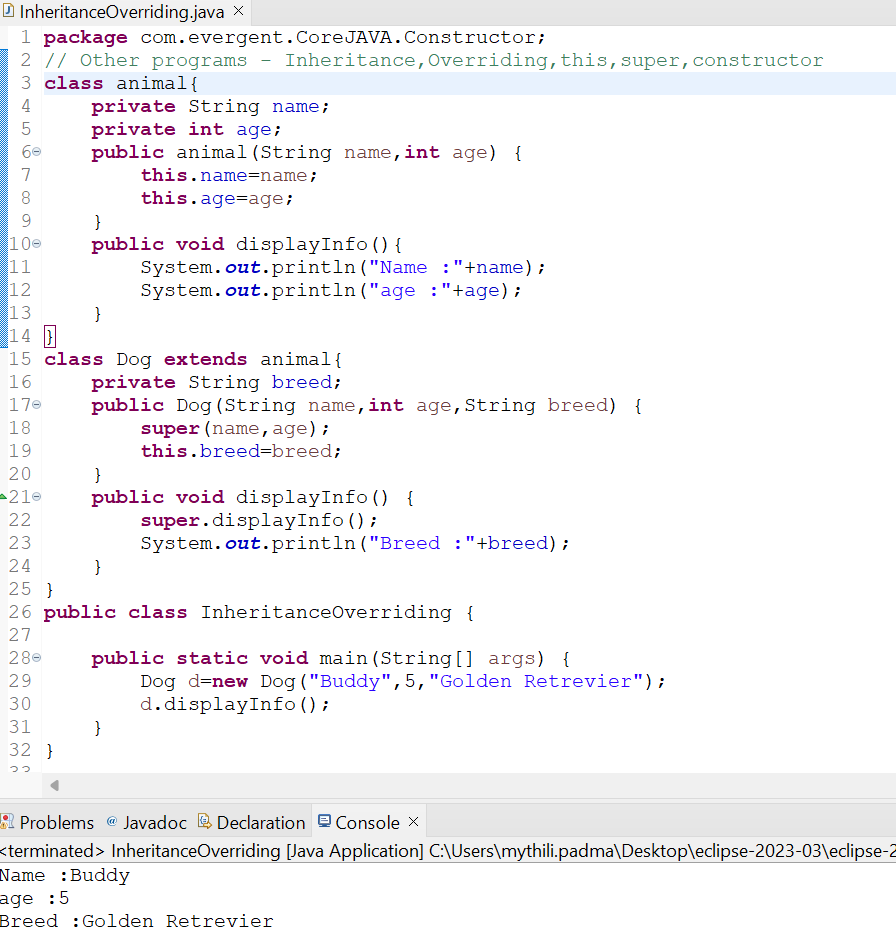
**Program 3: Program 4:**

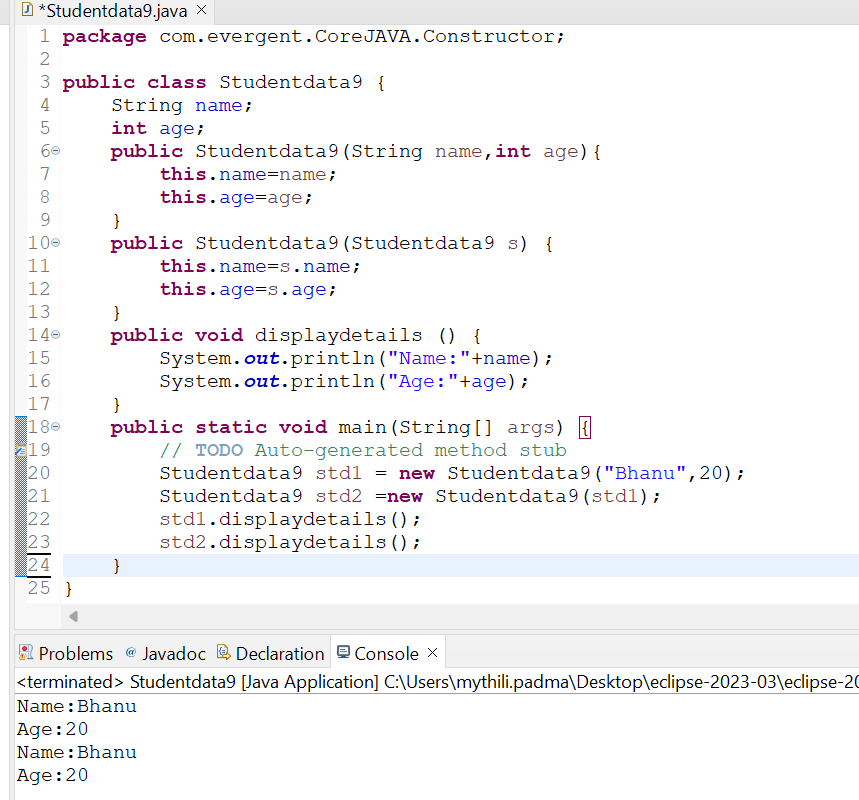
**Program 5:** **Program 6:**

**Program 7:** **Program 8:**

**Program 9:**



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

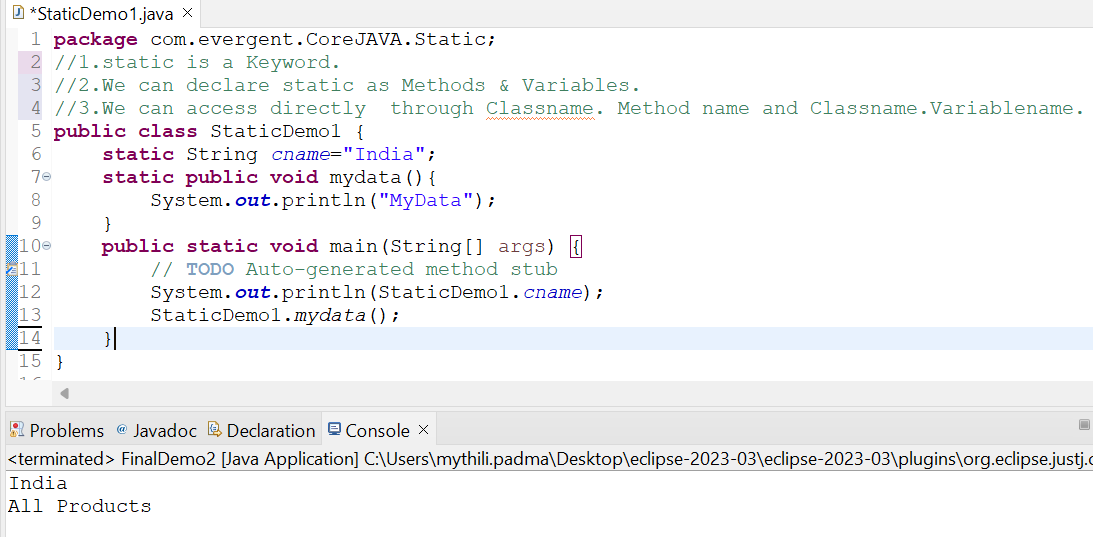
**Static :**

1. static is a Keyword.
2. We can declare static as Methods & Variables.
3. We can access static variables & static methods directly through Classname. Method name and Classname.Variablename.
4. Static methods can access static methods and static variables.
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 into JVM at that time static block is initialzed.
8. If Static variable is modified it will reflect globally

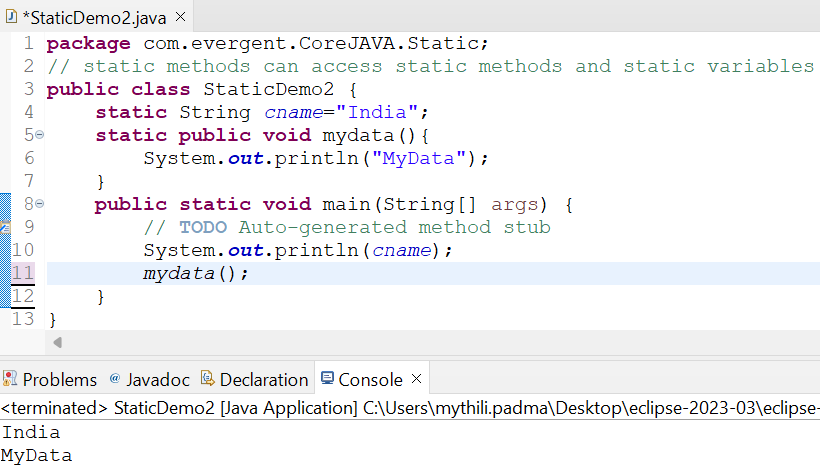
**Final :**

1. final is a Keyword.
2. We can declare final as Variables, Methods & Class.
3. Final variables can not be modify.
4. Final methods can not be Override.
5. Final Class can not be extends (Inherited).

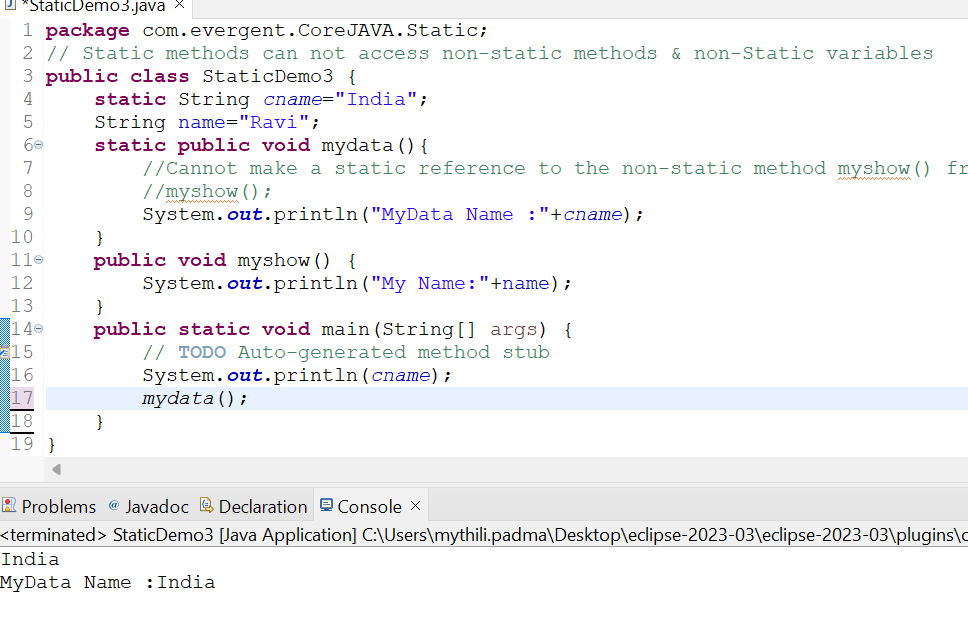
**Program 1:**



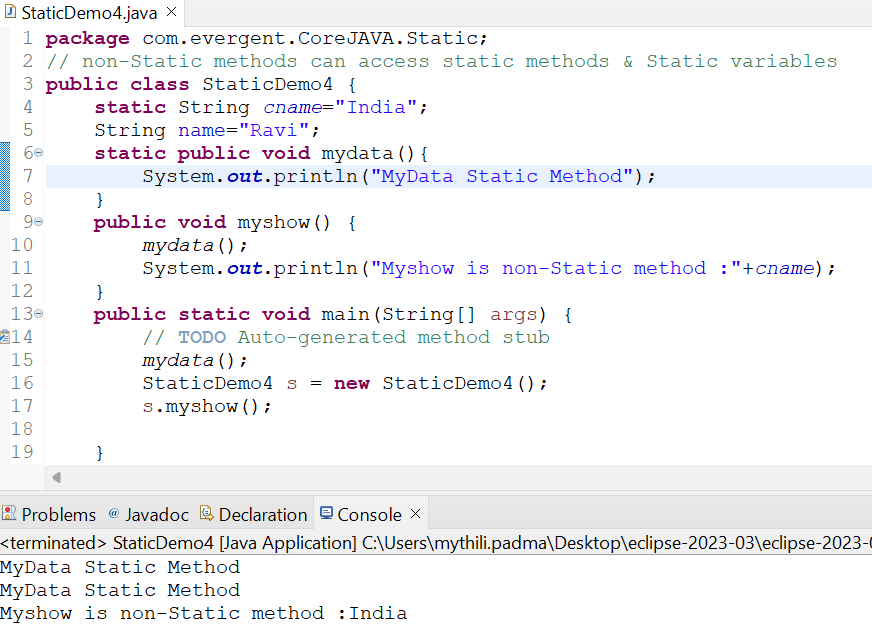
**Program 2:**



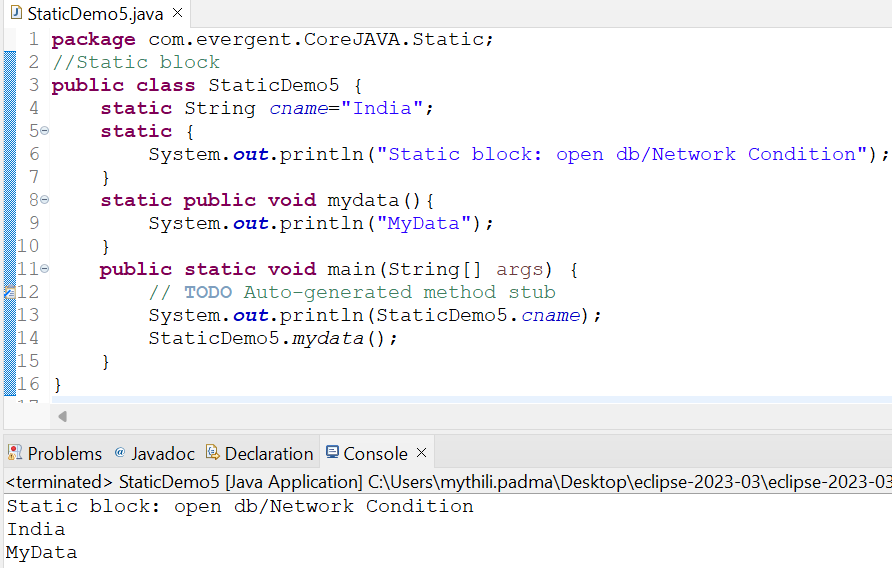
**Program 3:**



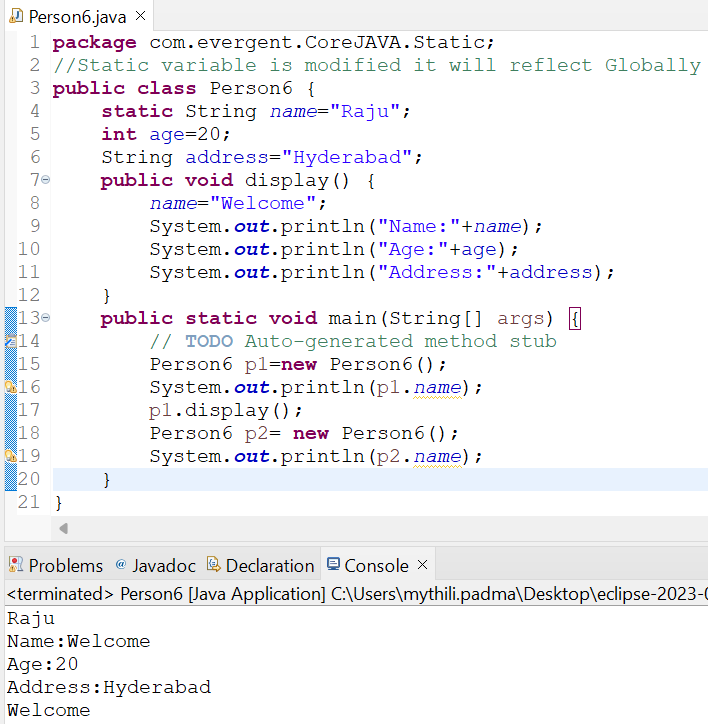
**Program 4:**



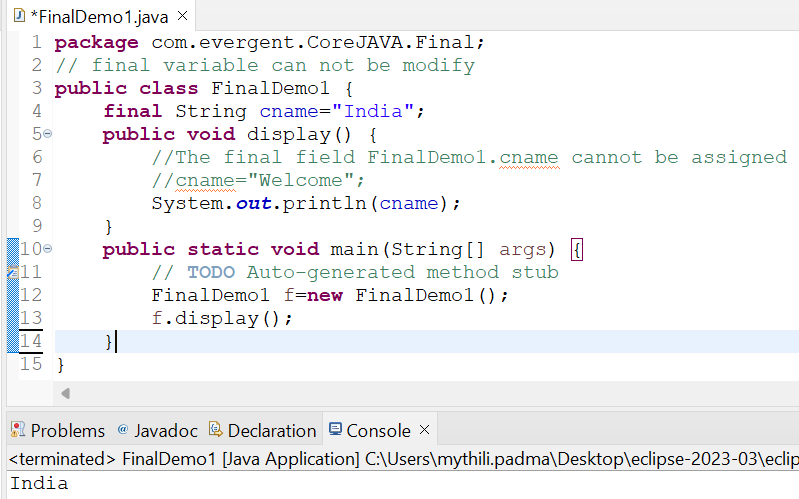
**Program 5:**



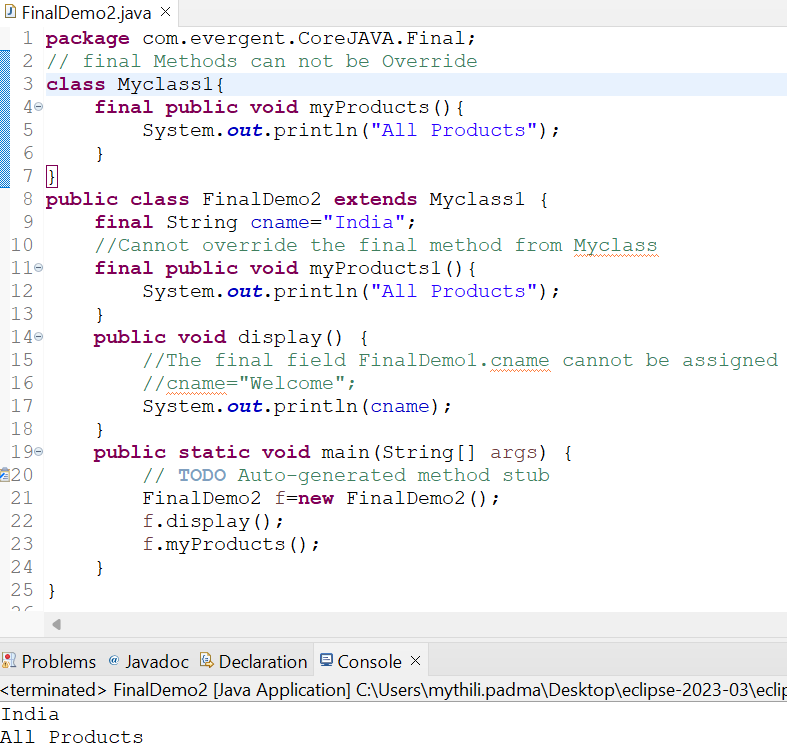
**Program6:**



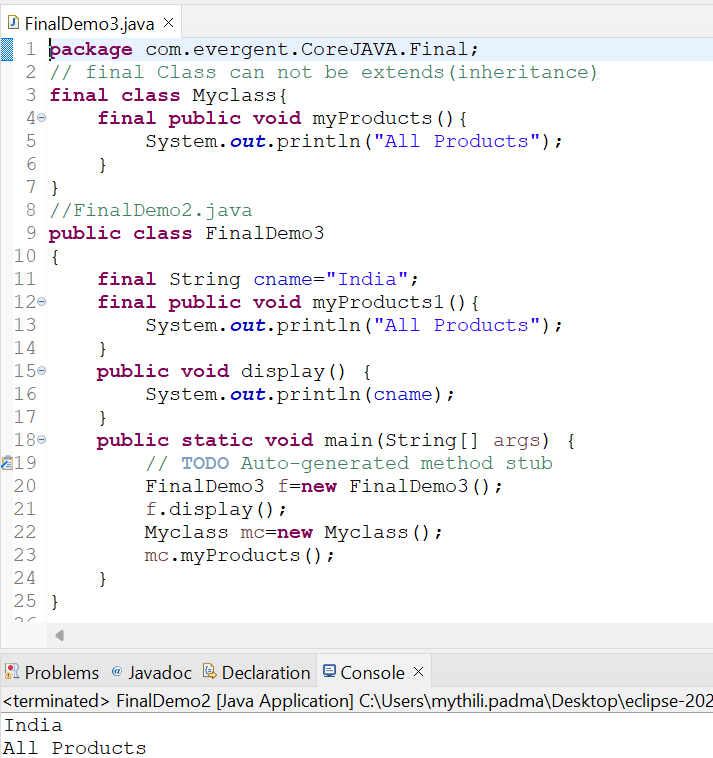
**Program 7:**



**Program 8:**



**Program9:**



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

**String :**

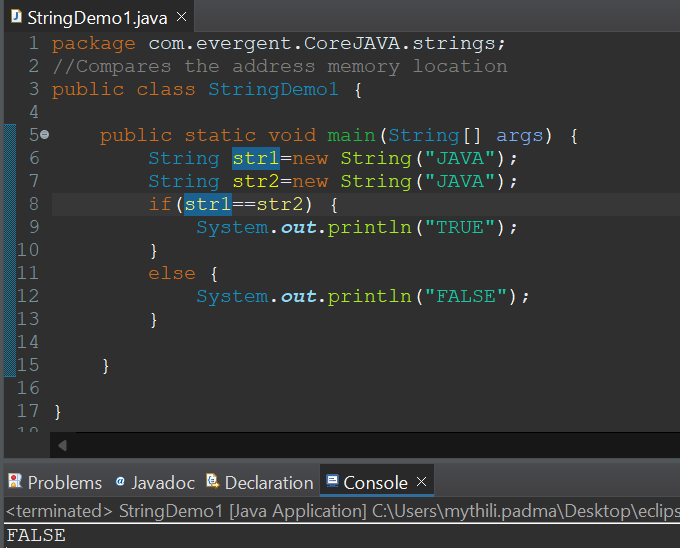
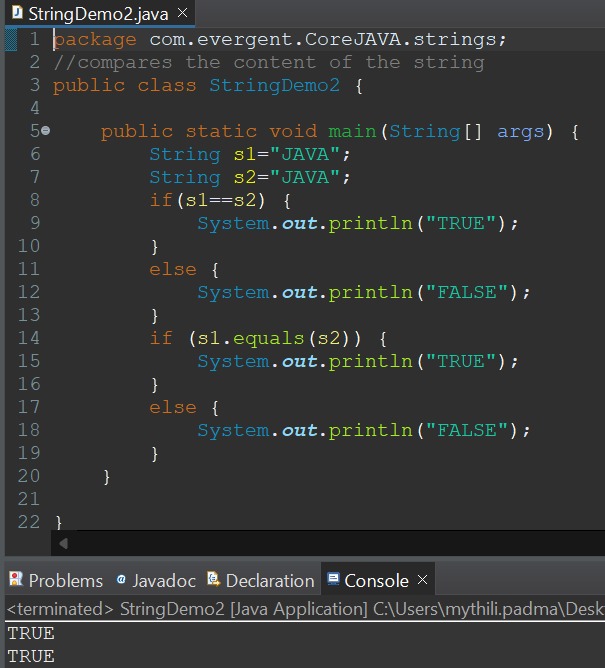
* 1. **String class**  
      -String is Final Class  
      -Strings are immutable  
      -Why strings are immutable?  
      Once we declare any string object, it is constant. If we are trying to modify an existing string it will create another memory location, the existing object is eligible for Garbage Collection.  
      -String class having methods  
      -All string class methods are non-synchronized  
     **1.2 String Buffer**  
      -String is Final Class  
      -Strings are mutable  
      -String class having methods  
      -All string class methods are synchronized

-It is not recommended to use in development but still in API.

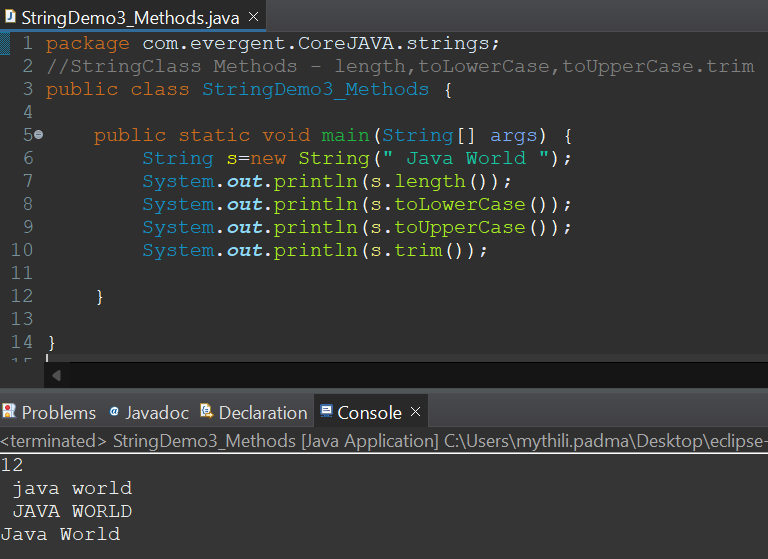
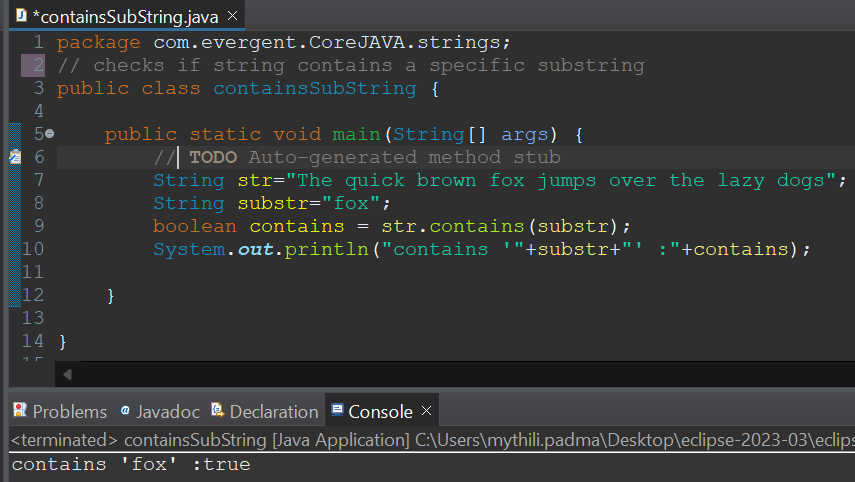
-Methods --> append(),insert(),delete(),replace(),reversed(),capacity(),length()  
**1.3 String Builder** -String is Final Class  
 -Strings are mutable  
 -String class having methods  
 -All string class methods are non-synchronized

-Methods --> append(),insert(),delete(),replace(),reversed()

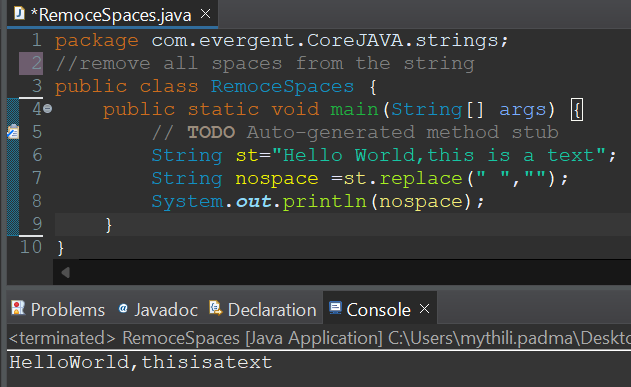
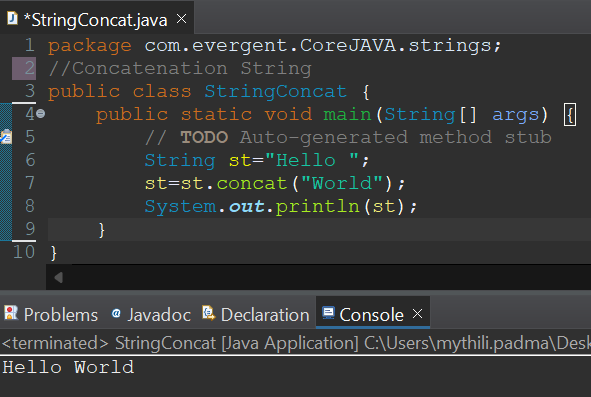
**Program 1: program 2:**

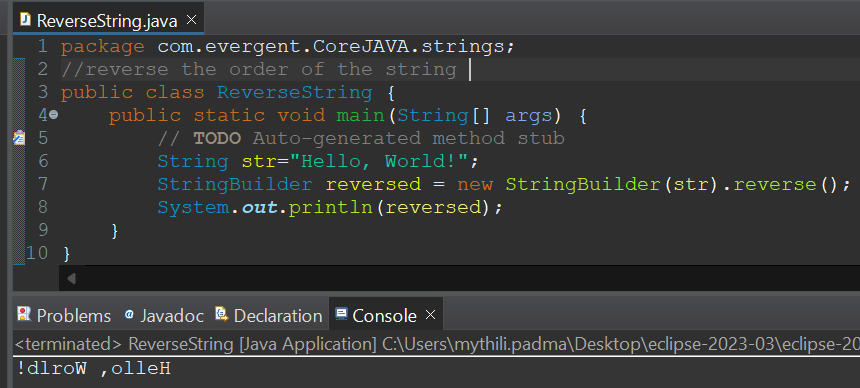
**Program 3: Program 4:**

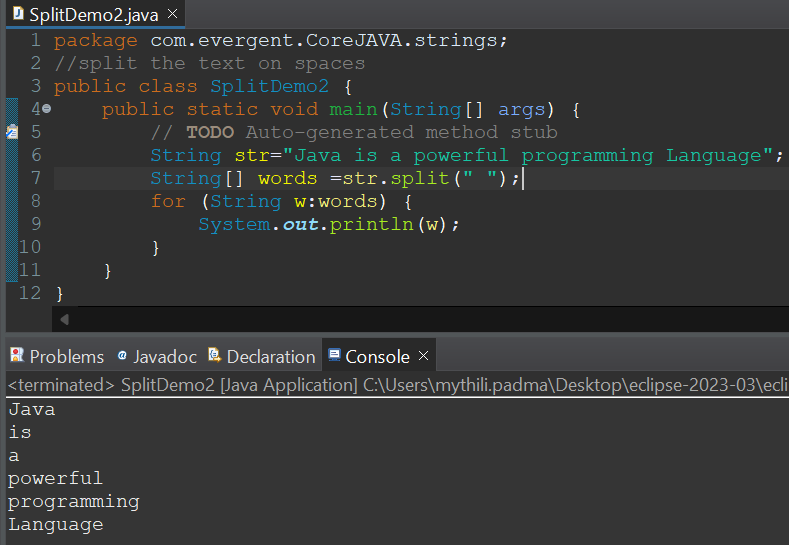
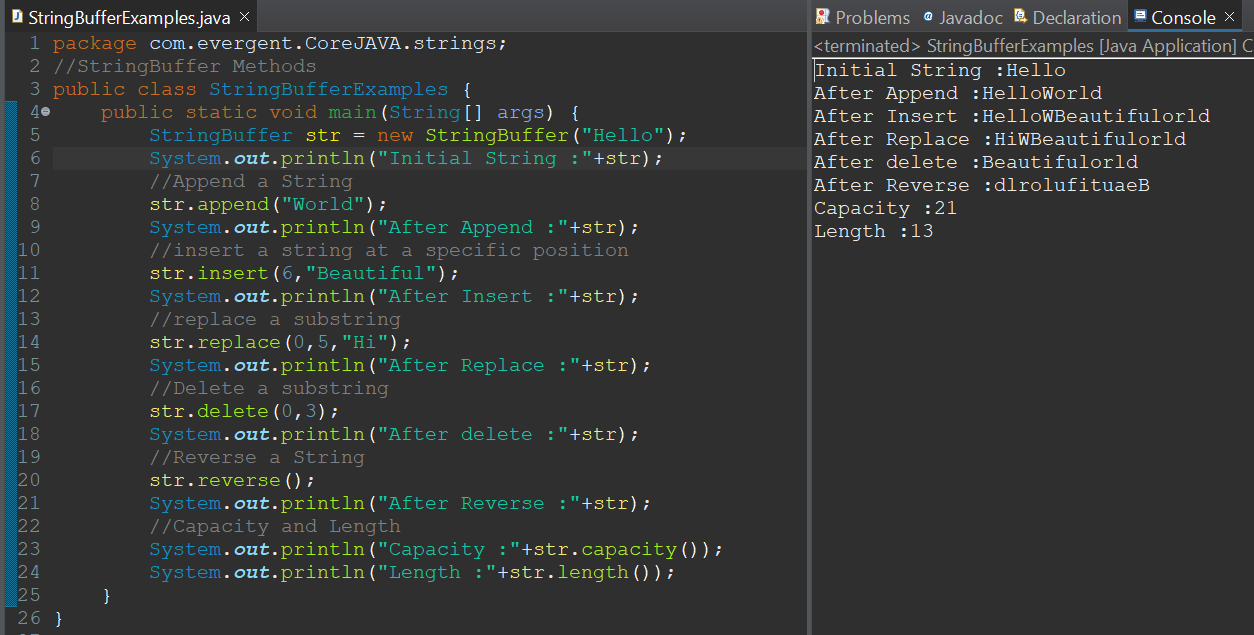
**Program 5: Program 6:**

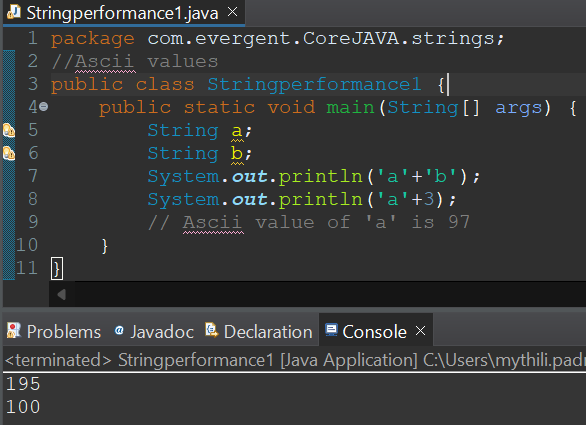
**Program 7: Program 8:**

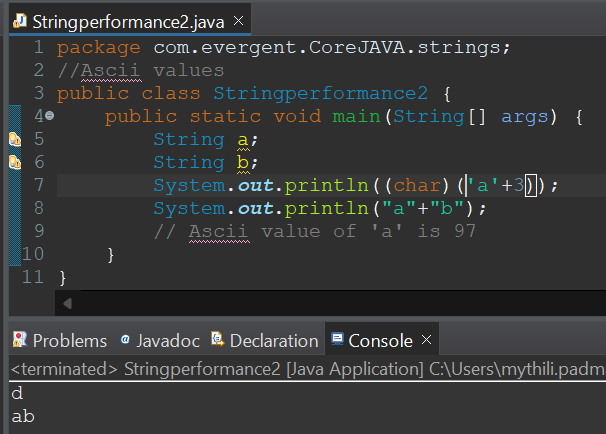
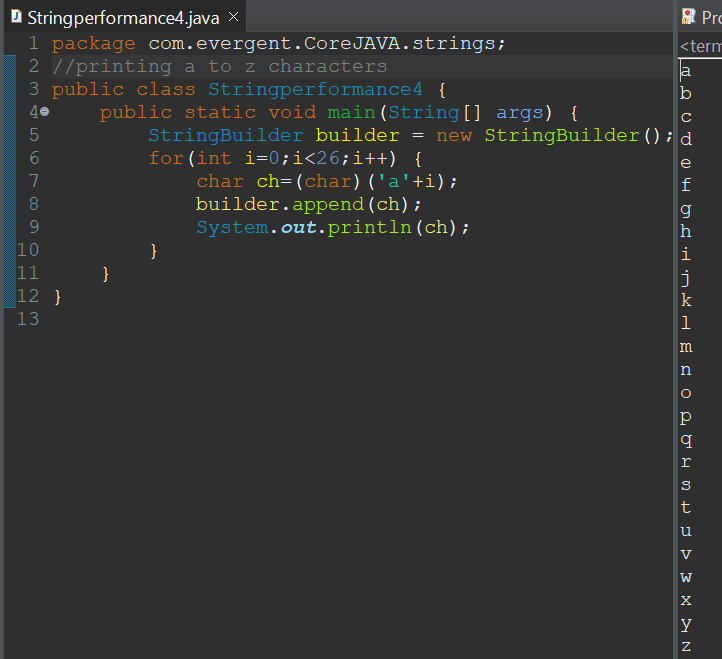
**Program 9: Program10:**

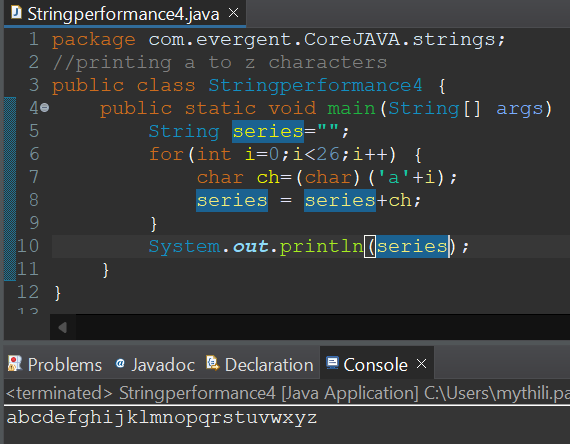
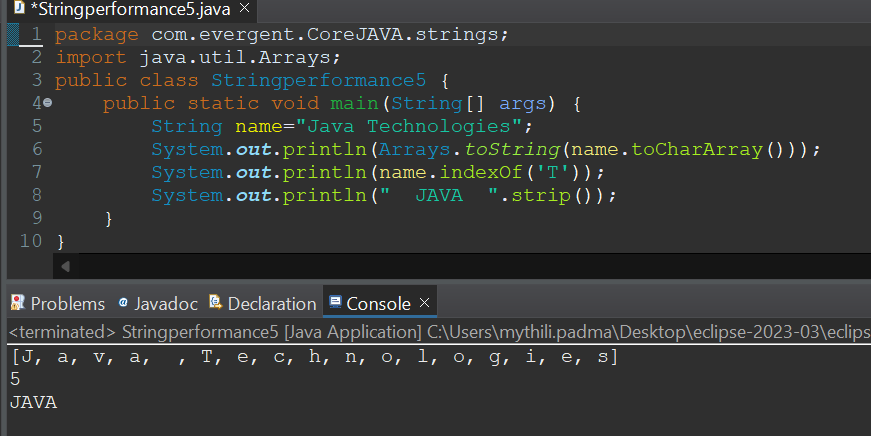
**Program 11: Program 12:**

**Program 13: Program 14:**

**Program 15: Program 16:**

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

* **Immutable**

1.We can create our own immutable class?

a.We can declare class as final.

b.The class is declared as final so that it cannot be sub classed..

c.Private final fields:

i.The fields name and age are private and final constructor.

d.Constructor:

ii.The constructor initializes the final fields when the person object is created.

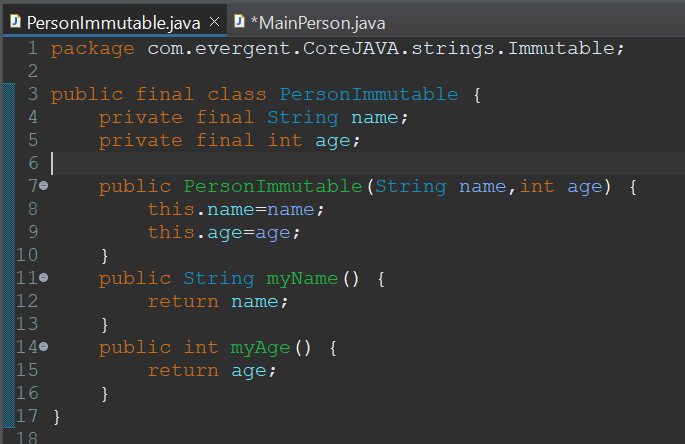
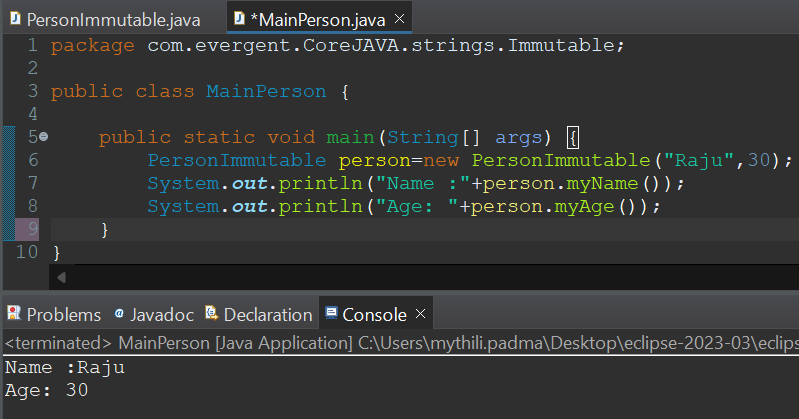
* **Object Class Methods :**

1. toString()
2. hashCode()

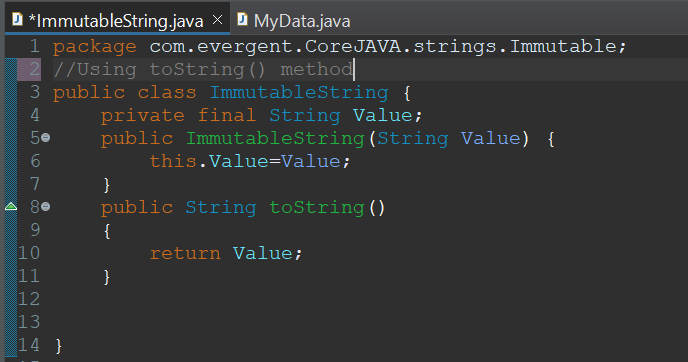
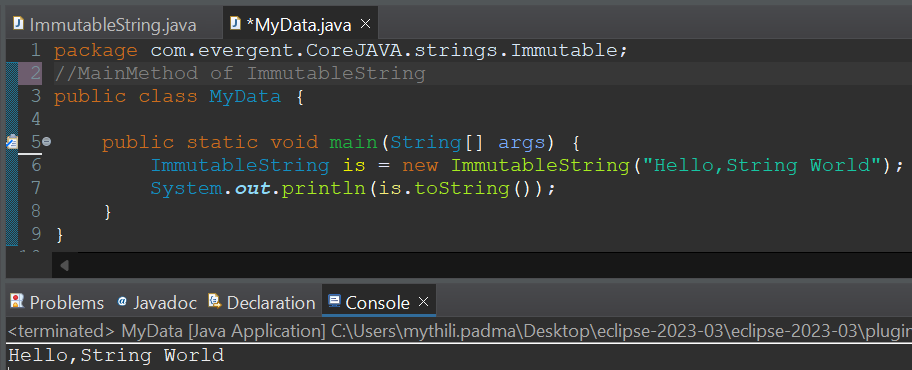
* I**nterfaces:**

1. Interface is a keyword.
2. We can declare methods signatures only but not implementations.
3. By default all interface methods are abstract.
4. If any class implements interface that class should override all interface methods otherwise the class will be showing compile time error.
5. We cannot create objects to interface, but we can create references to interface.
6. We can declare variables inside the 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.

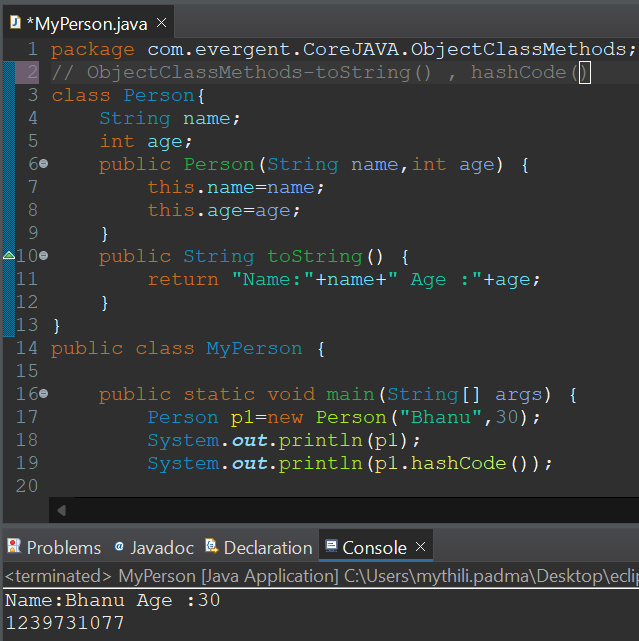
**Program 1:**

** ---> **

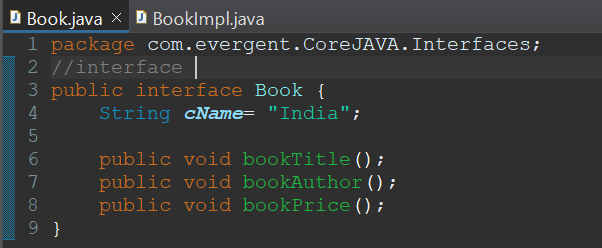
**Program 2:**

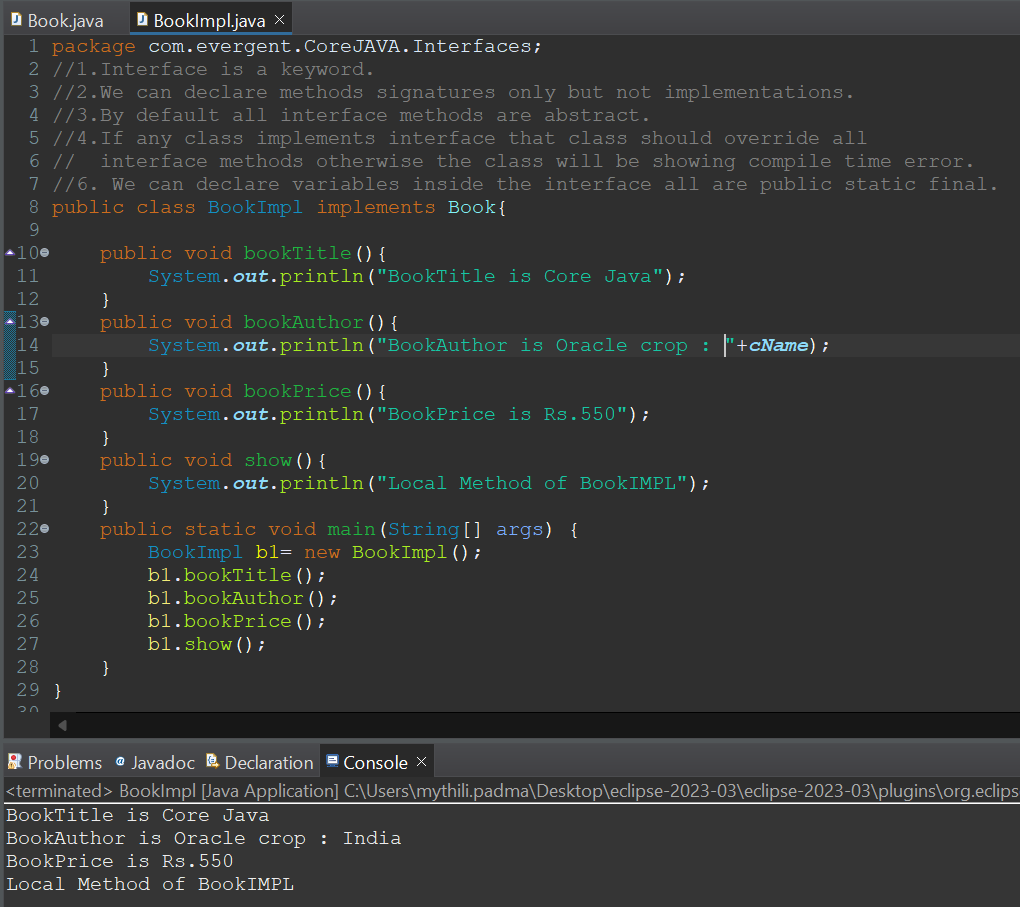
** ---> **

**Program 3:**

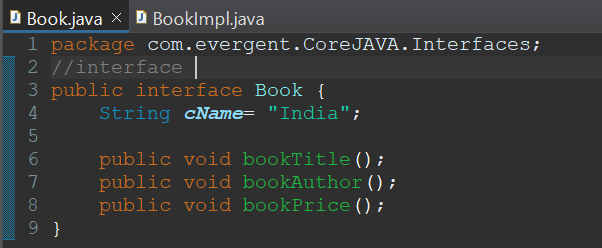
****

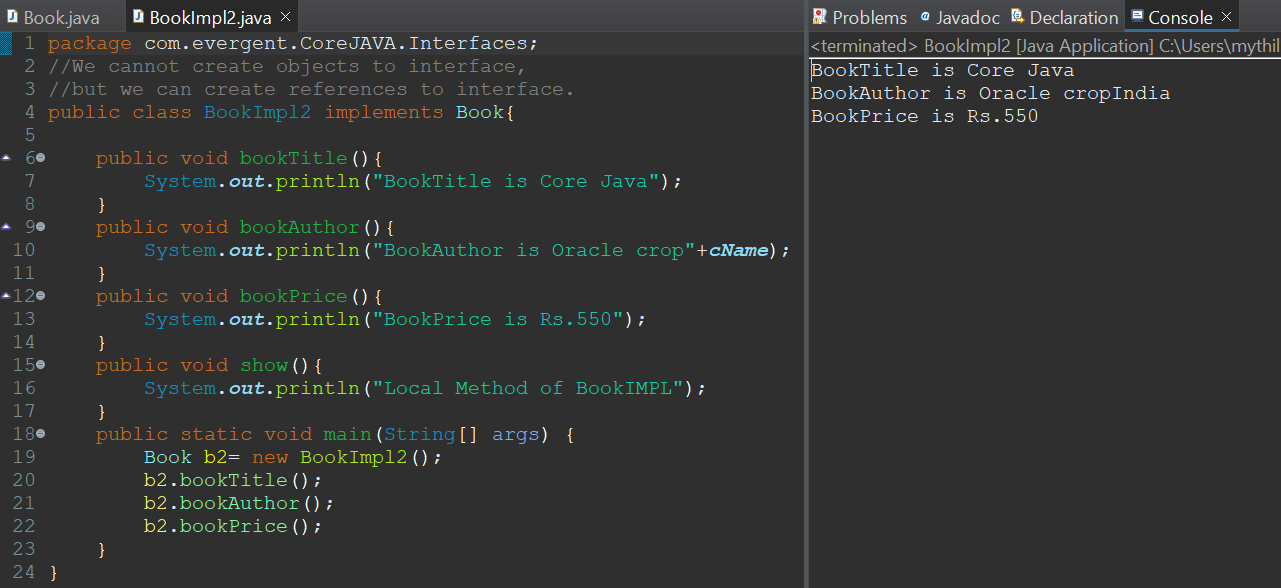
**Program 4:**

****

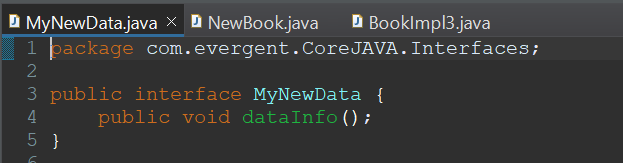
****

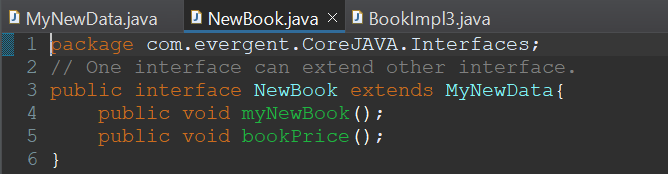
**Program 5:**

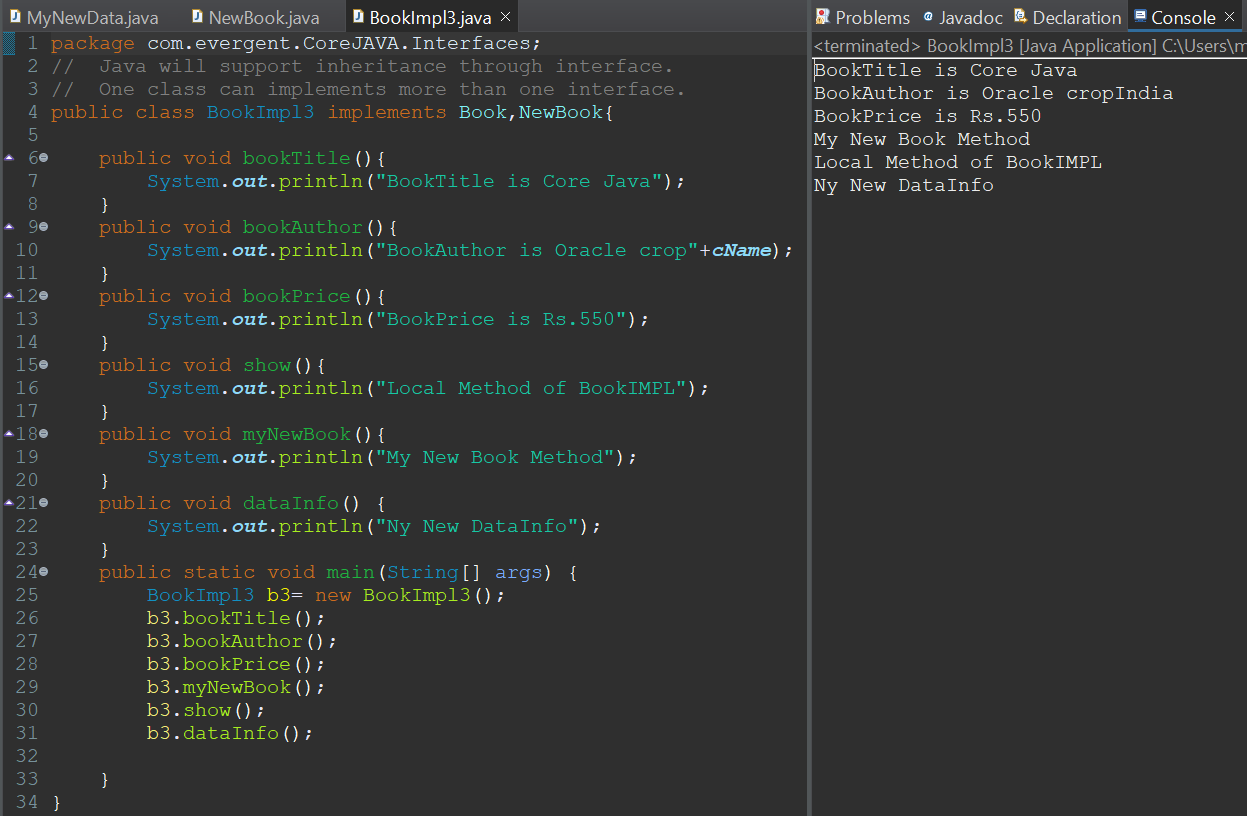
****



**Program 6:**

****

****

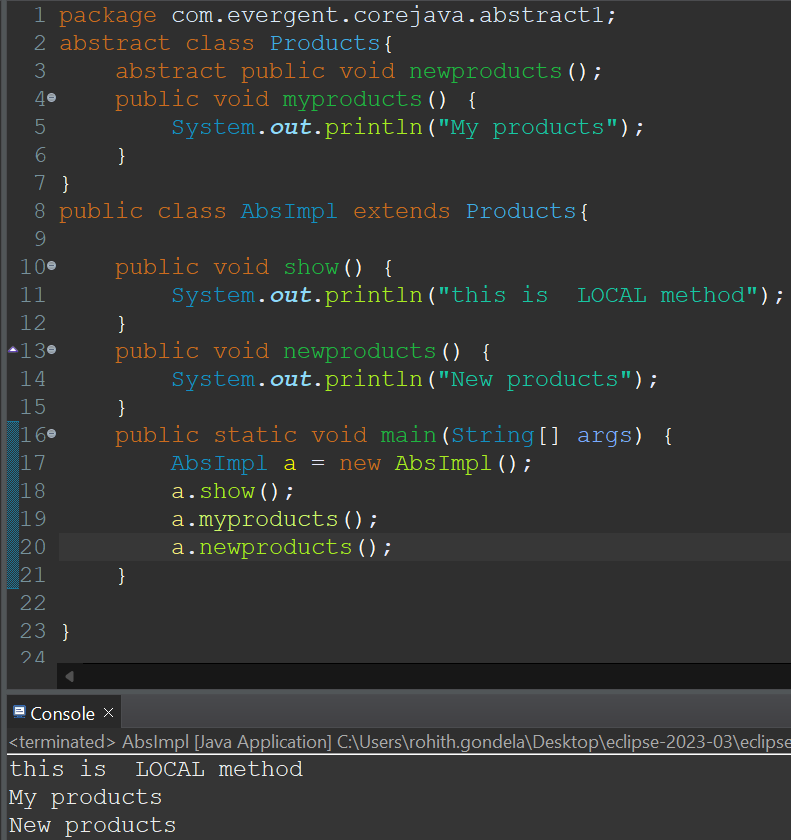
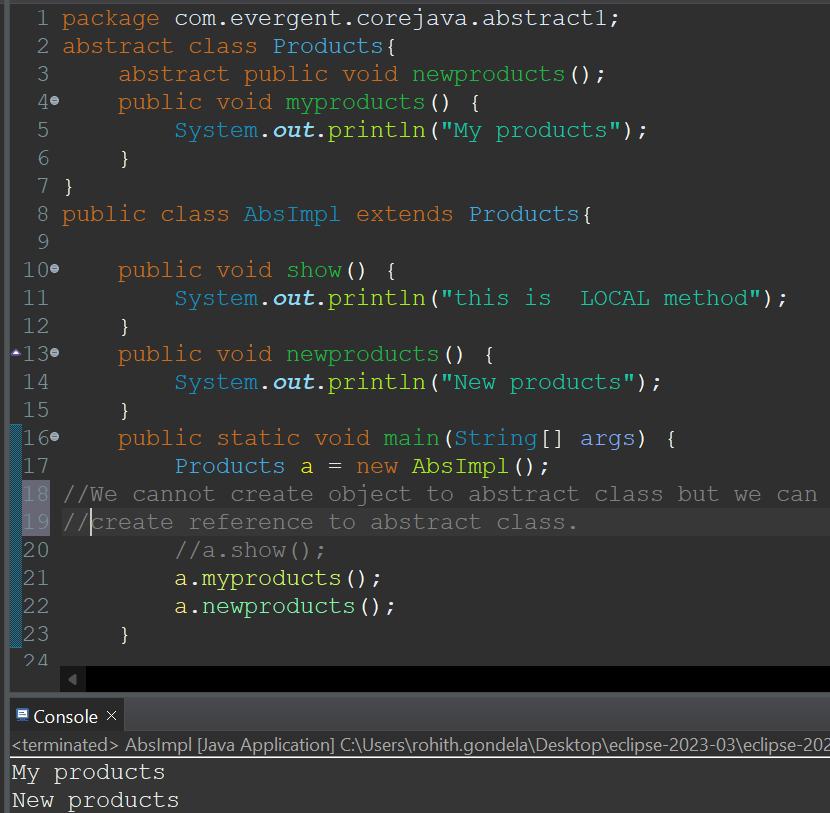


**Day-8 ABSTRACT CLASS 14/8/2024**

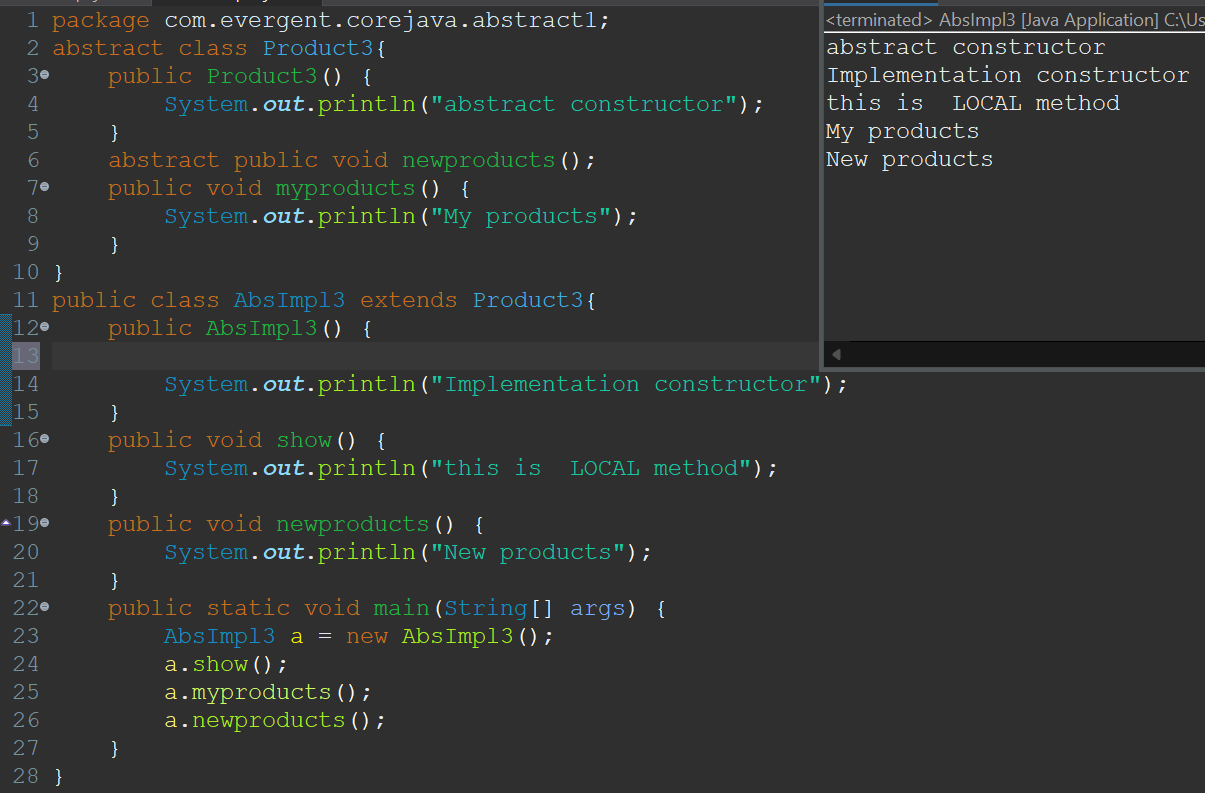
**KEY-POINTS:**

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

**Program-1: Program-2:**

** **

**Program-3: using constructors**

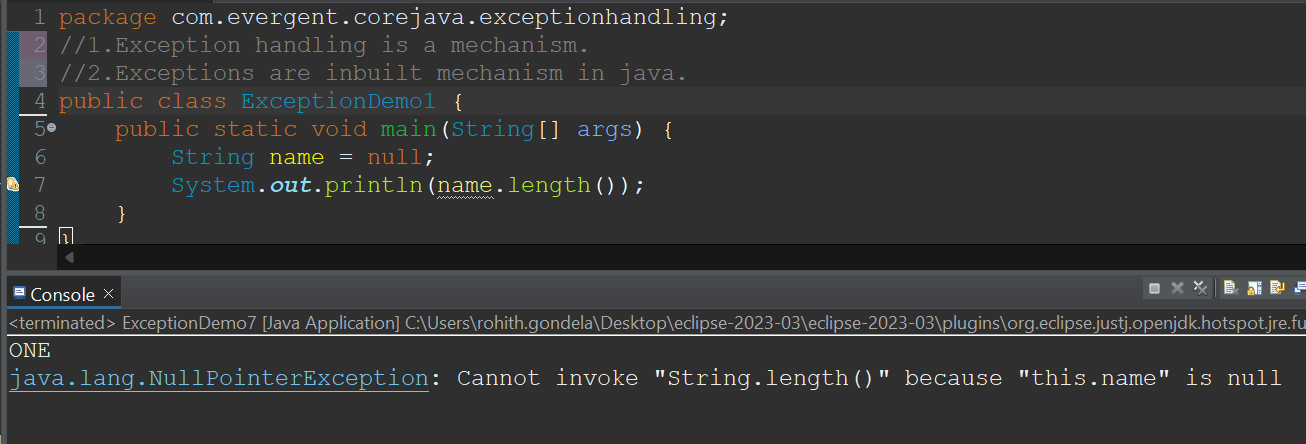
****

**Day-10 EXCEPTION-HANDLING 19/8/2024**

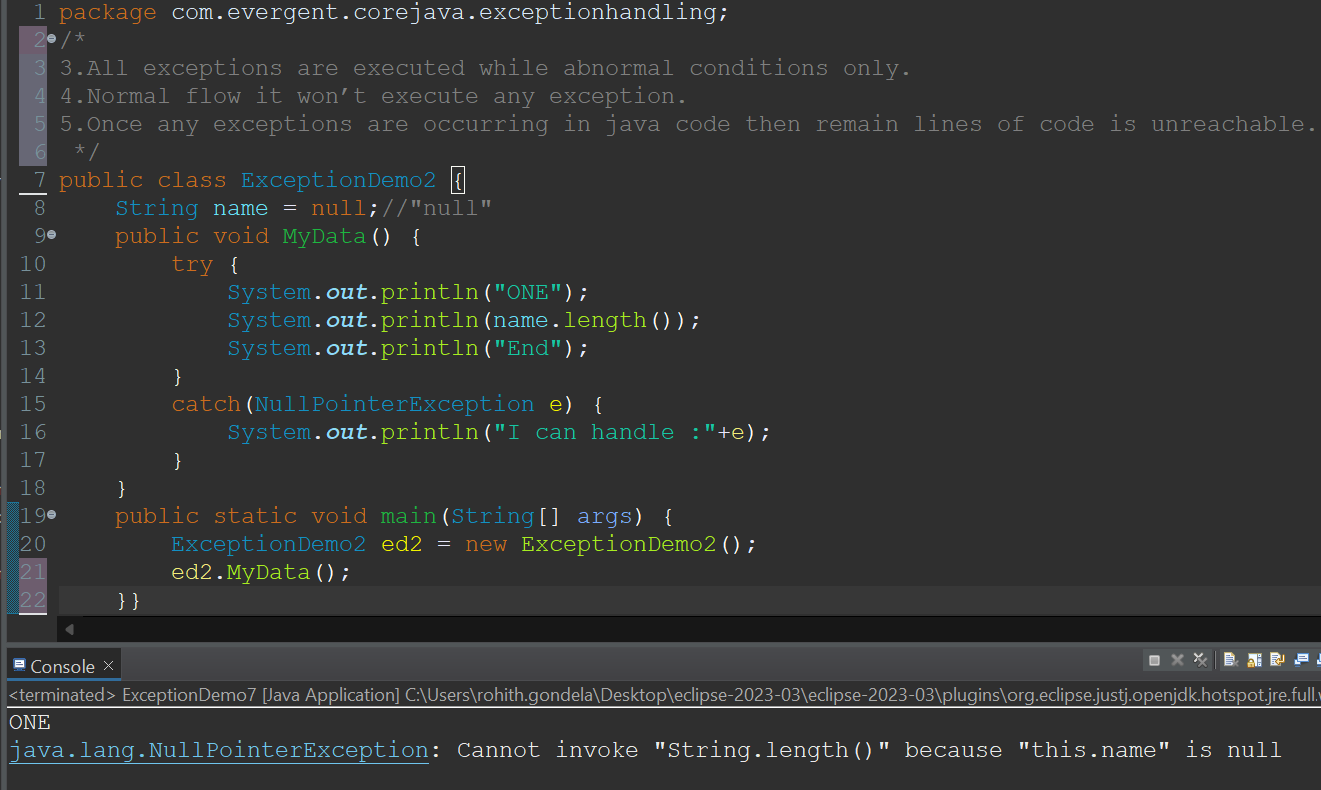
**KEY-POINTS:**

1. Exception handling is a mechanism.
2. Exceptions are inbuilt mechanism in java.
3. All exceptions are executed while abnormal conditions only.
4. Normal flow it won’t execute any exception.
5. Once any exceptions are occurring in java code then remain lines of code is unreachable.
6. Java.lang.Throwable is a super class for Exception and Error.
7. There are 2 types of Exceptions:
   1. Checked Exception
   2. Unchecked Exception
8. All checked Exceptions are compile-time Exceptions.
9. All unchecked Exceptions are run-time Exceptions.
10. There are 5keywords in Exception handling
    1. Try
    2. Catch
    3. Finally
    4. Throws
    5. Throw
11. Try for business logic.
12. Catch 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.
16. Try followed by either catch block or finally block.
17. We should follow exception hierarchy.
18. We can create our own exceptions.
19. Our own exceptions extends exceptions or runtime exceptions.
20. All exception classes are into java.lang package.
21. If there is 2 exceptions in class, Developer should be handle 1st exception then only 2nd exception will be handled.
22. Errors are not in developer control.
23. Exceptions protect business logic.
24. For one try block we can write multiple catch blocks.

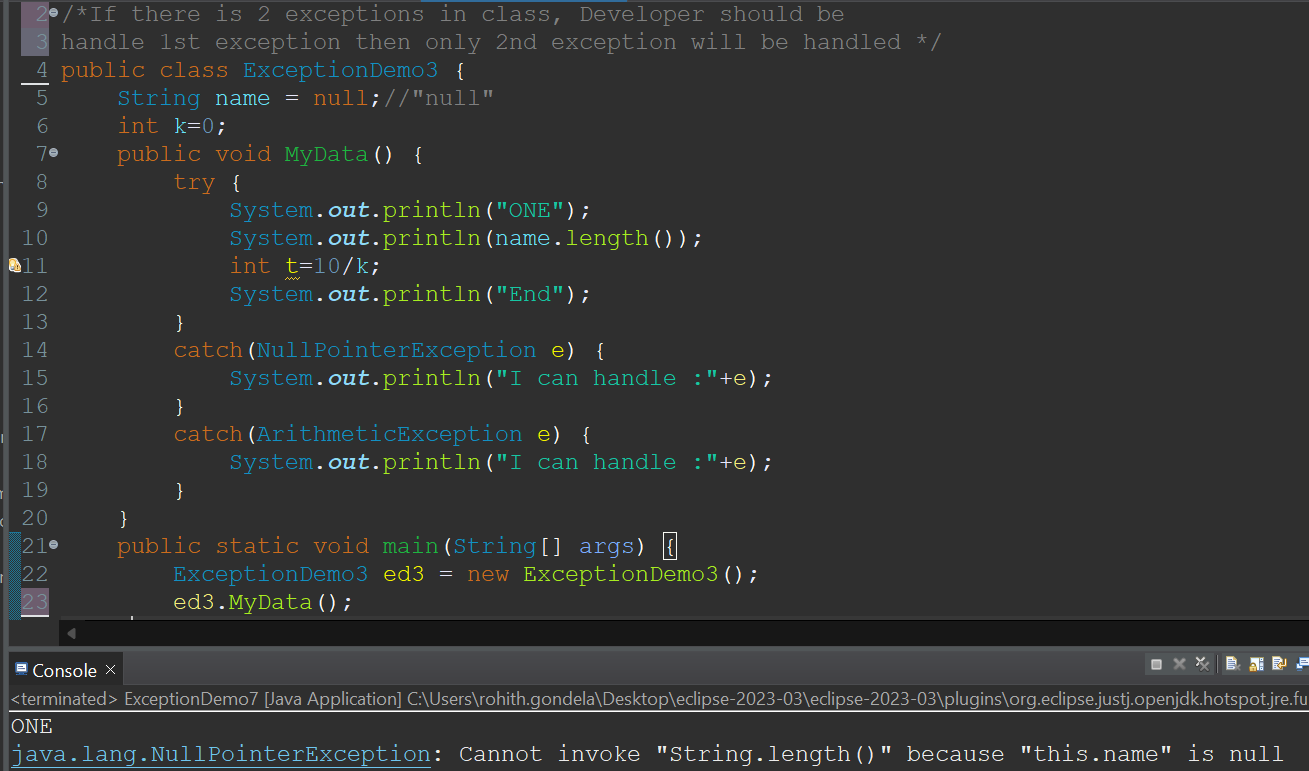
**Program-1:**

****

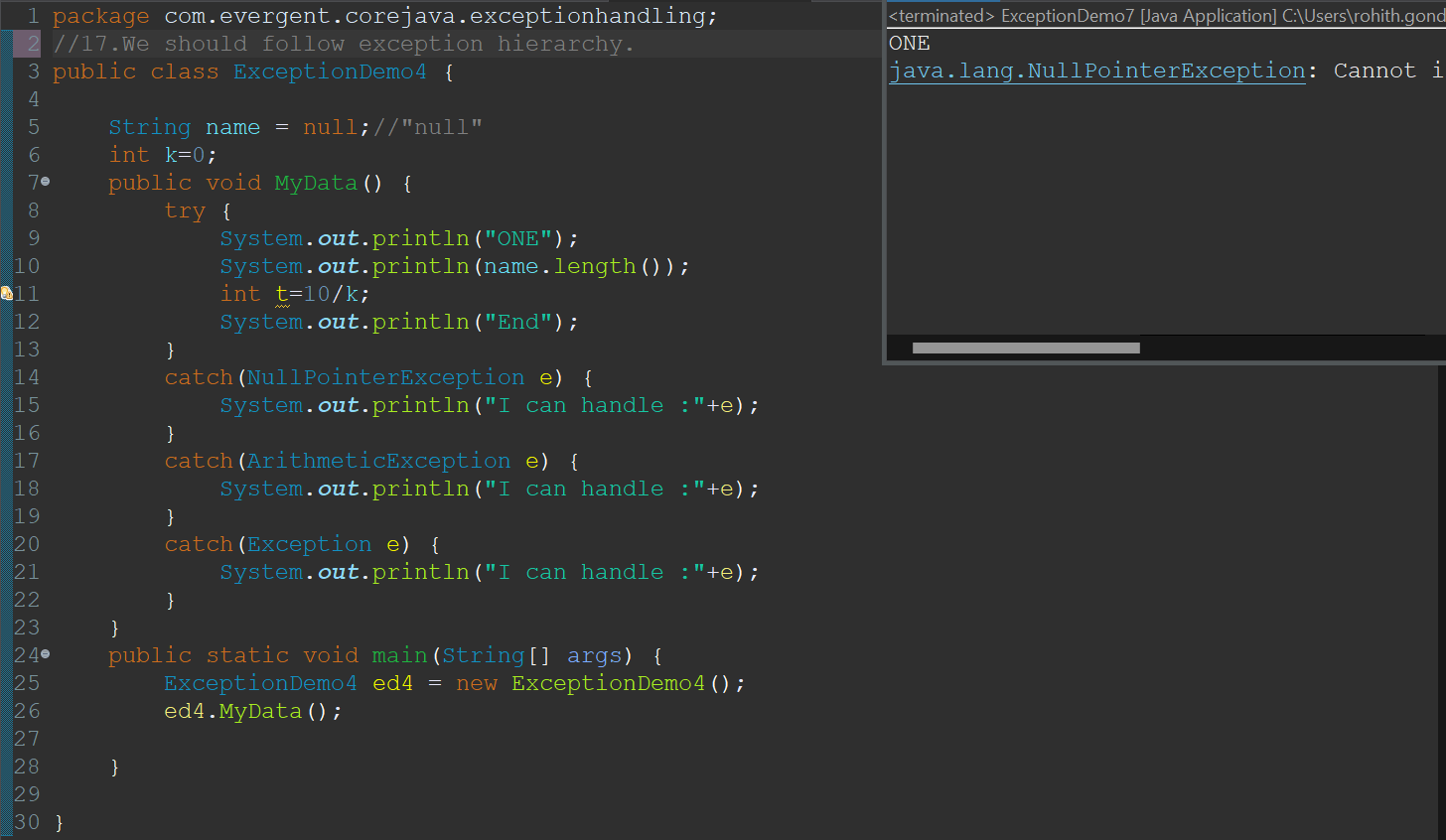
**Program-2:**

****

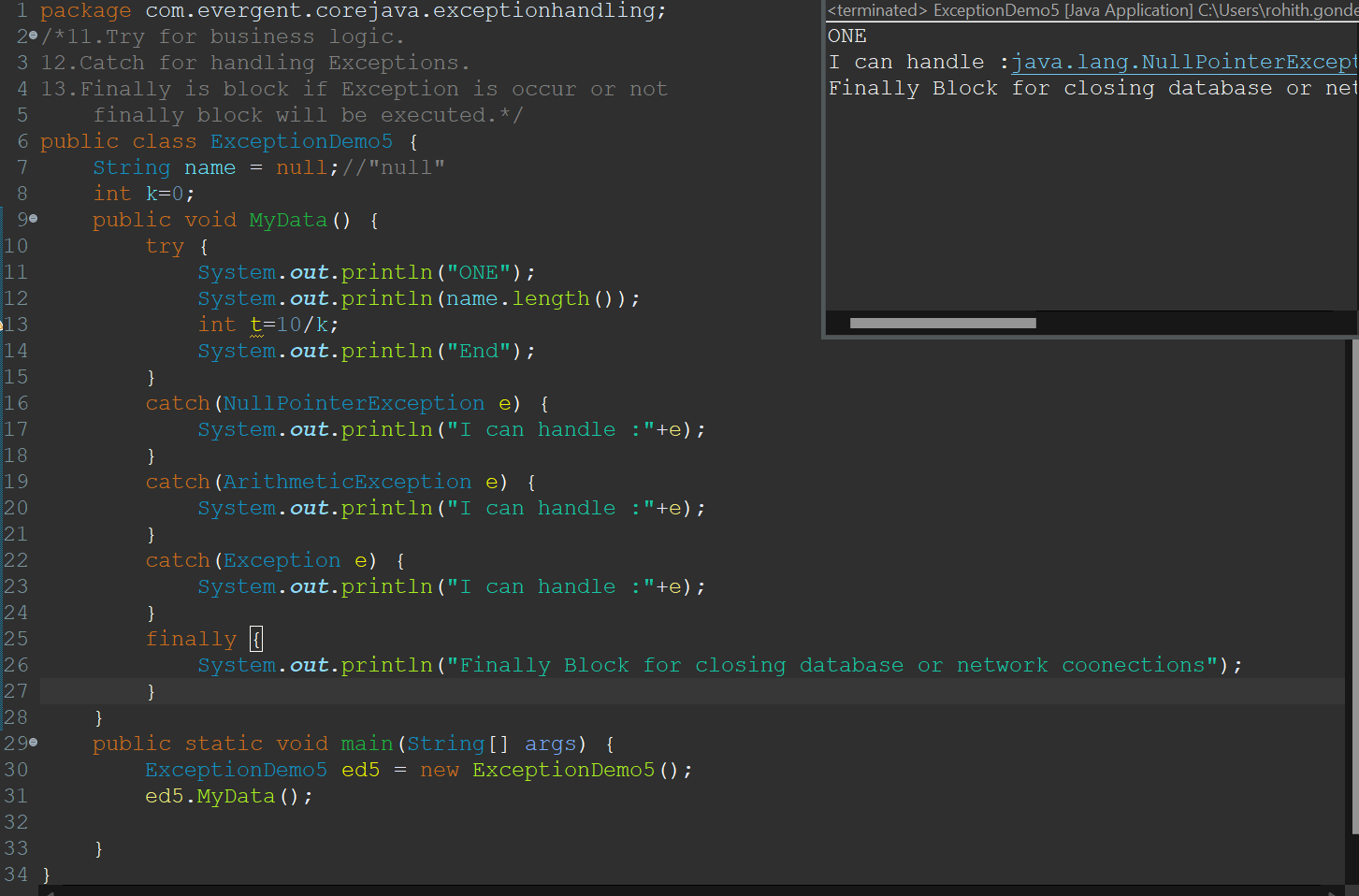
**Program-3:**

****

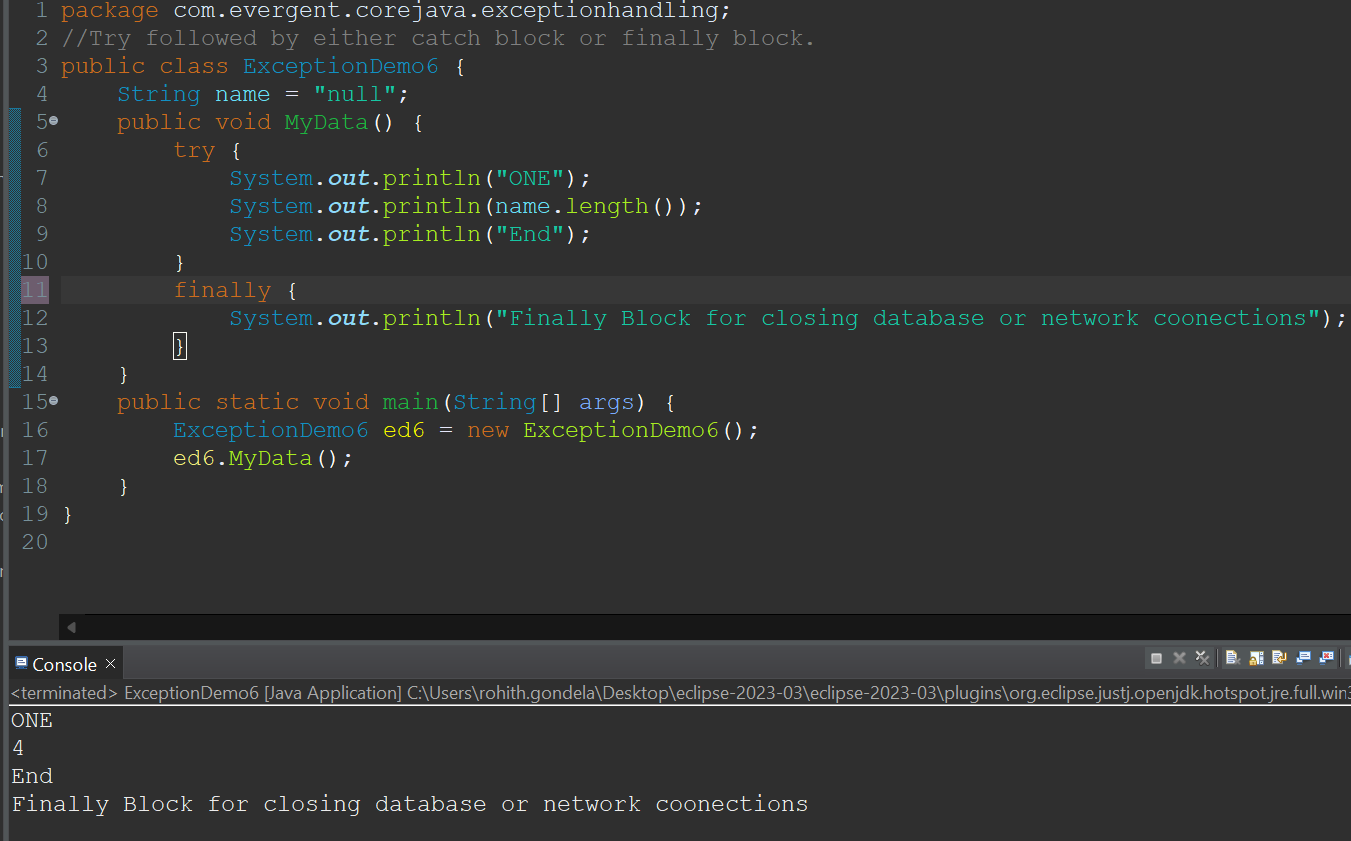
**Program-4:**

****

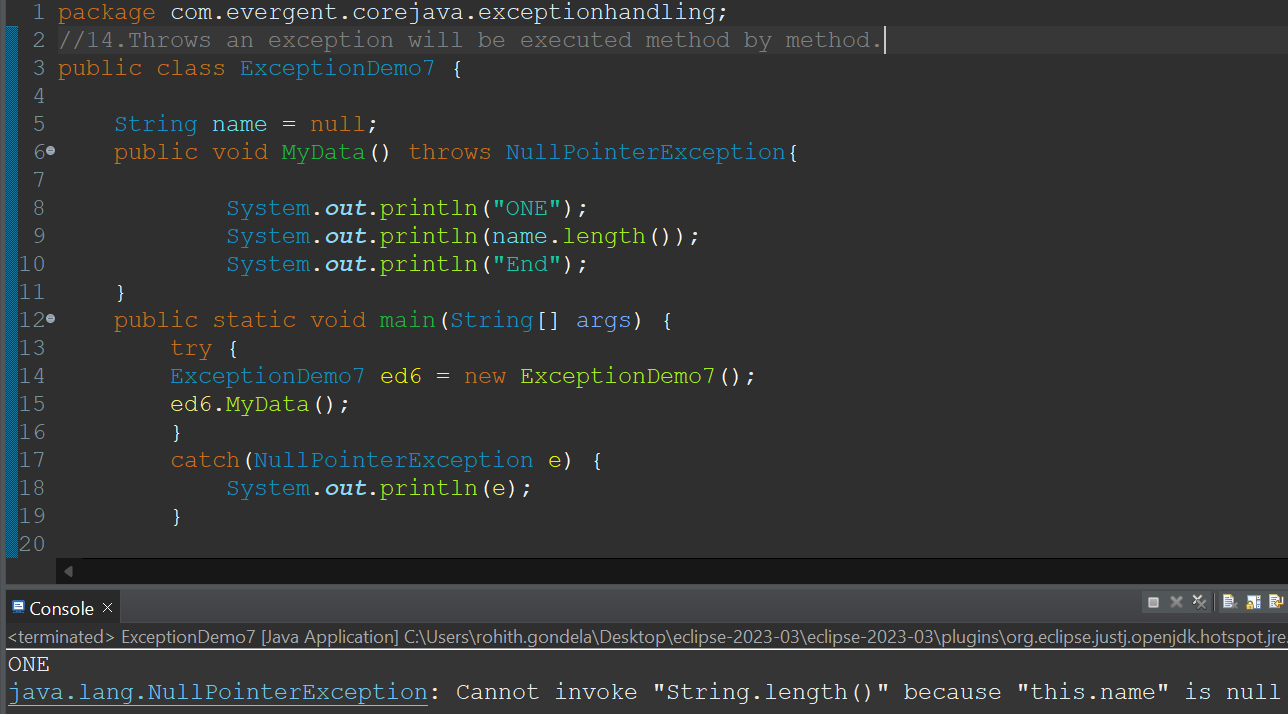
**Program-5:**

****

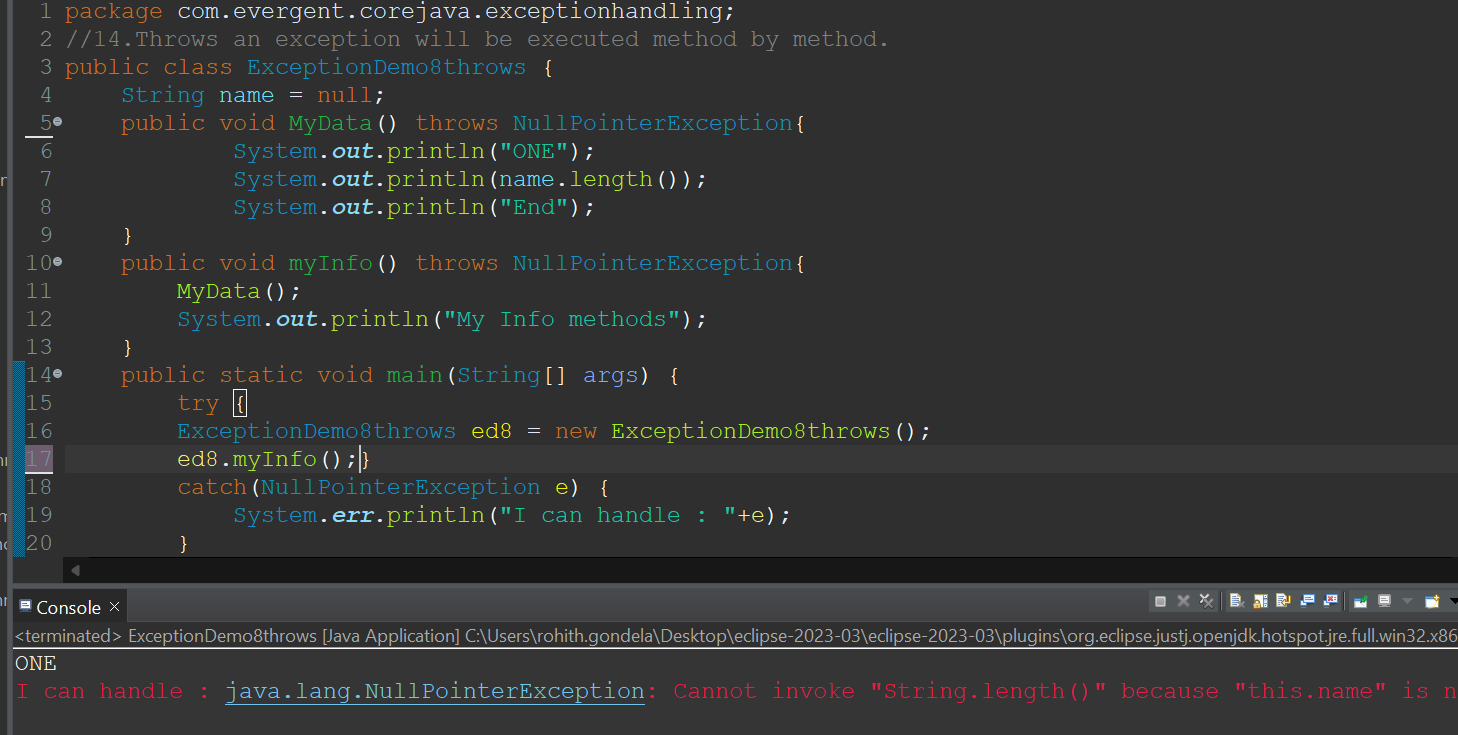
**Program-6:**

****

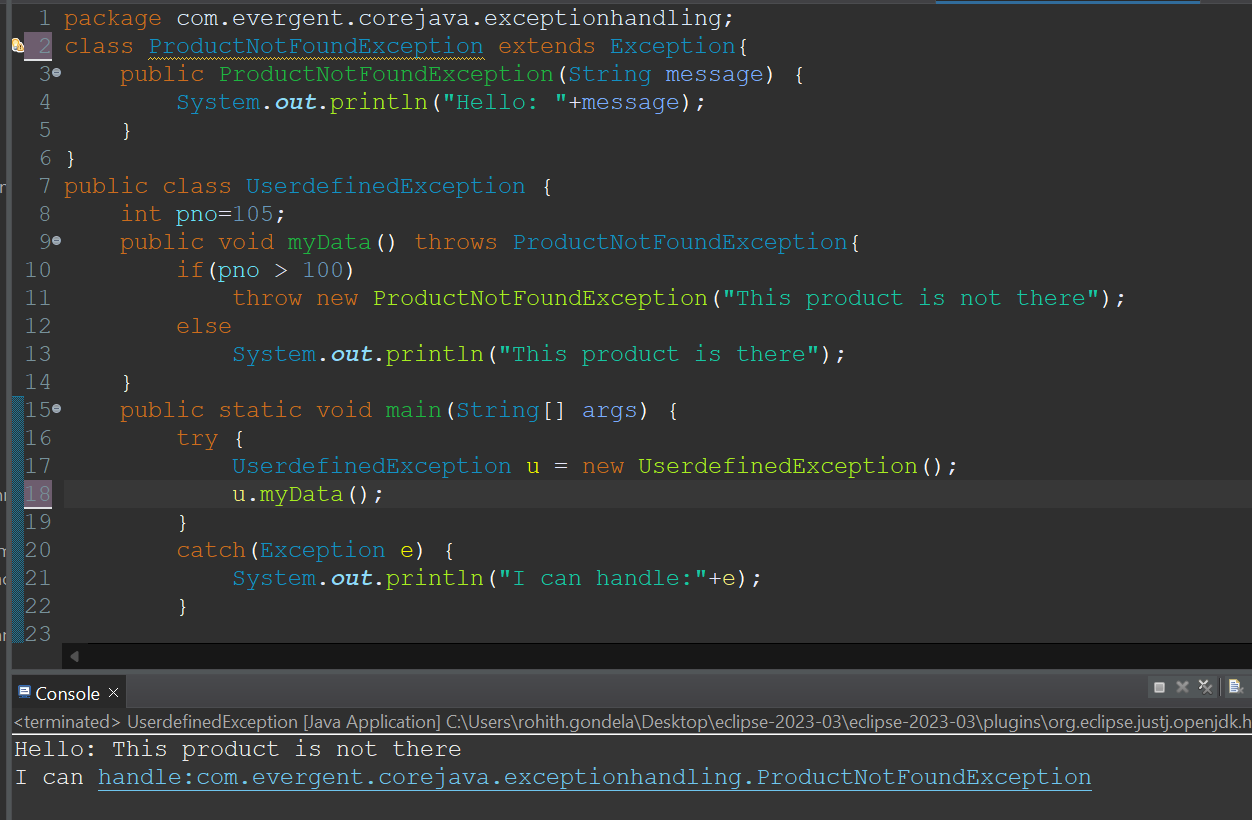
**Program-7:**

****

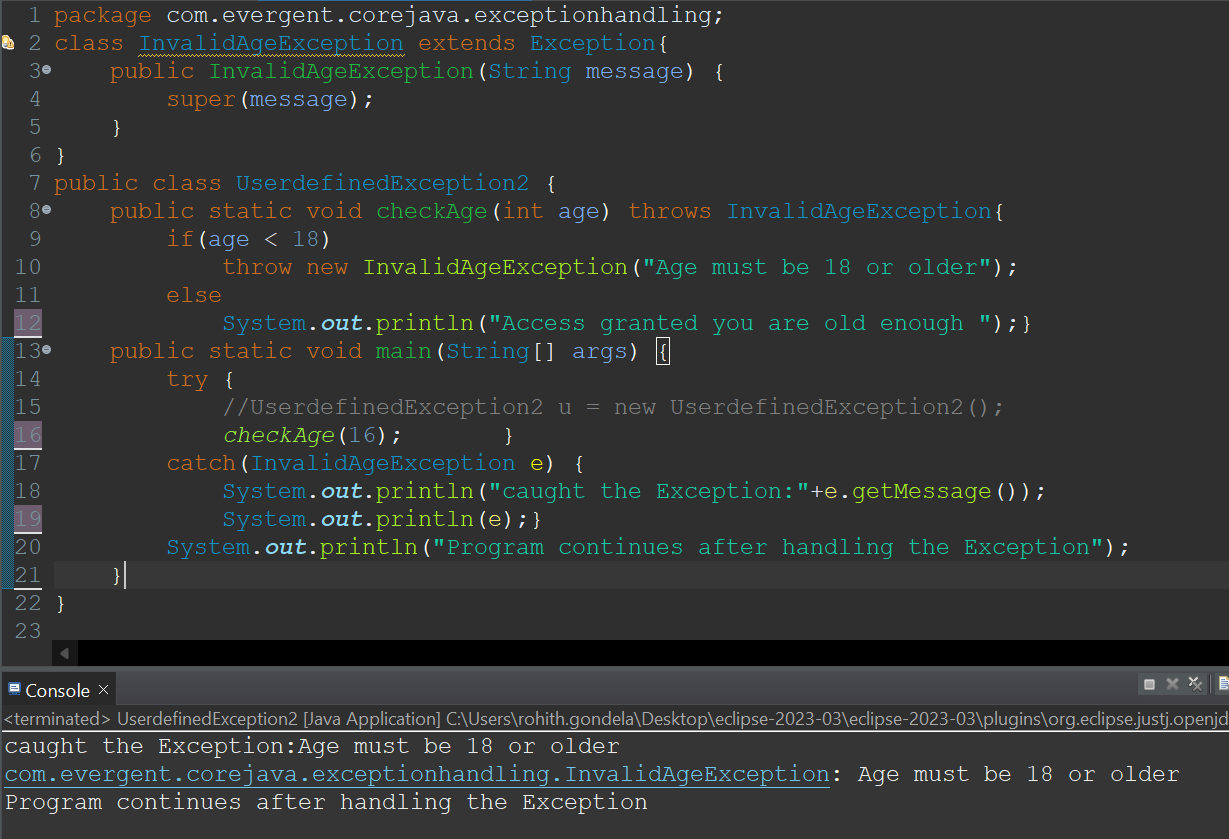
**Program-8:Using throws keyword**

****

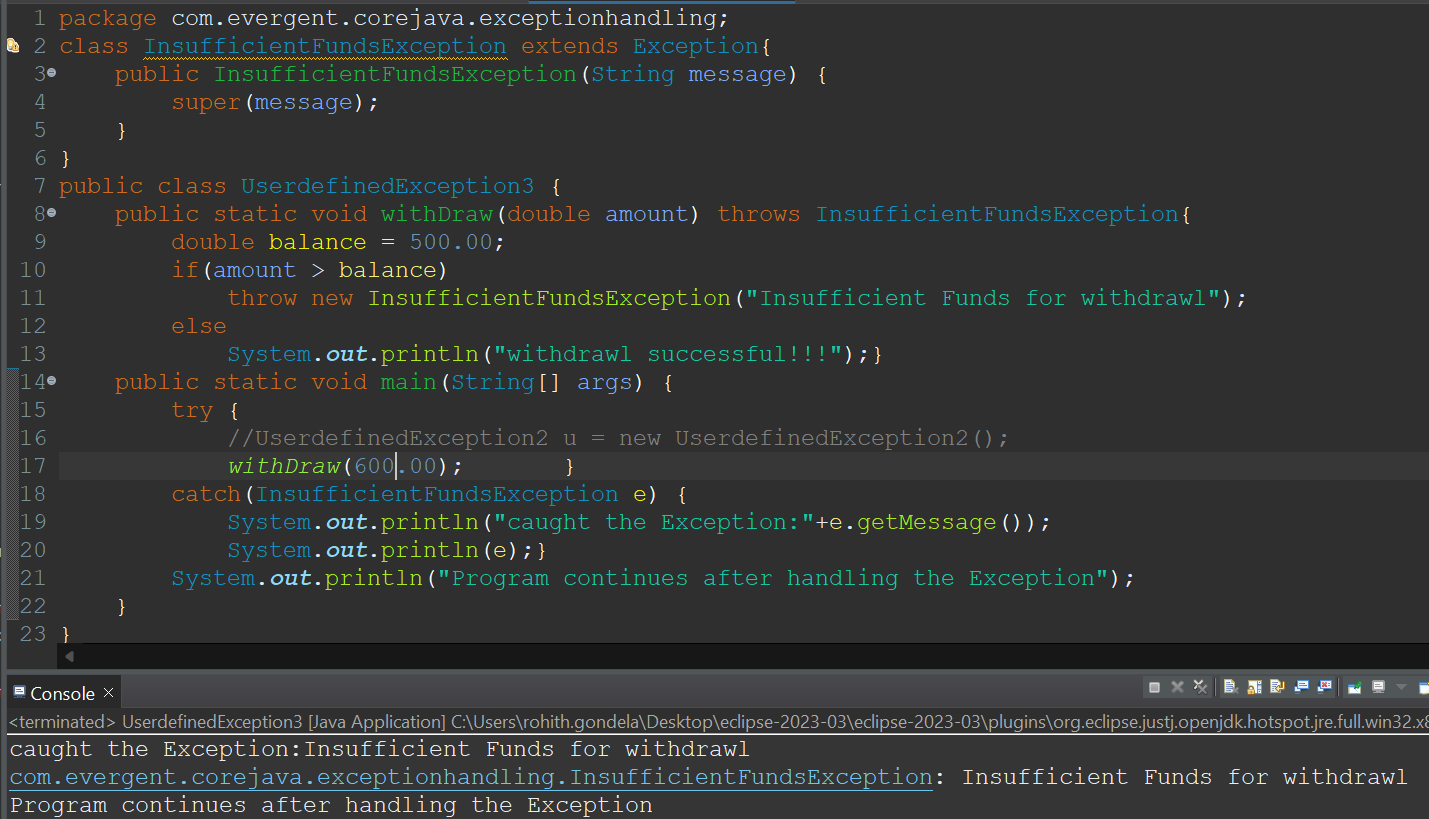
**Program-9:User defined Exception**

****

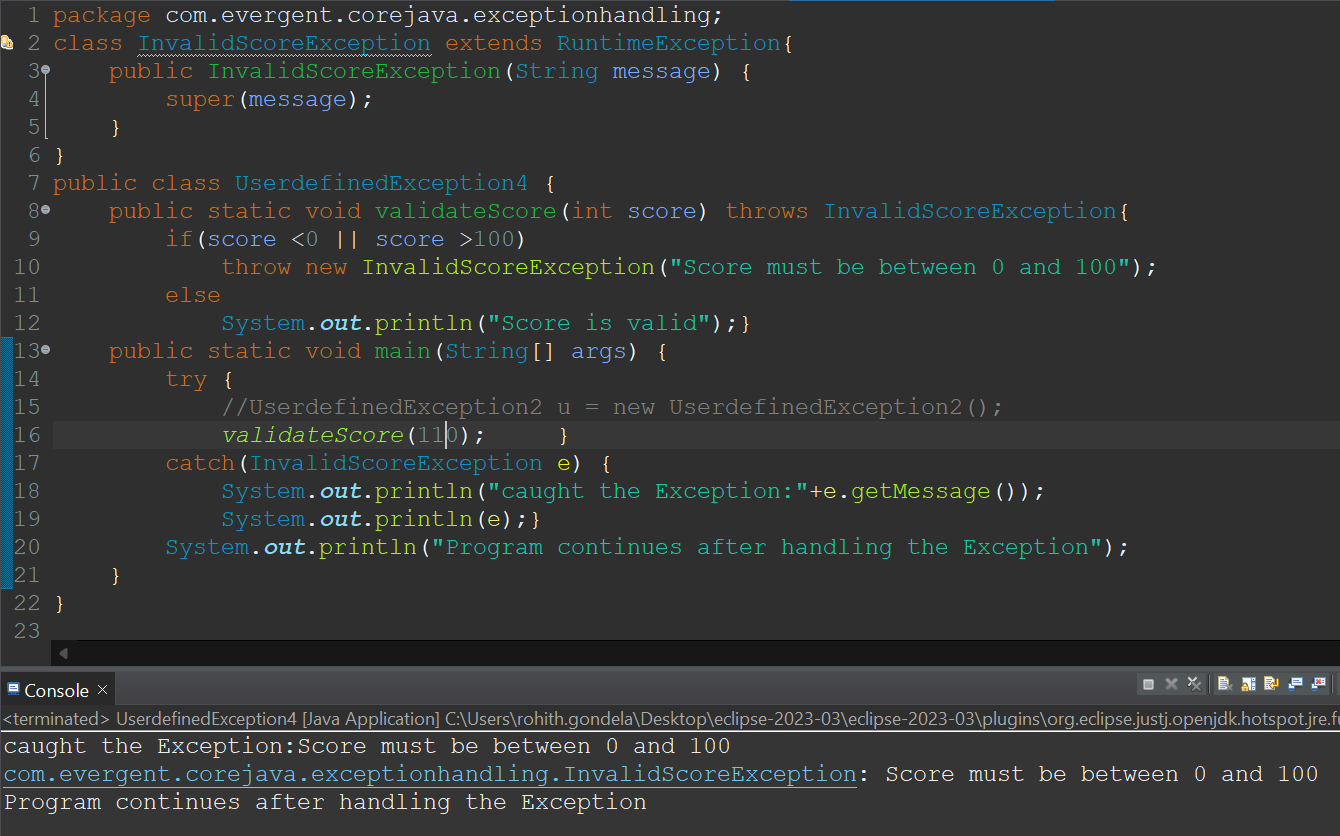
**Program-10:Invalid Age Exception**

****

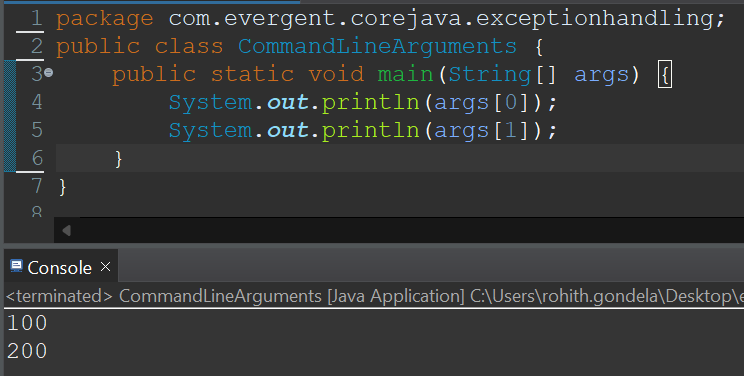
**Program11:Insufficient funds Exception.**

****

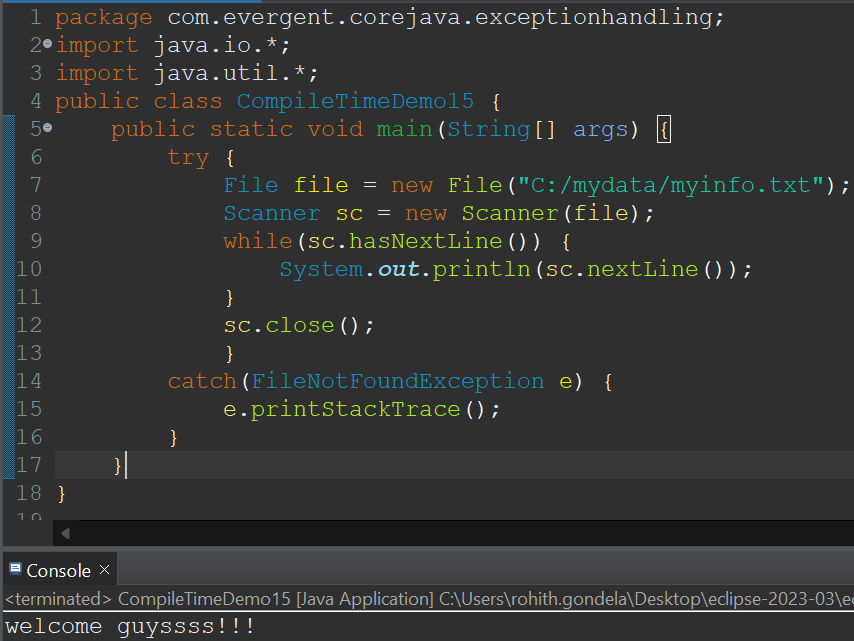
**Program-12:InvalidScore Exception:**

****

**Program-13: Command Line Arguments**

****

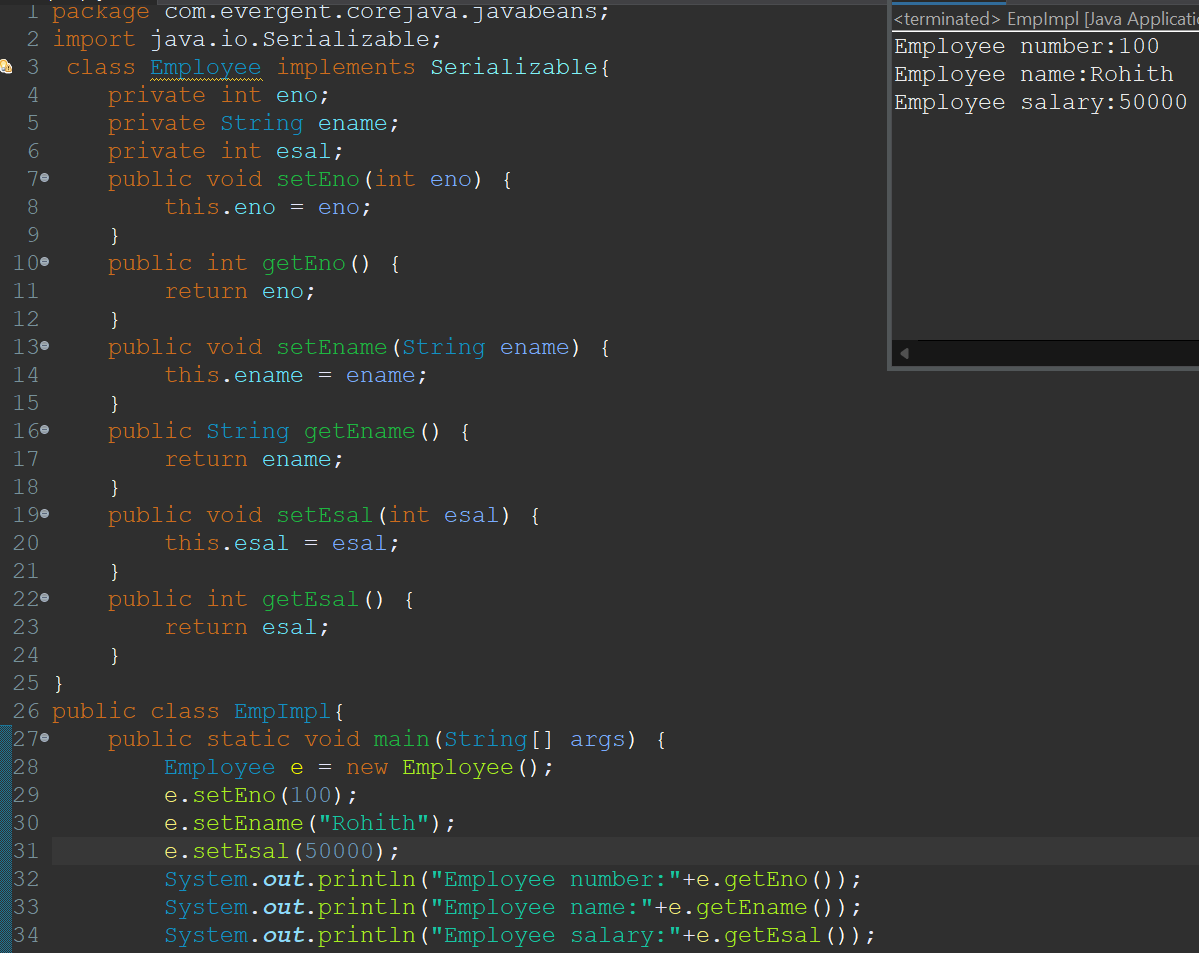
**Program-14:File Not Found Exception**

****

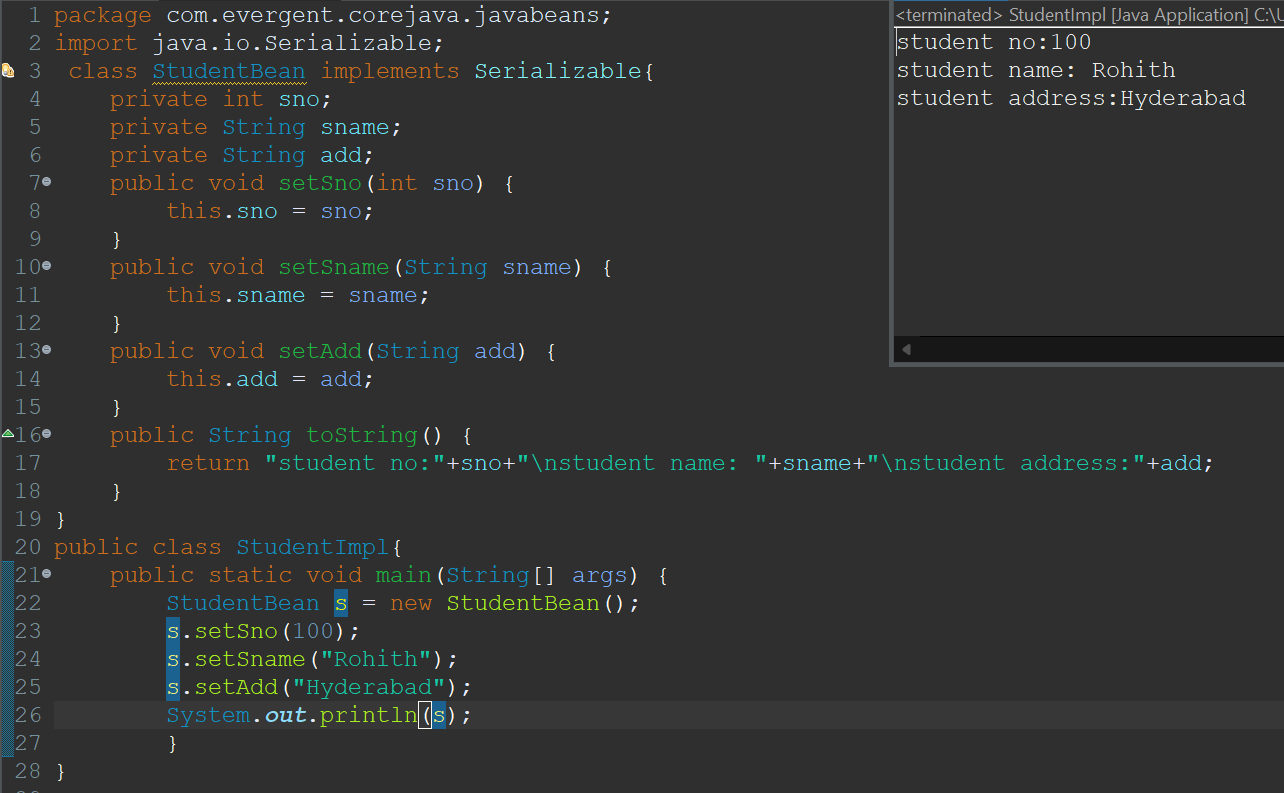
**Day-12 JAVA BEANS 21/8/2024**

* **KEYPOINTS:**
* Java bean is a mechanism.
* Java bean is light weight.
* All attributes are private.
* get/set methods are public.
* Implements java.io.Serializable.
* We can achieve tightly encapsulation through java beans.

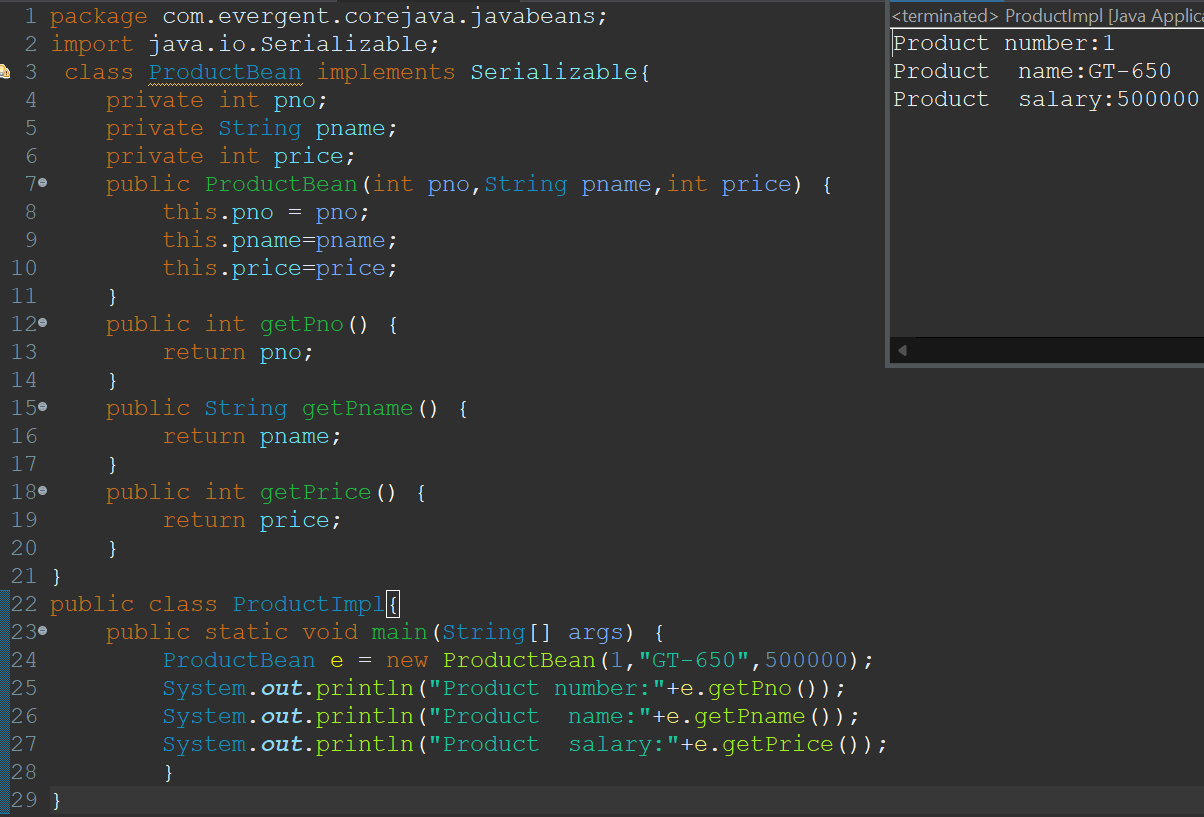
**Java beans -Program-1:using getters ans setters.**

****

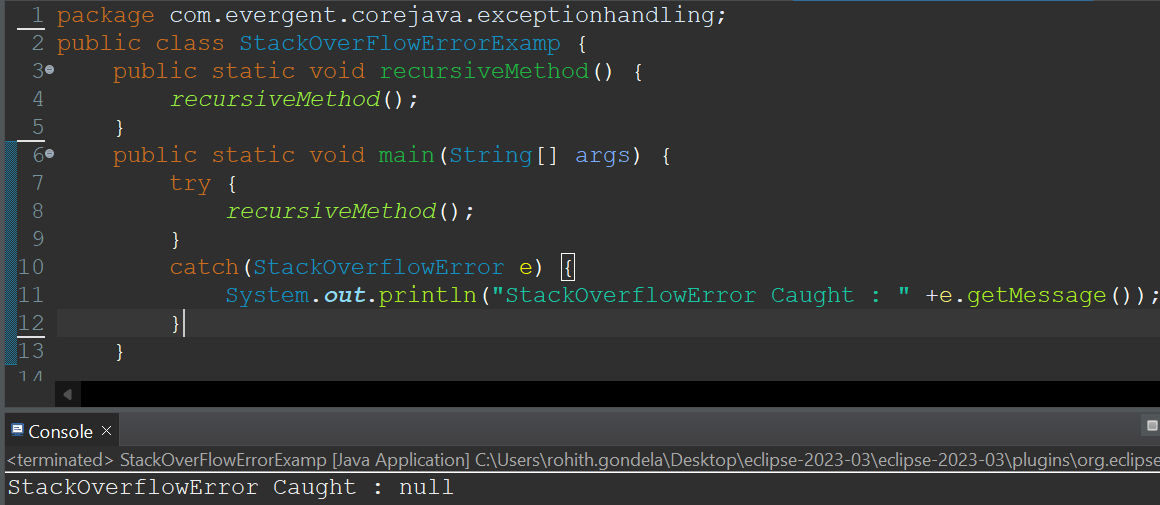
**Java beans -Program-2:using toString().**

****

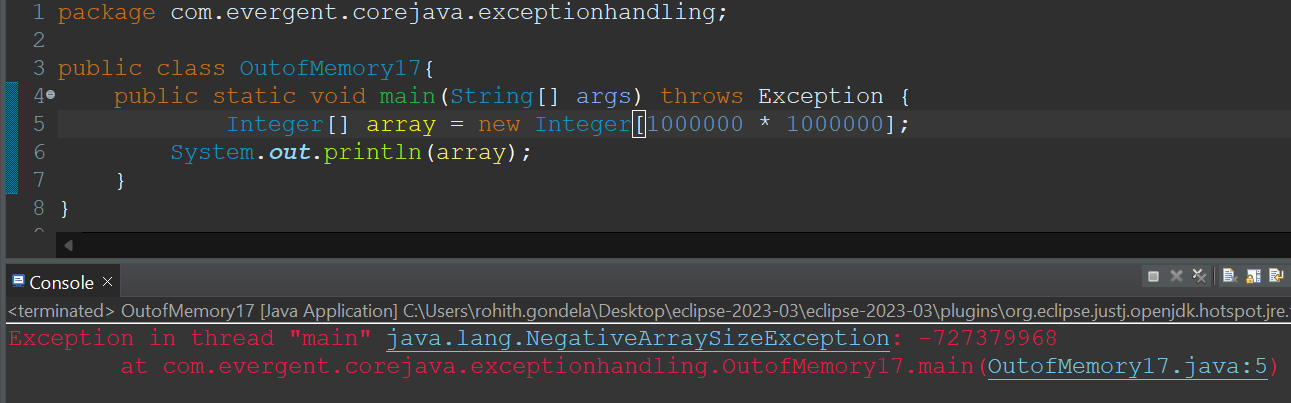
**Java beans -Program-3:using constructor initialization.**

****

**Stack over flow Error:**

****

**Out of memory Error:**

****

**Day-13 COLLECTION FRAMEWORK 22/8/2024**

Vector(Legacy)

LinkedList

ArrayList

TreeSet

HashSet

<List>

<Set>

**<Collection>**

**KEY POINTS:**

1. Java.util.Collection is the root interface.
2. Advantages:
   1. Size growing
   2. Different data types are storing
   3. Multiple methods

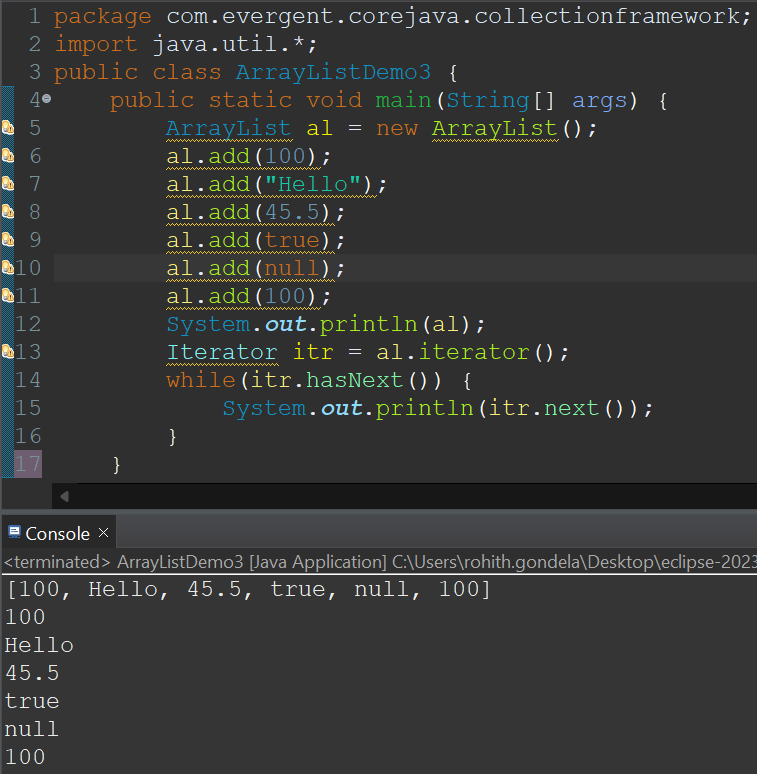
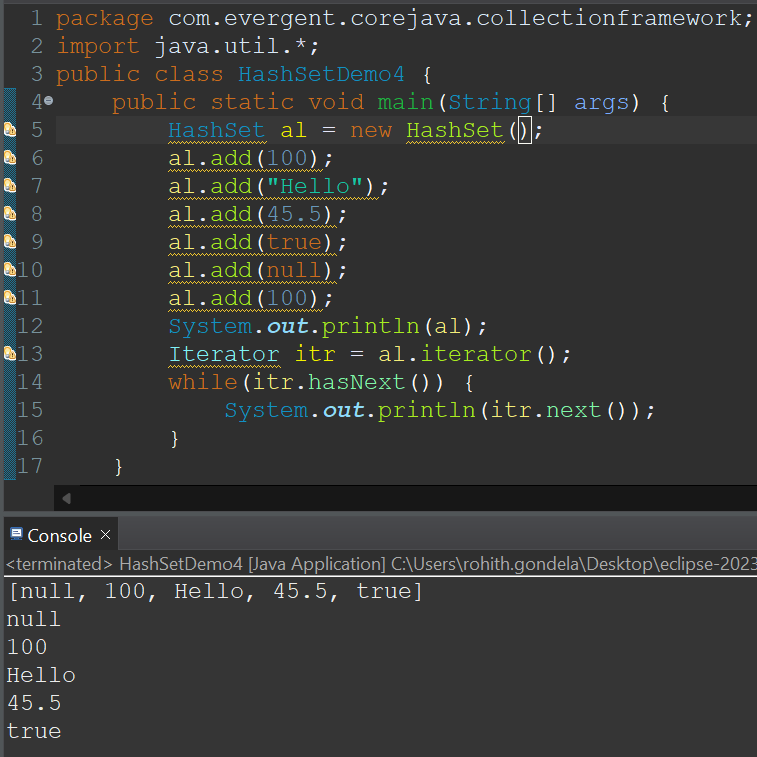
**Program-1: ArrayList**

****

**Program-2: HashSet**

****

**Retriving the elements from ArrayList Retriving the elements from HashSet**

** **

**Program-4:TreeSet**

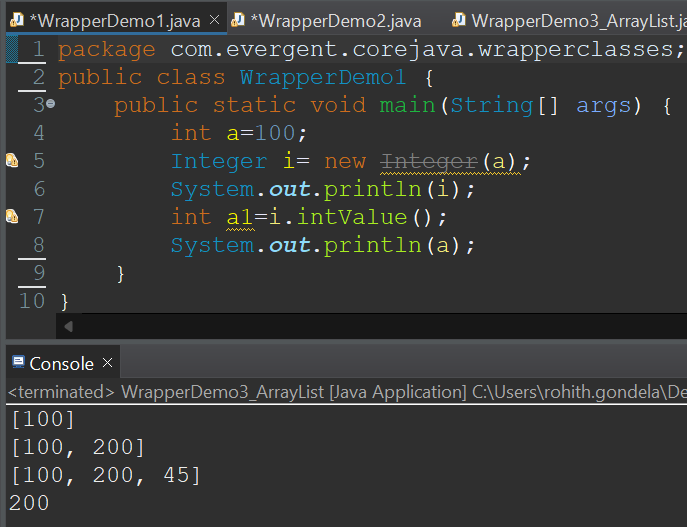
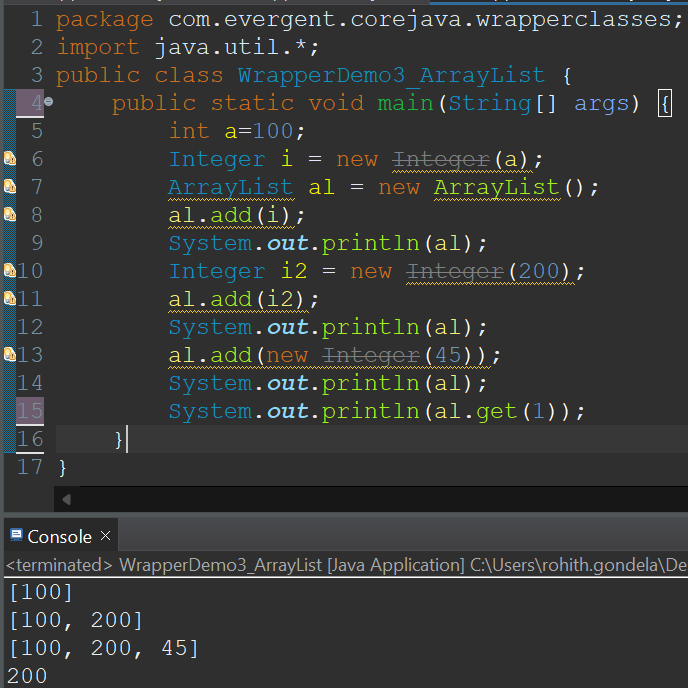
****

**Day-14 WRAPPER CLASSES 23/8/2024**

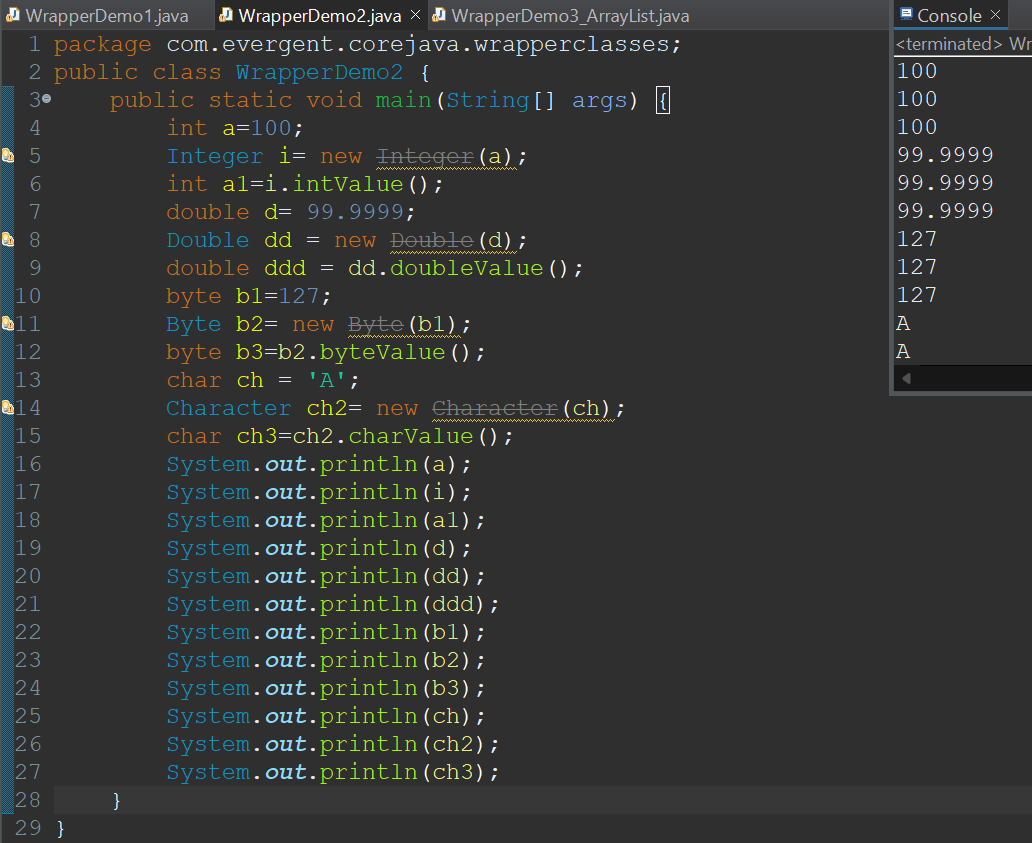
**KEY POINTS:**

1. A Wrapper class in Java is a class whose object wraps or contains primitive data types.
2. They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
3. The classes in java.util package handles only objects and hence wrapper classes help in this case also.
4. Data structures in the Collection framework, such as [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/) and [Vector](https://www.geeksforgeeks.org/vector-vs-arraylist-java/), store only objects (reference types) and not primitive types.
5. An object is needed to support synchronization in multi-threading.

**PROGRAM-1: PROGRAM-2:**

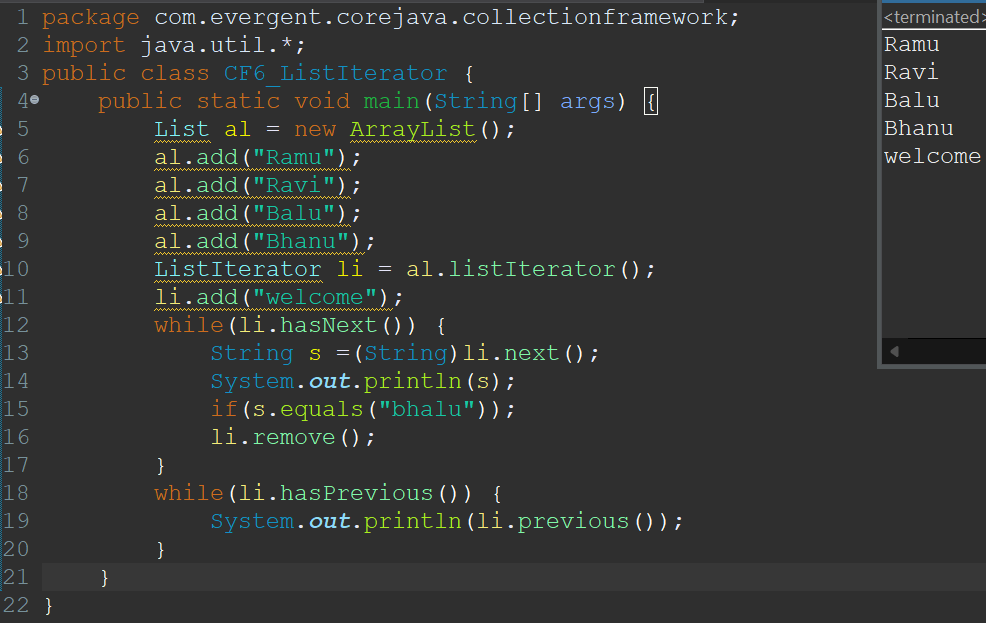
 

**PROGRAM-3:**



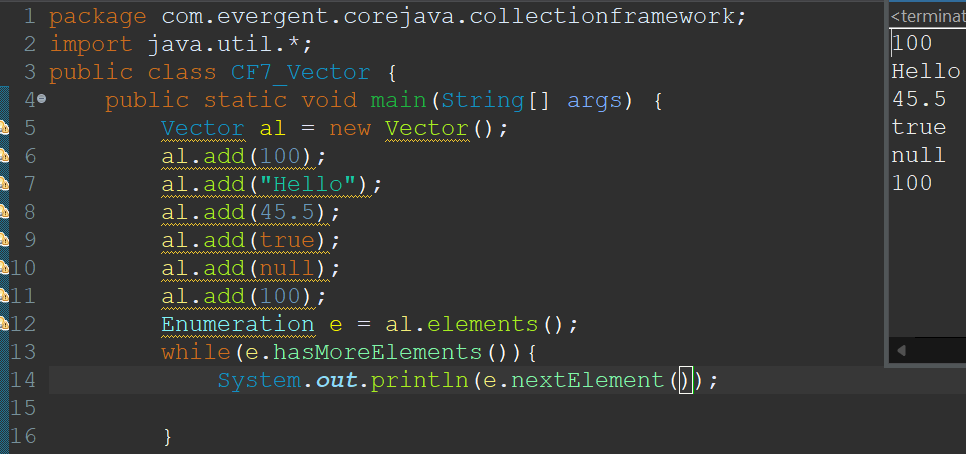
**LIST ITERATOR:**

* It is cursor in Java.
* It can do both transversal(Forward and Backward)
* It has more methods than iterator
* It is better to use compared to iterator and enumeration



**VECTOR:**

* It is a implementation class for List Interface.
* It is legacy API (jdk-1.0) but not recommended to use.
* Vector is not used because one it is legacy and another reason the size of the vector increase doubly not dynamically like Arraylist.
* The main advantage of vector is Synchronization.Hence it is used in databases and banking security.
* But now instead of vector we have similar alternate way is Synchronized Arraylist.

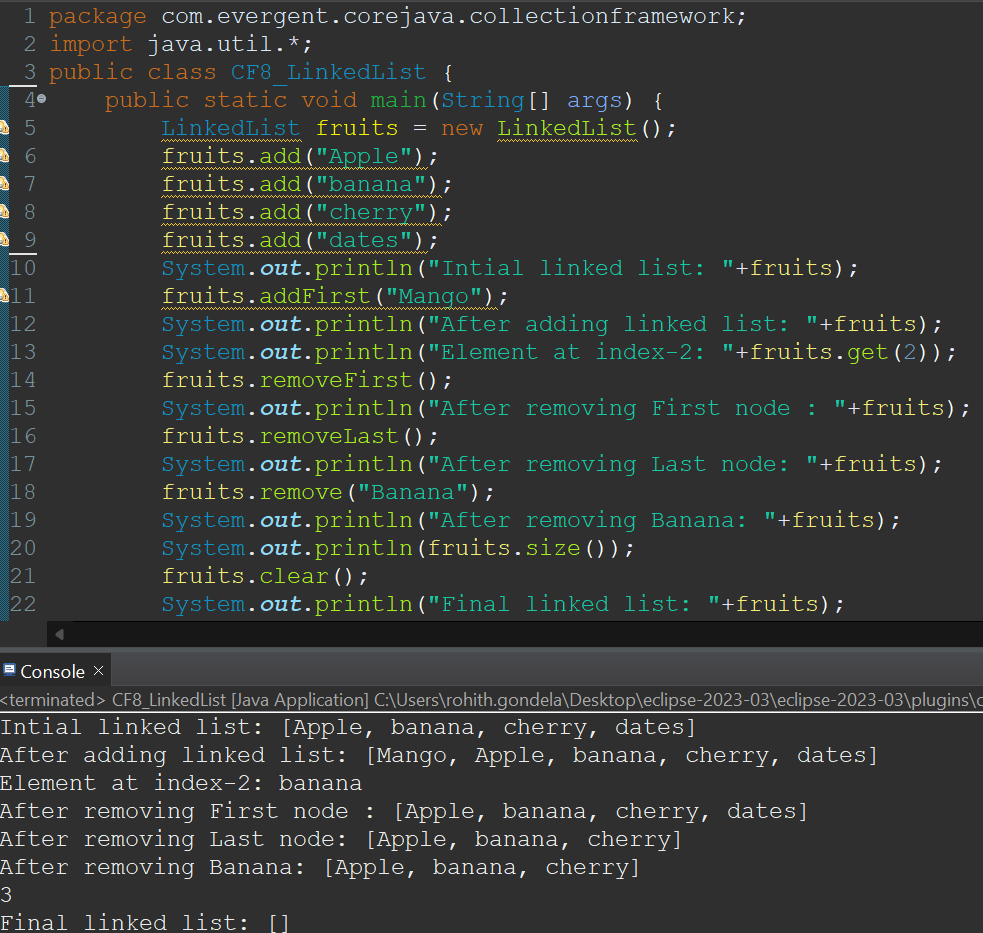


**LINKED LIST:**

A linked list contains a group of elements in a form of nodes.Each node will have three fields,the data field contains data and the link fields contains addresses to previous and next nodes.

LINK DATA LINK LINK DATA LINK LINK DATA LINK

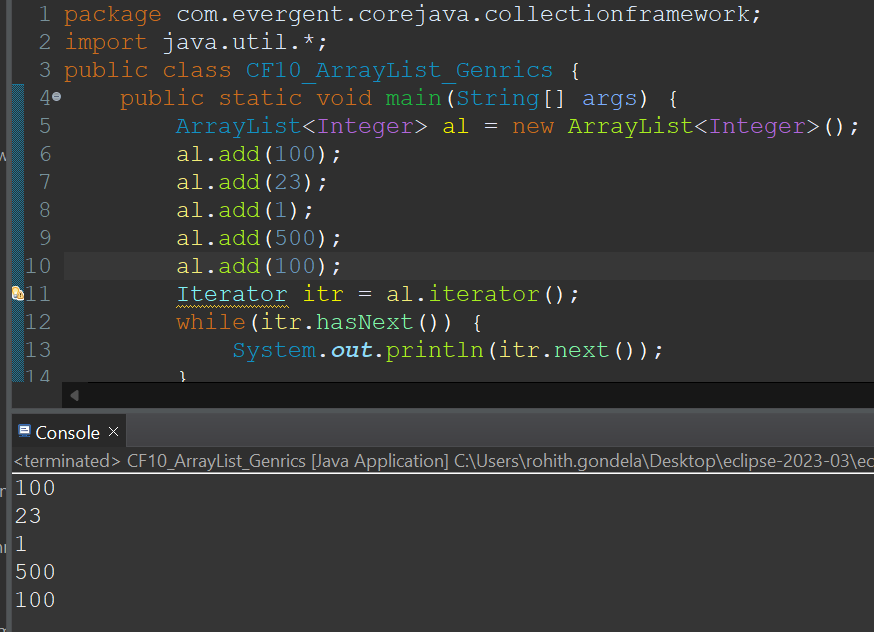
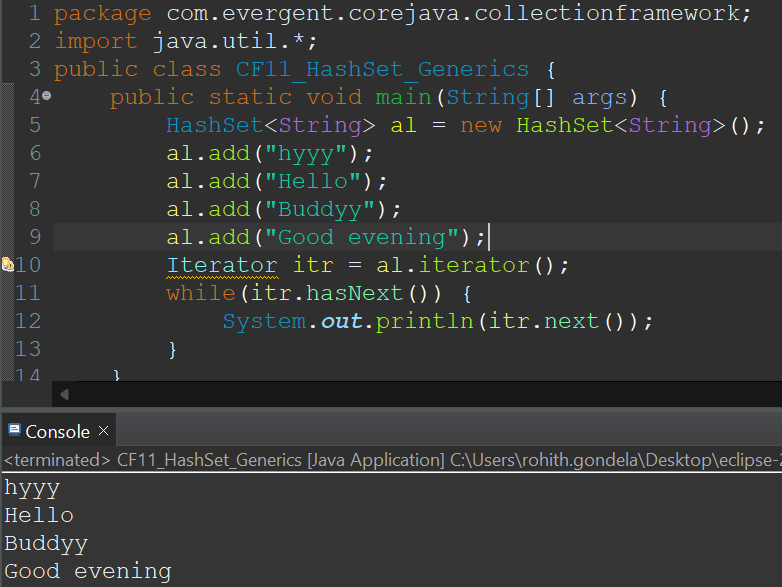
NODE-1 NODE-2 NODE-3



**GENERICS:**

* It was introduced in jdk-1.5.
* If we want to store only one type of data then we can choose Generics.
* Generics only takes wrapper classes.
* It allows only similar data.
* It can be denoted as <>.

Program-1:Using Integer class Program-1:Using String class

**Another CF-Program with different type**

