**JAVA Training Index**

**Day 1: 05-Aug-2024:**

1. Languages & Packages
2. Java Features
   1. Why Java is platform independent
3. JDK, JRE, JVM
4. Basic Java
   1. Datatypes
   2. Operators
   3. Conditions(if, if-else)
   4. Loops(Nested for loops)
5. Packages

CoreJAVA-Workspace :

CoreJAVA-Development

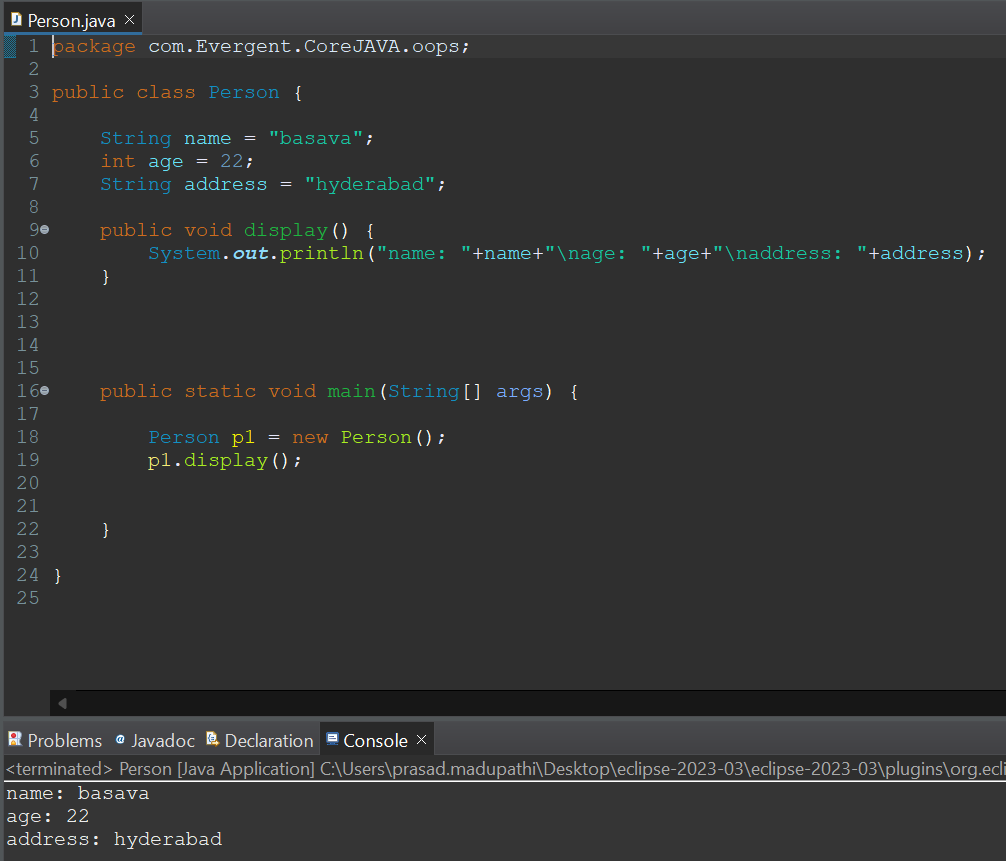
CoreJAVA-Application

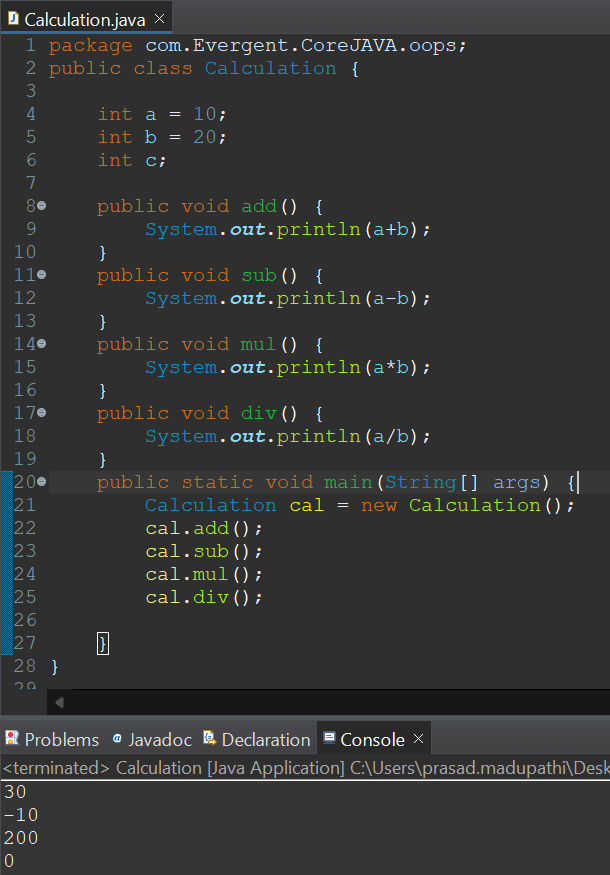
**Day 2: 06-Aug-2024:**

1. Logical Programming
2. Patterns
3. Switch Case
4. Arrays
   1. 1D Array's
   2. 2D Array's
5. Enum's
6. Event Management System

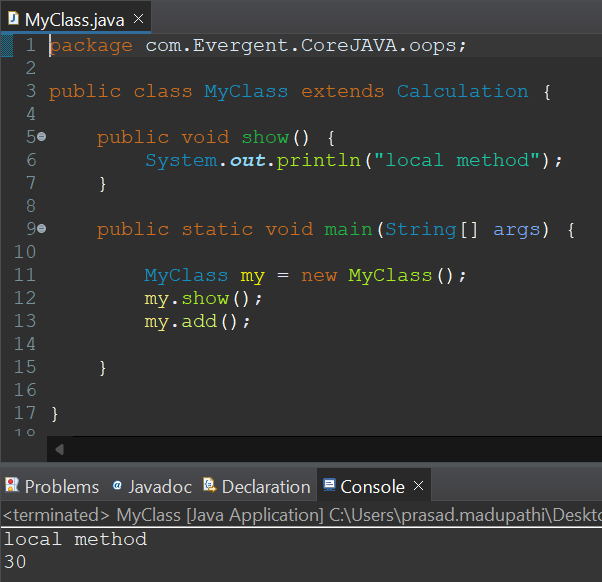
**Day 3: 07-Aug-2024:**

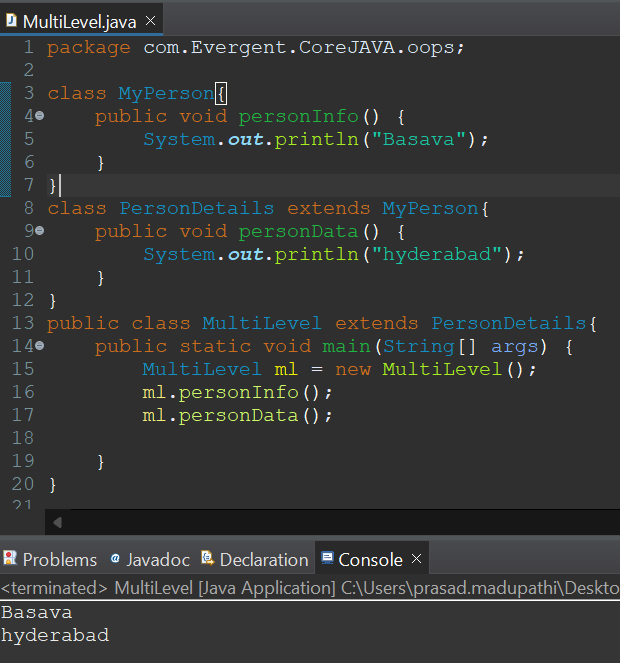
1. **OOPS**
   1. **Encapsulation:**
      1. Binding attributes and methods together inside a class and object creation is called Encapsulation.
      2. We cannot access attributes and methods without object creation. But we can access attributes and methods through object creation with reference.



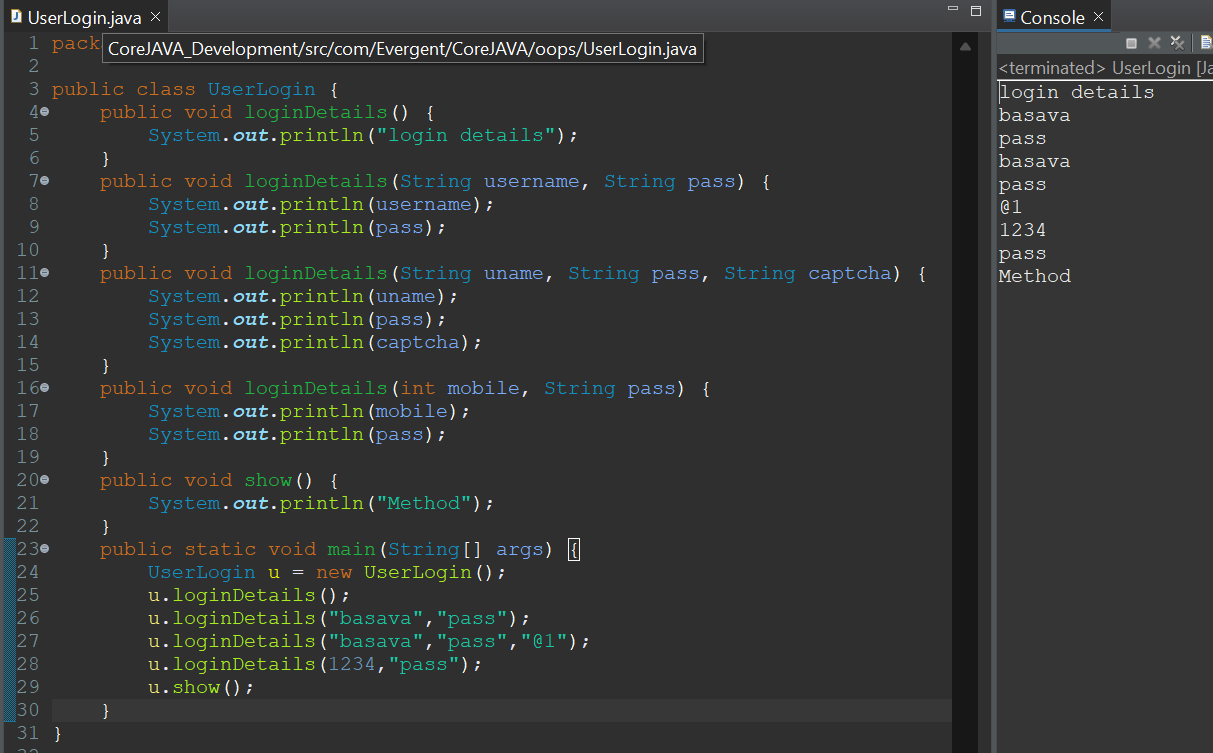


* 1. **Inheritance:**
     1. Re-usability of existing functionalities from super class to its subclass.
     2. Java won’t support multiple inheritance through classes, but it will support through Interfaces.

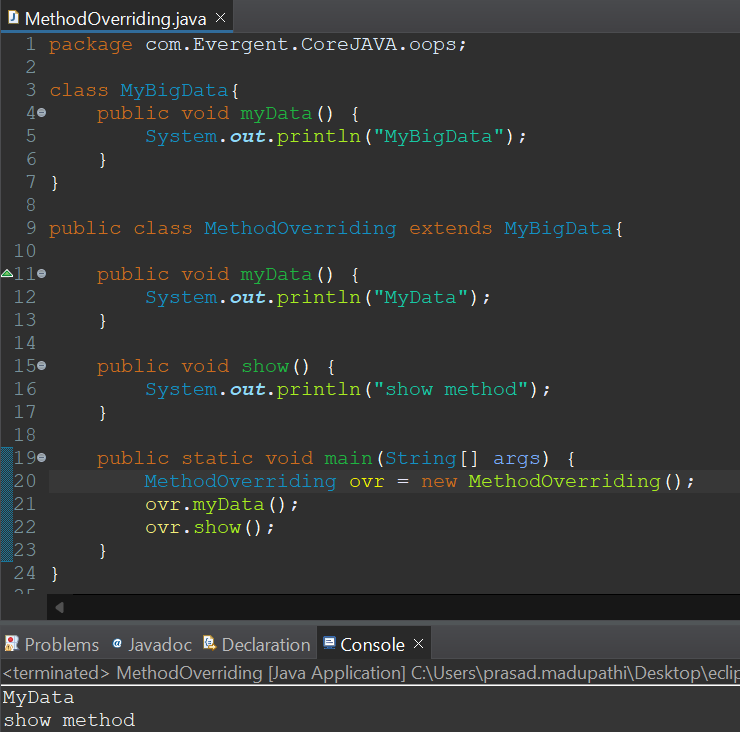




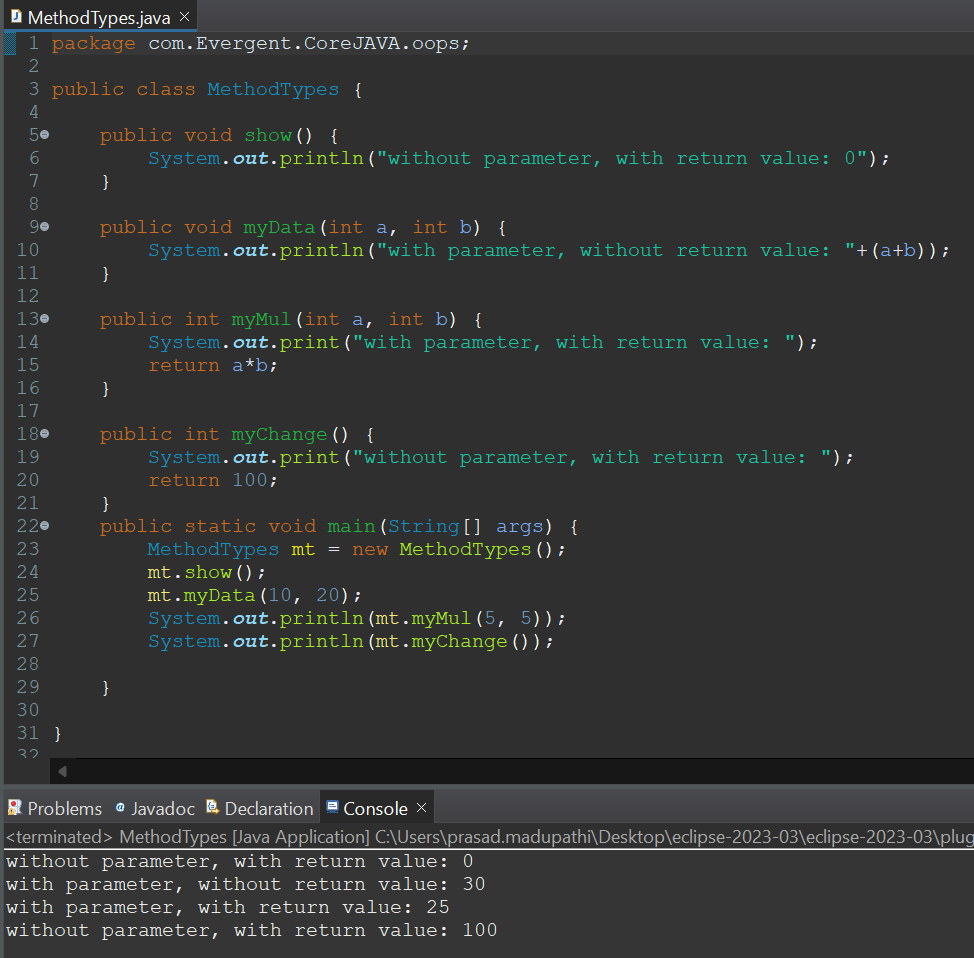
* 1. **Polymorphism:**
     1. Overloading:
        1. Method names are same, parameters should be different, return type may or may not be same, It happens in same class or different class.



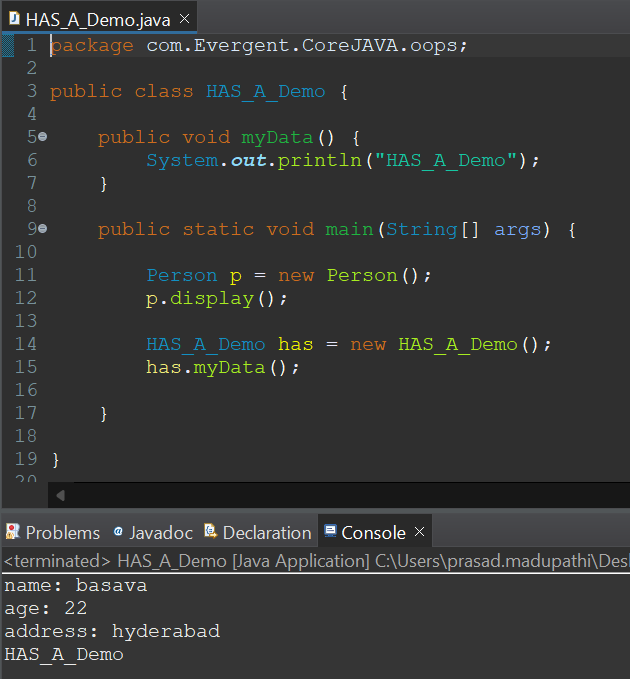
* + 1. Overriding:
       1. Method names are same, parameters same, return type also same, It will happen in two different classes through Inheritance.



* 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.
     4. Without parameters, Without return values.



* 1. IS-A and HAS-A relationship

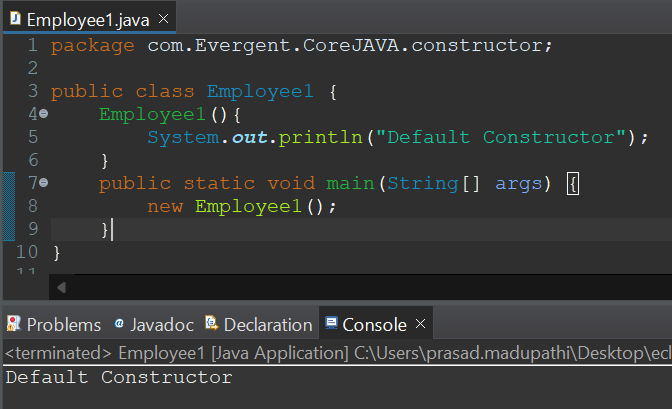


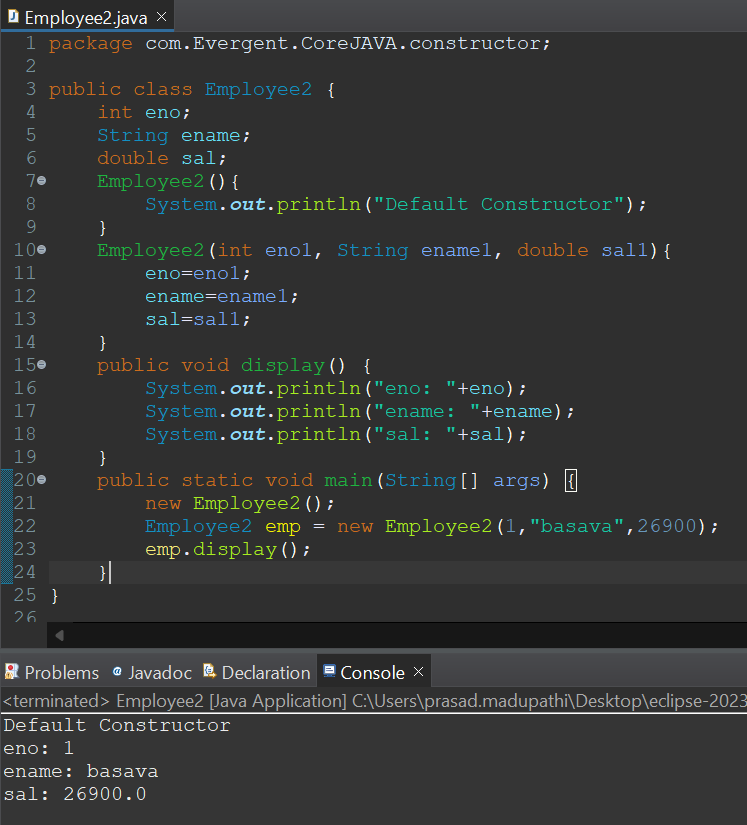
1. **System class**
   1. System - is a class
   2. Out - is reference of PrintStream
   3. Println - method

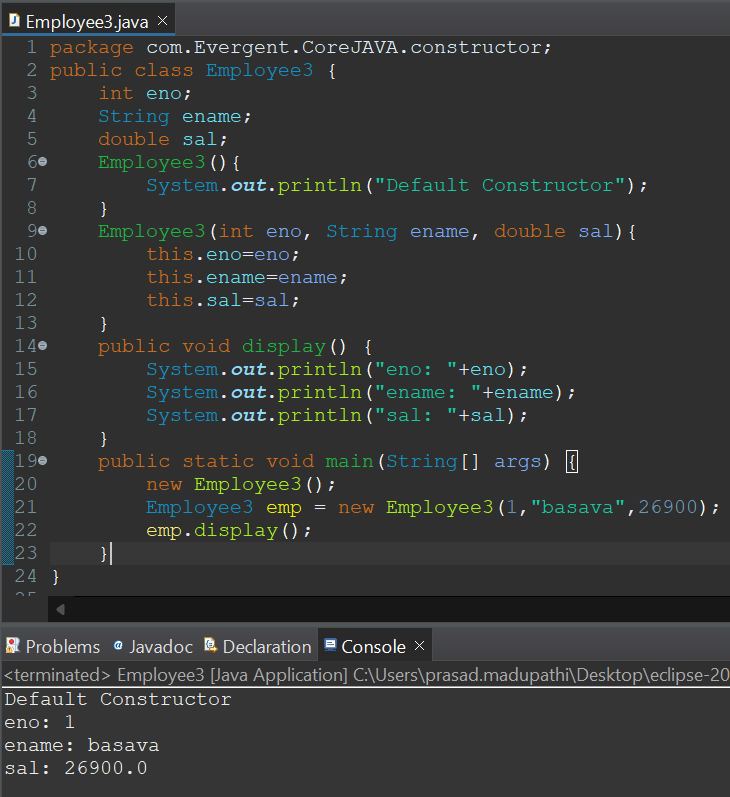
**Day 4: 08-Aug-2024:**

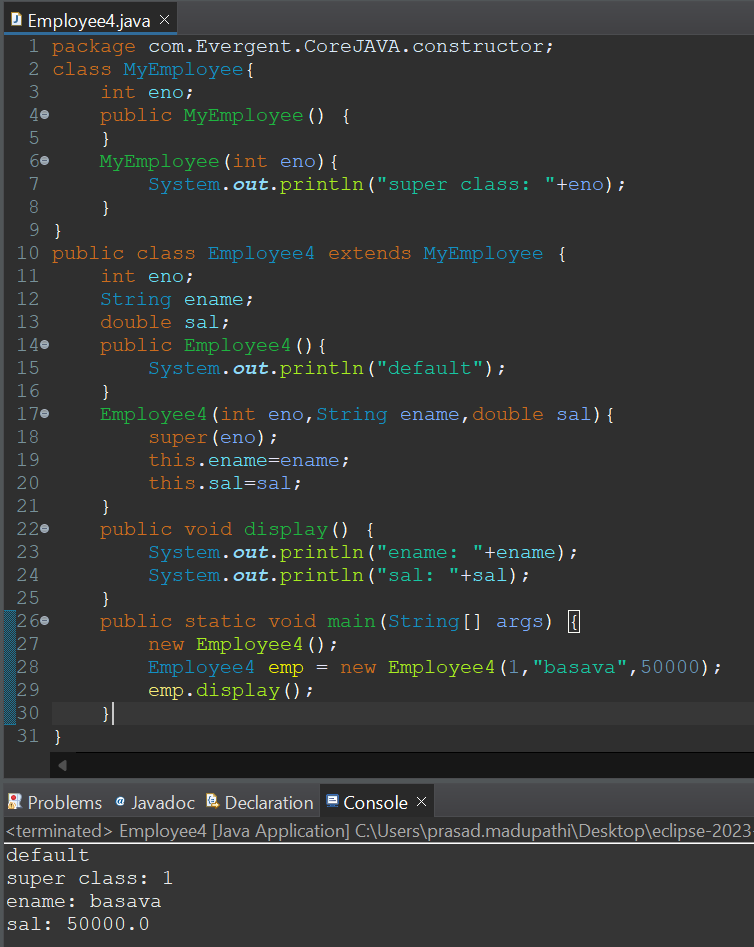
1. **Constructor:**
   1. Constructor is mainly used for initialization of object.
   2. Class name & Constructor name should be same.
   3. There are two types of constructors
      1. Default constructor
      2. Parameterized constructor
   4. We can access constructor while creation of object
   5. Constructor doesn’t have any return type not even void, if we declare as void it will consider it as a method.
   6. Every class has a default constructor.
   7. this, super
   8. Copy constructor
   9. Always constructors are overloaded

**Programs:**

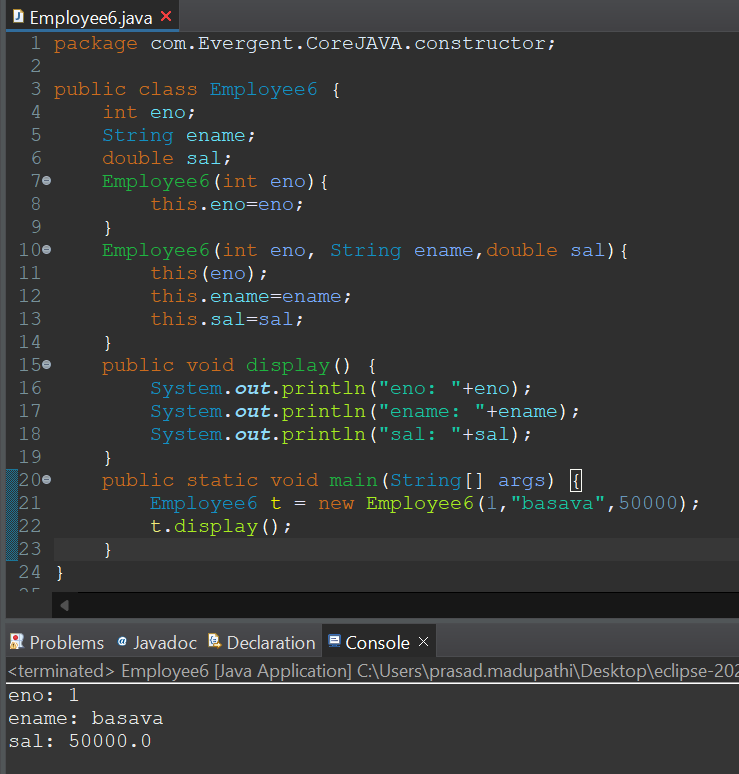
****

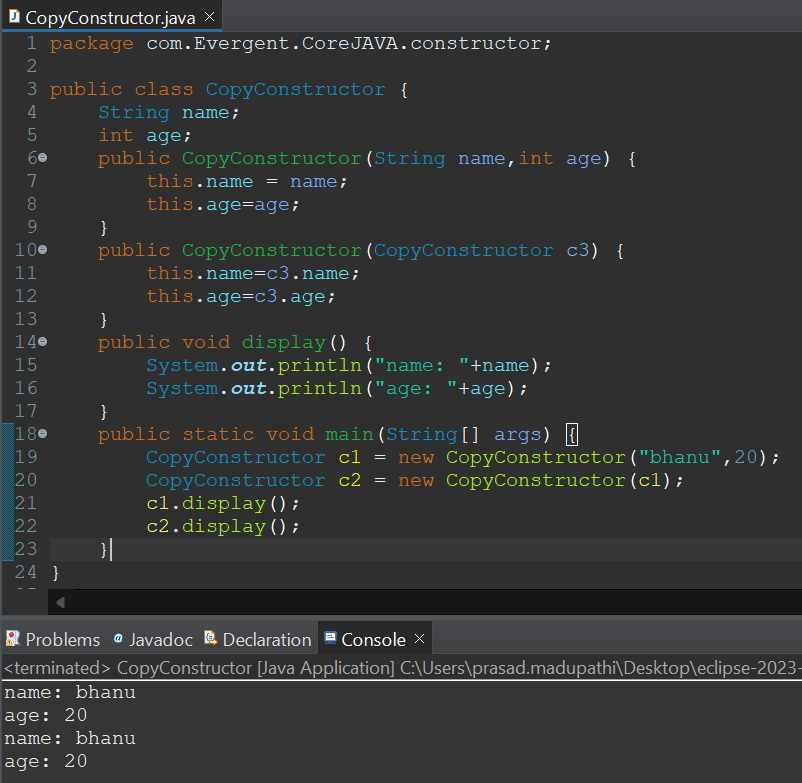
****





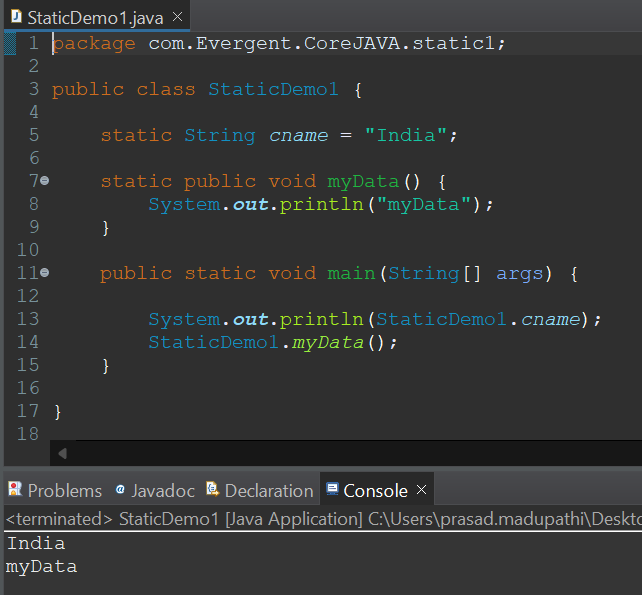


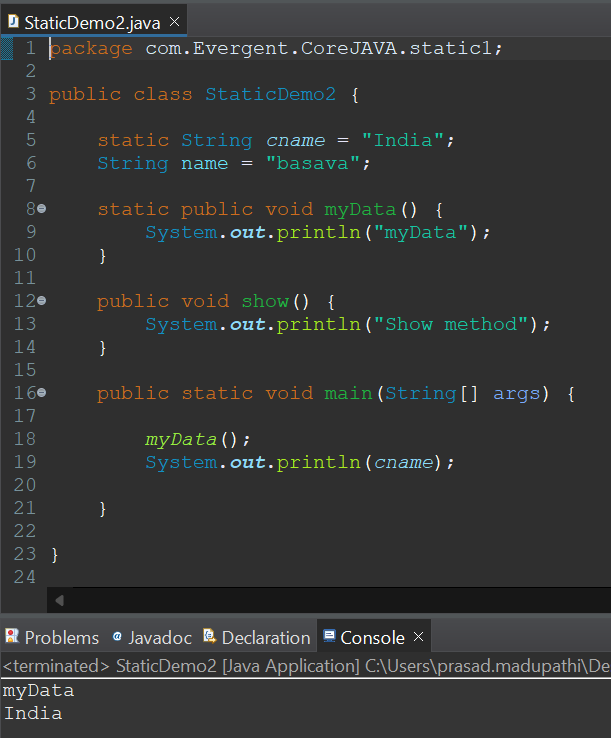


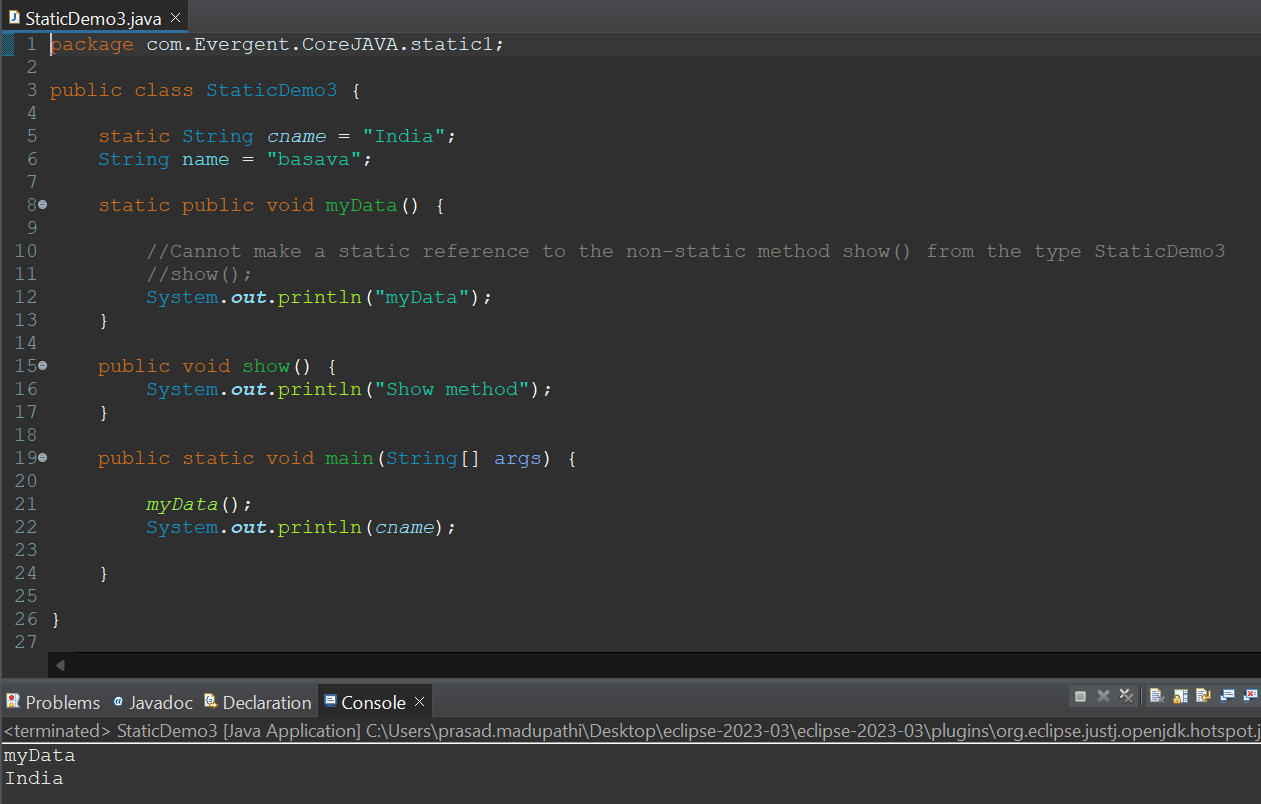


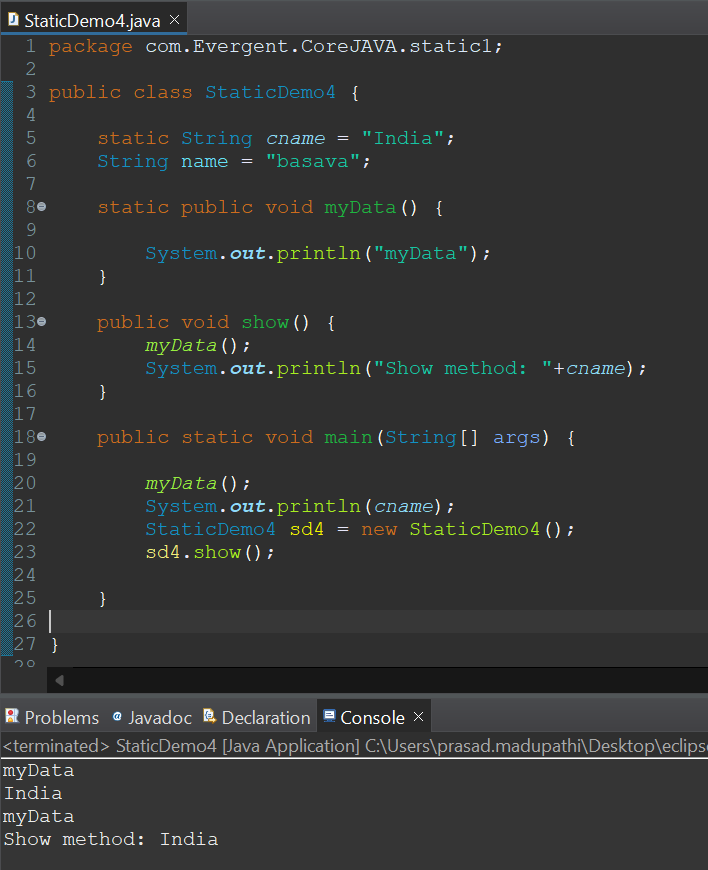
**Day 5: 09-Aug-2024:**

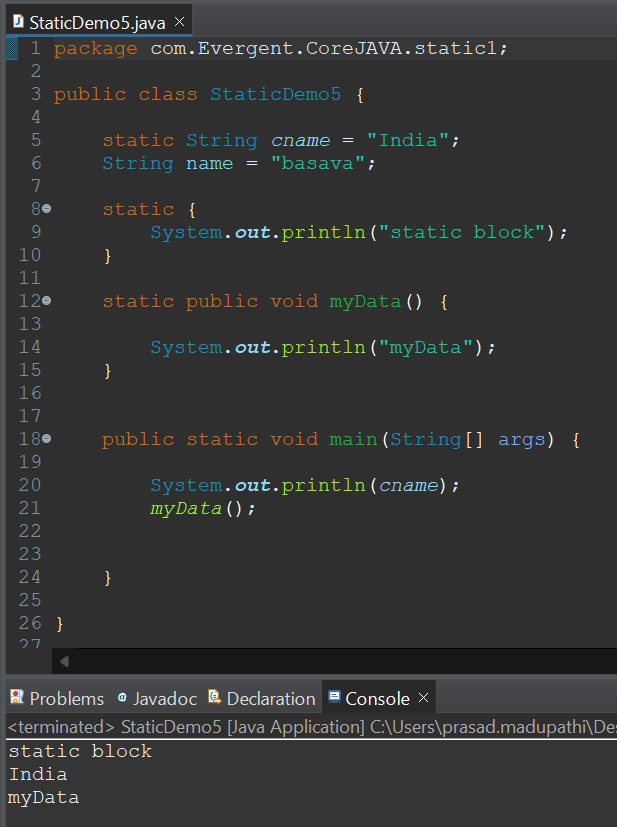
1. **Static:**
   1. Static is a keyword.
   2. We can declare static as variables, methods.
   3. We can access static variables and methods directly by classname.variablename, classname.methodname.
   4. Static methods can access static variables & methods.
   5. Static variables cannot access non-static variables & methods.
   6. Non-static methods can access static variables and methods.

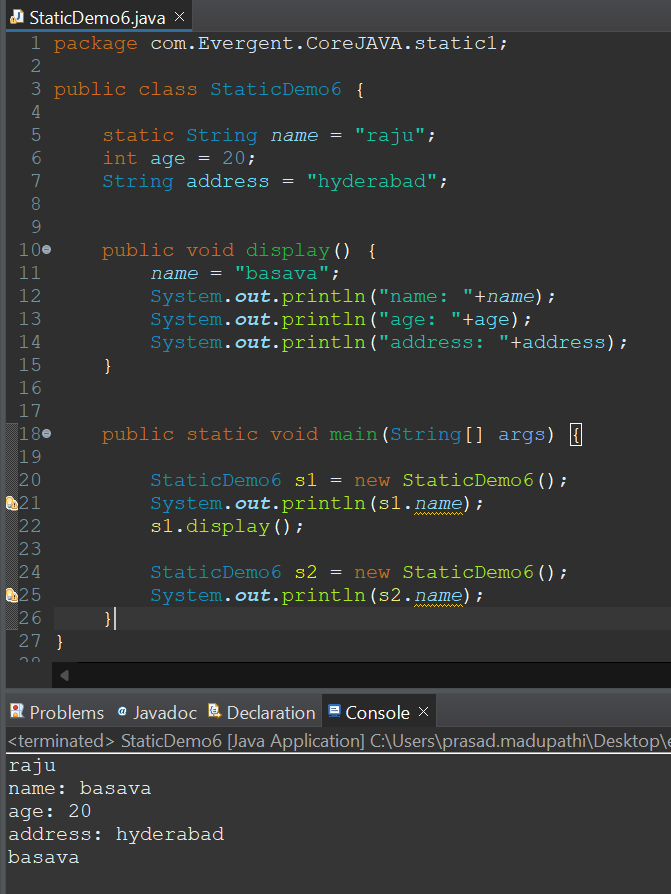




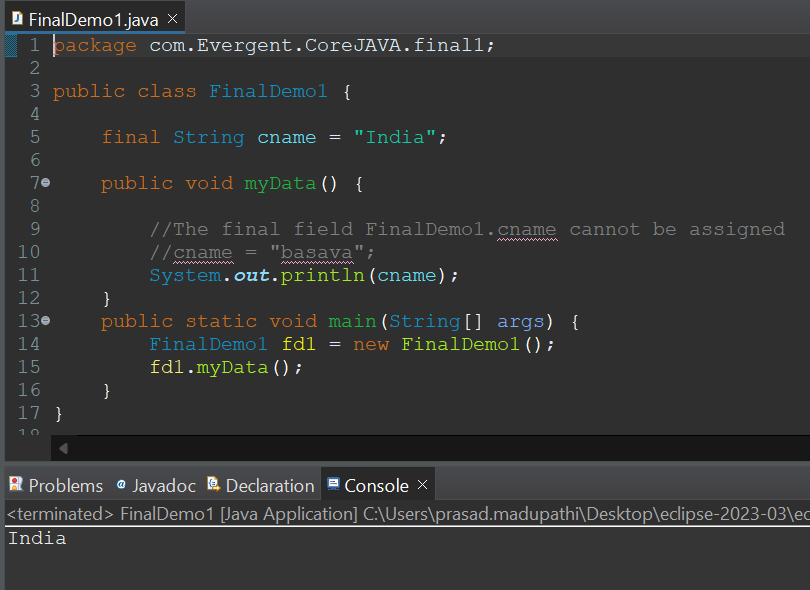




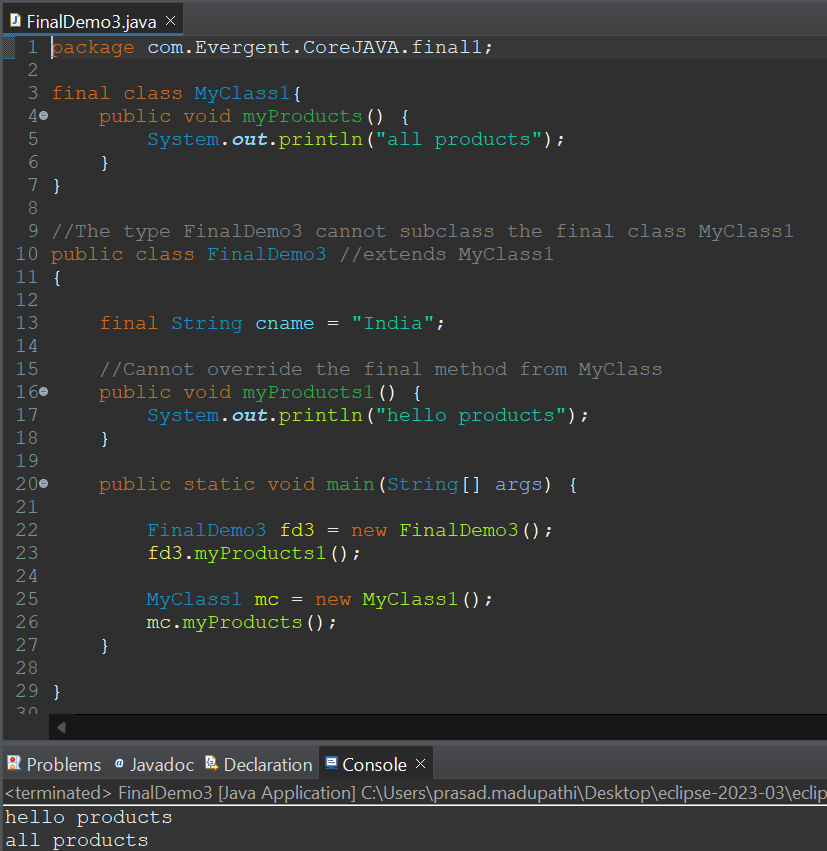




1. **Final:**
   1. Final is a keyword
   2. We can declare final as variables, methods and classes.
   3. Final variable - we cannot modify.
   4. Final methods - cannot be overridden.
   5. Final class - cannot be inherited.
   6. We can initialize final variables in constructors.
   7. Final methods can access through HAS-A relation.



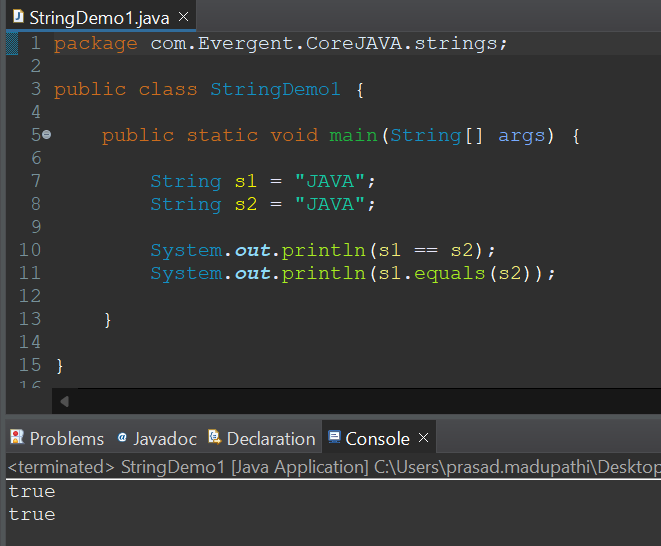
****

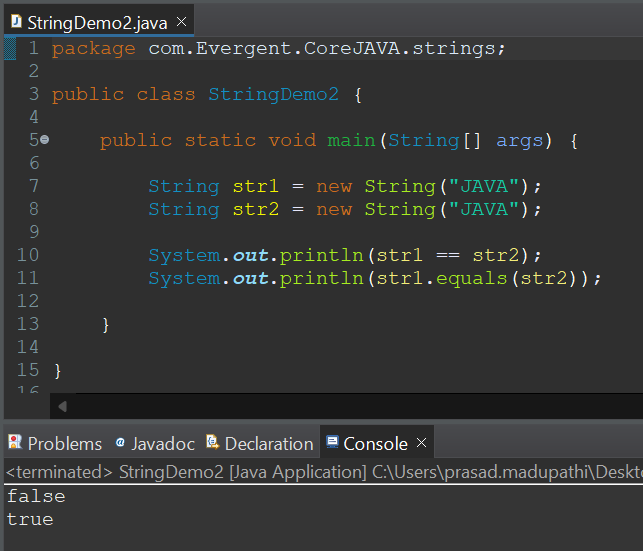
****

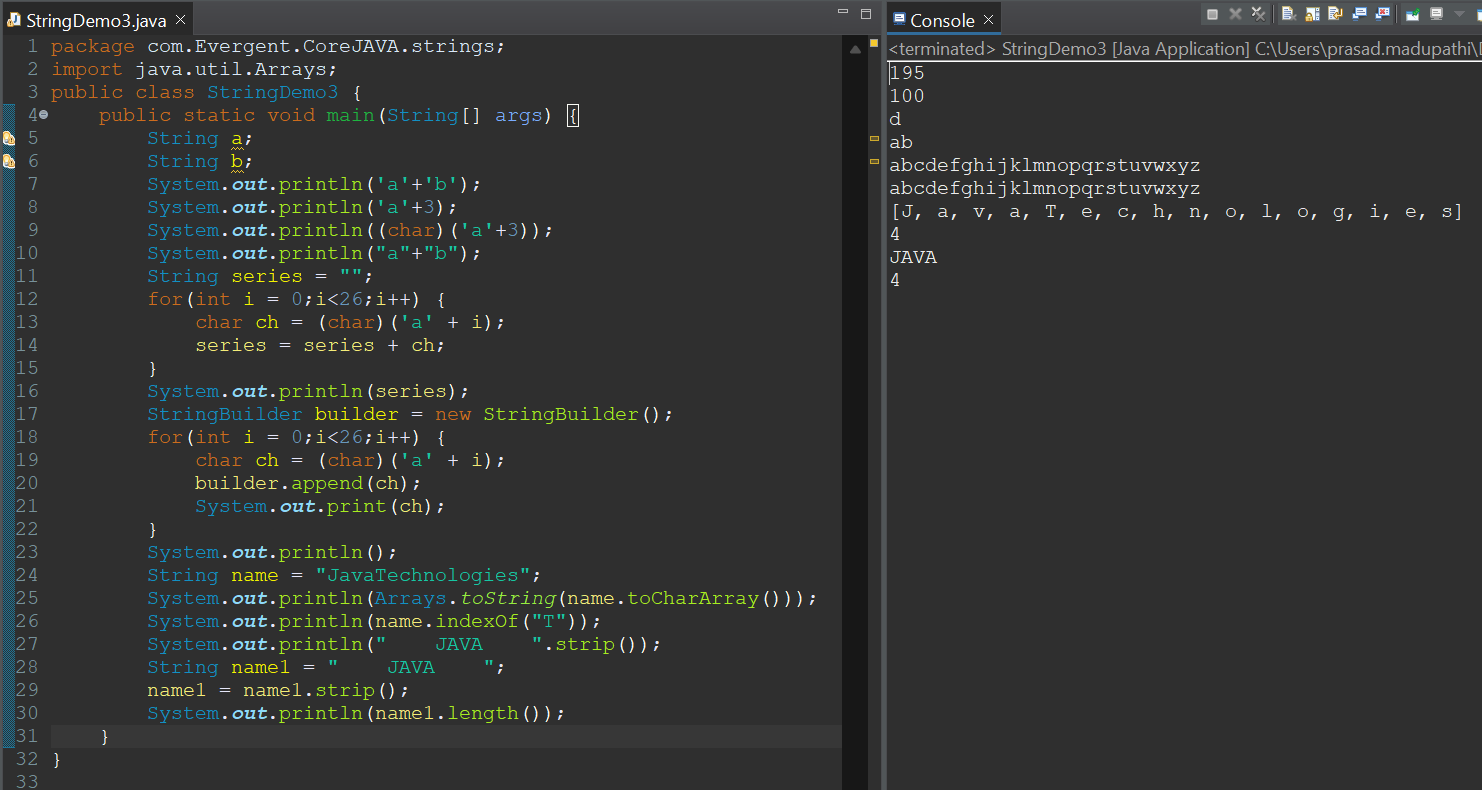
**Day 6: 12-Aug-2024:**

**Strings**

1. **String class:**
   1. String is a sequence of characters, often used to represent text.
   2. Strings are objects in java and are instances of the string class, which is part of the java.lang package.
   3. Key features:
      1. Immutable: Once string object is created, it cannot be changed. Any modification to string creates a new string object.
   4. Java optimizes memory usage by storing strings in a special area of memory known as string constant pool.
   5. If two strings have the same value and are created without using new keyword, they will reference the same object in the string pool.
   6. We can create a string in java in multiple ways:
   7. Using literals:
      1. String str = “hello world”;
   8. Using the new keyword:
      1. String str = new String(“hello world”);
   9. String class methods:
      1. length();
      2. toUpperCase();
      3. toLowerCase();
      4. trim();
      5. contains();
      6. concat();
      7. split();

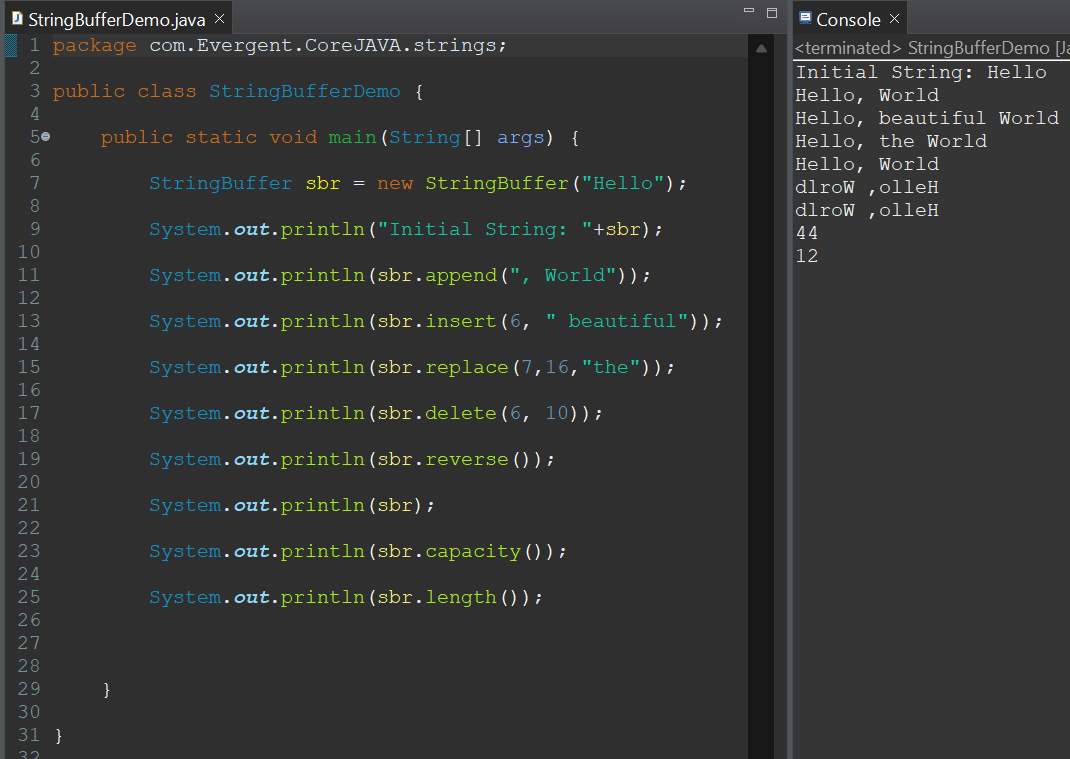




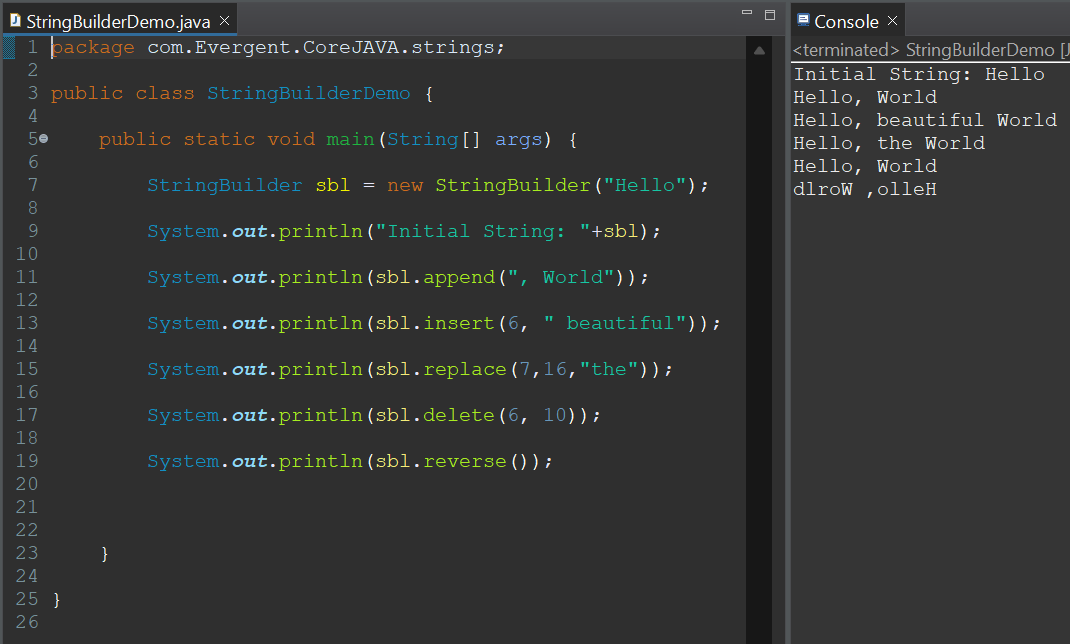




1. **StringBuffer:**
   1. StringBuffer is a final class.
   2. StringBuffer is Mutable.
   3. All methods in StringBuffer are synchronized(thread safe).
   4. It is legacy API(not recommended to use).
   5. StringBuffer methods are:
      1. append();
      2. length();
      3. insert();
      4. replace();
      5. delete();
      6. reverse();
      7. capacity();

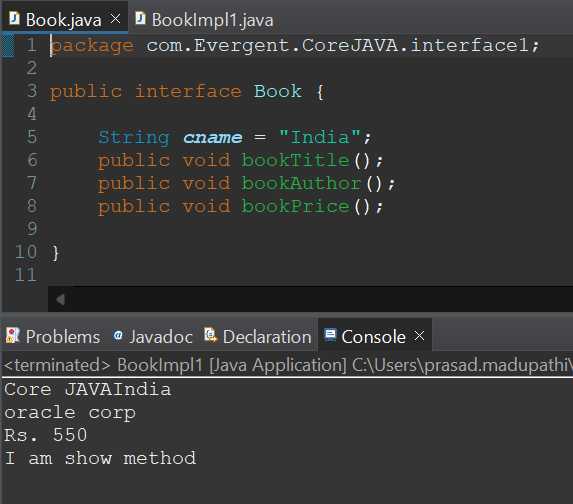


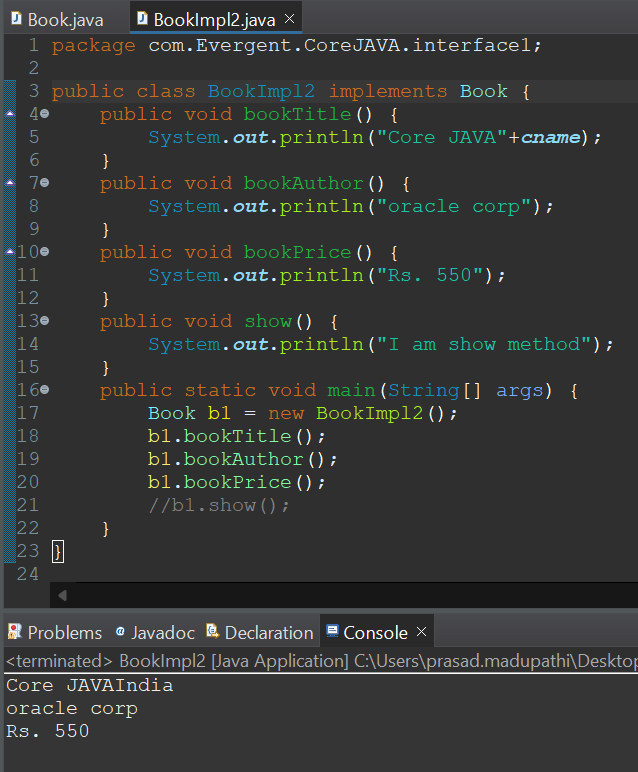
1. **StringBuilder:**
   1. StringBuilder is final class.
   2. StringBuilder is Mutable.
   3. All methods in StringBuffer are non-synchronized.
   4. StringBuffer methods are:
      1. append();
      2. length();
      3. insert();
      4. replace();
      5. delete();
      6. reverse();
      7. capacity();

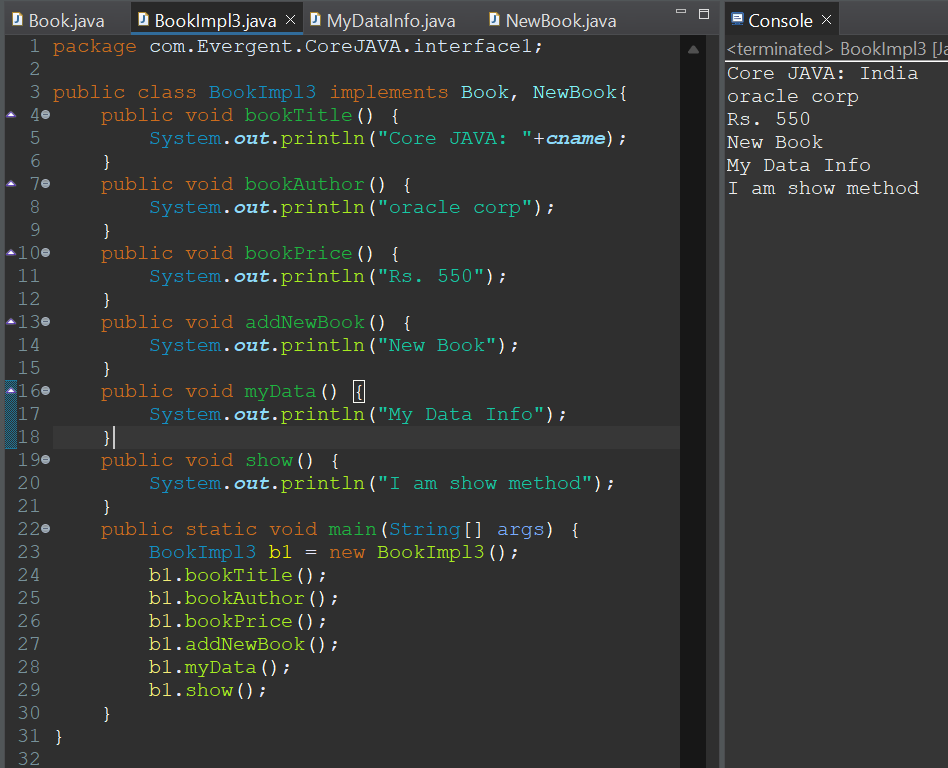


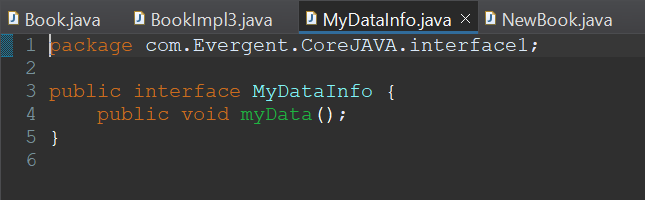
**Day 7: 13-Aug-2024:**

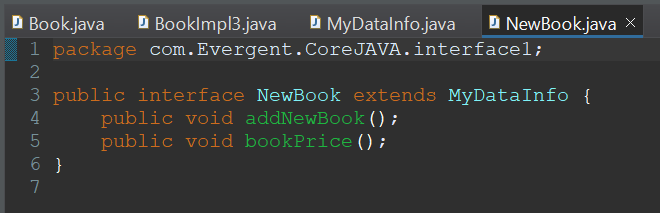
1. **Interfaces:**
   1. Interface is a keyword.
   2. We can declare method signature only but not implementation.
   3. By default all interface methods are abstract.
   4. If any class implements the interface then that class should be override all interface method, otherwise that class will be showing compile time error.
   5. We cannot create object to interface but we can create reference to interface.
   6. We can declare variables inside interface - by default (public static final) variables.
   7. JAVA will support multiple inheritance through interface.
   8. One class can implement interfaces.
   9. One interface extends other interfaces.
   10. We can declare interfaces without methods is called Marker Interfaces.
   11. Example:
       1. clonable
       2. serializable





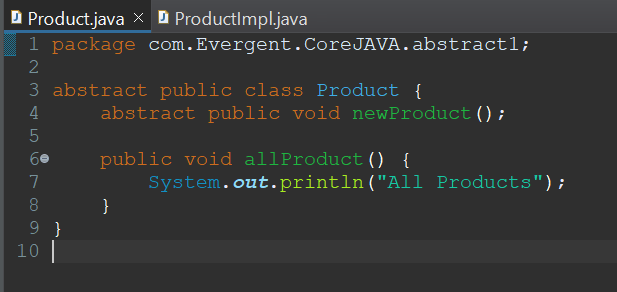




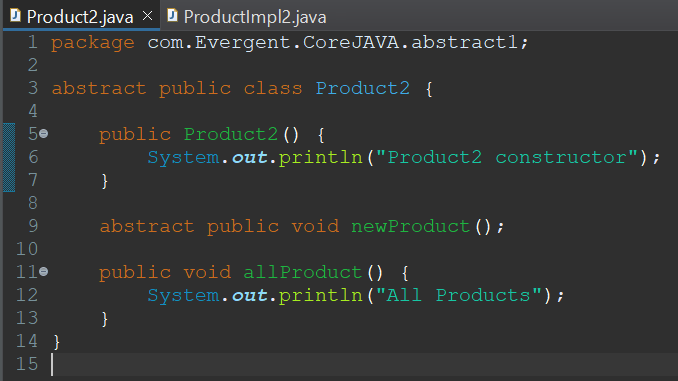


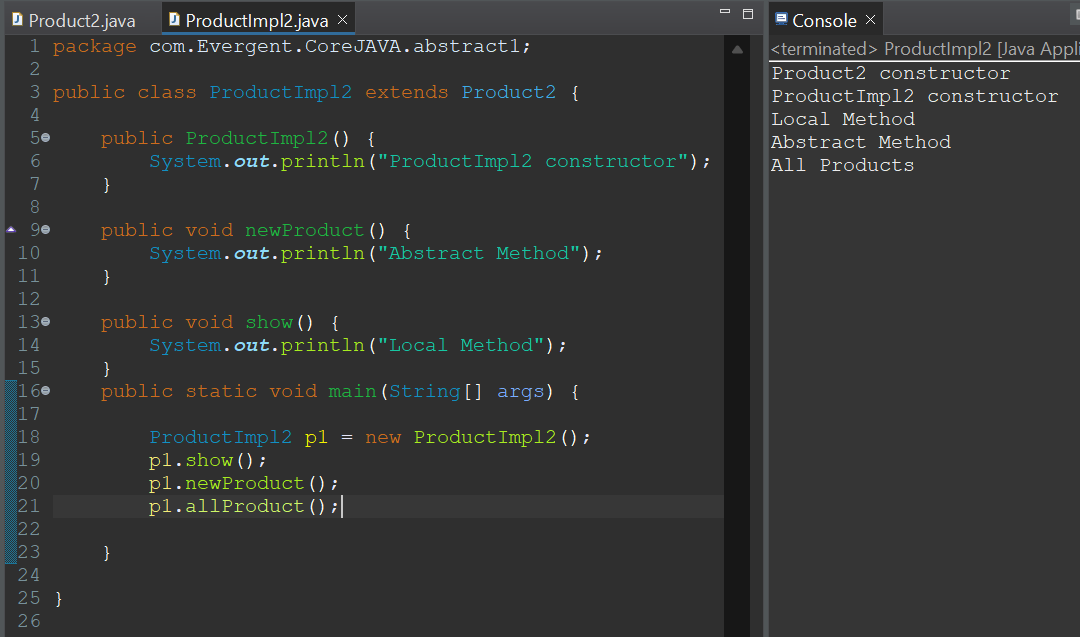
**Day 8: 14-Aug-2024:**

1. **Abstract class:**
   1. Abstract is a keyword.
   2. Abstract class can have abstract methods and concrete methods.
   3. 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.
   4. If any class extends abstract class that class should override all abstract methods, otherwise the class will be showing compile time error.
   5. We cannot create object to abstract class but we can create reference to abstract class.
   6. We have to declare abstract methods explicitly.
   7. We can create constructor to abstract class.
   8. We can access abstract class constructor through the sub-class object creation.



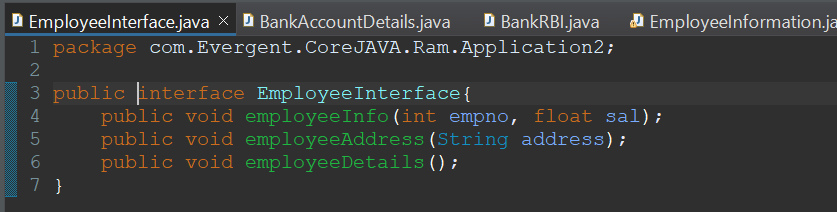


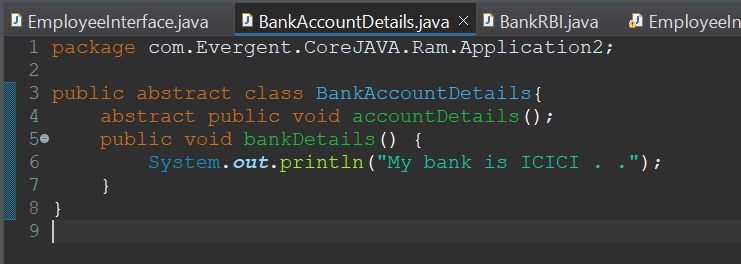


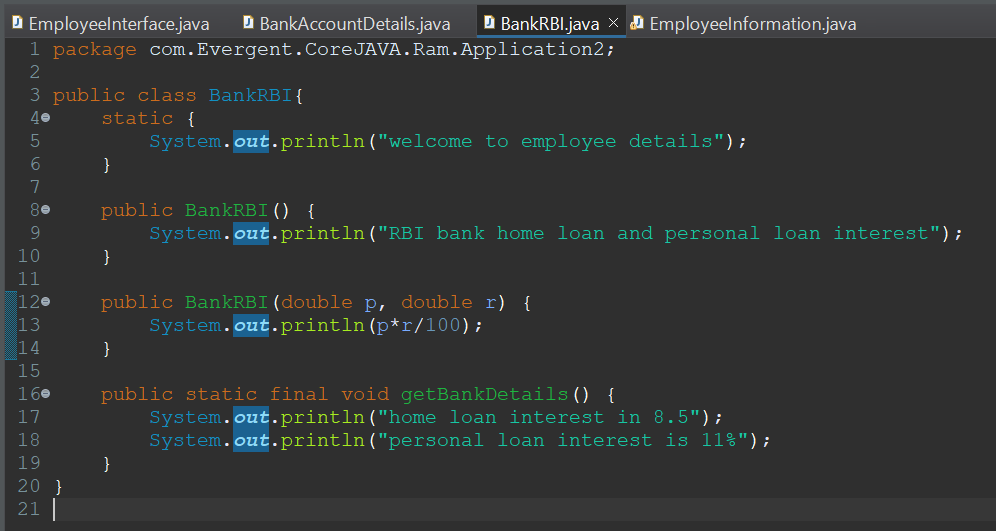


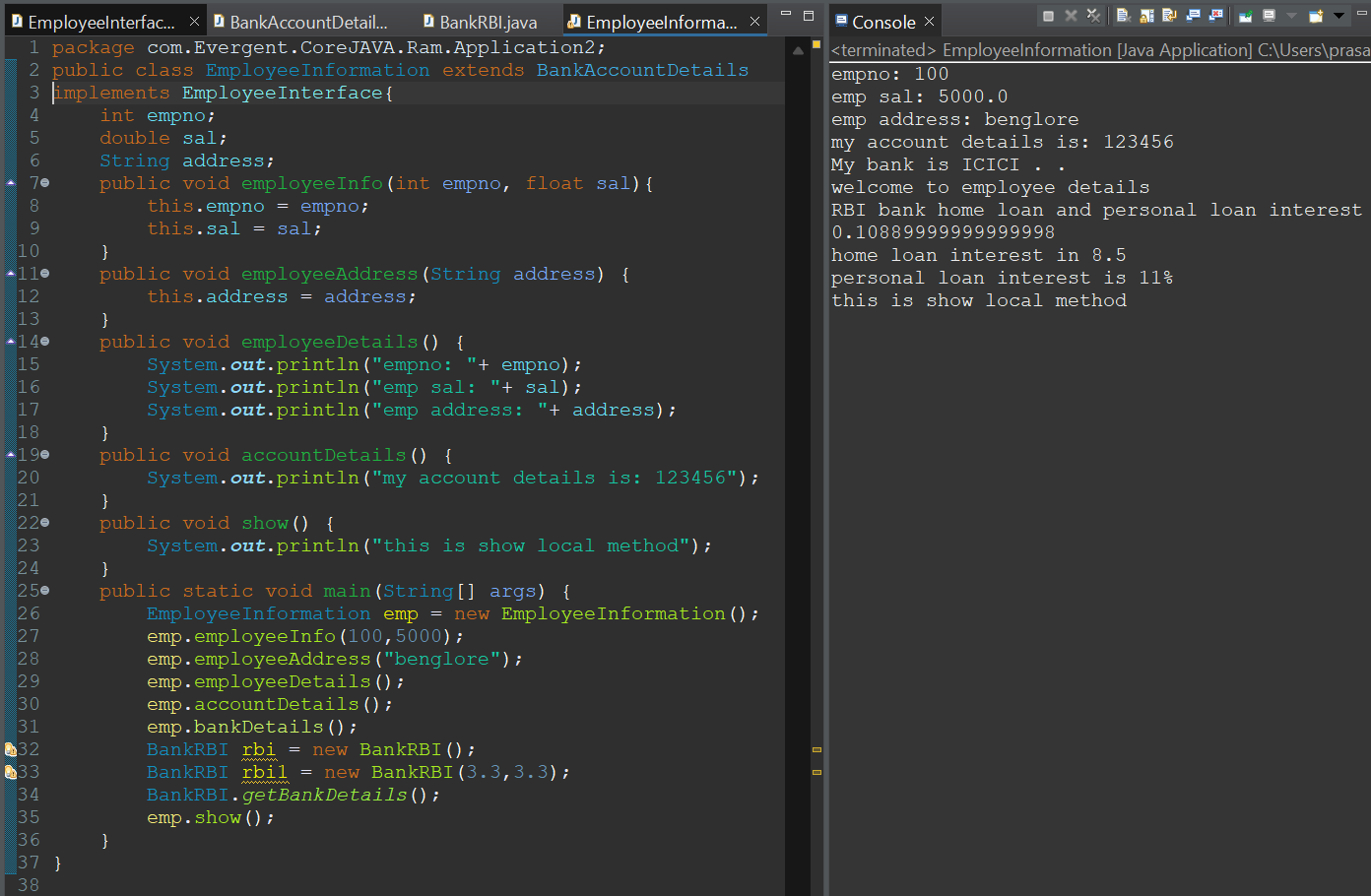
**Day 9: 16-Aug-2024:**

1. Java Concept Test
2. Application design and code

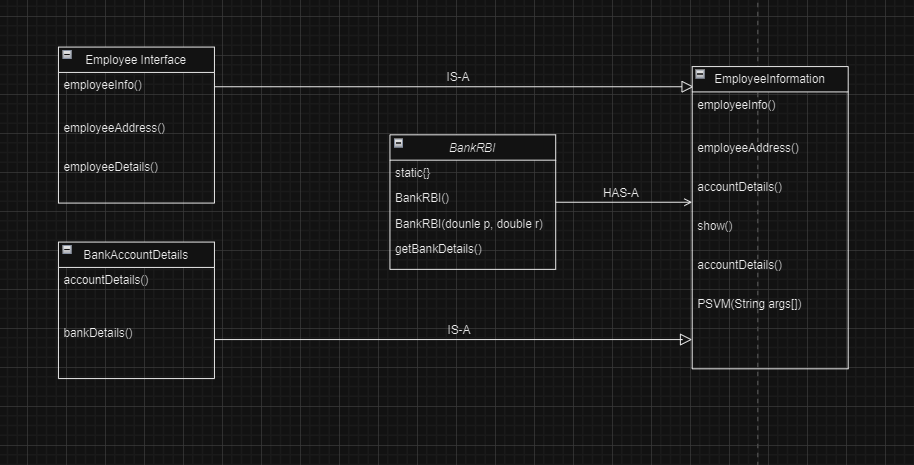






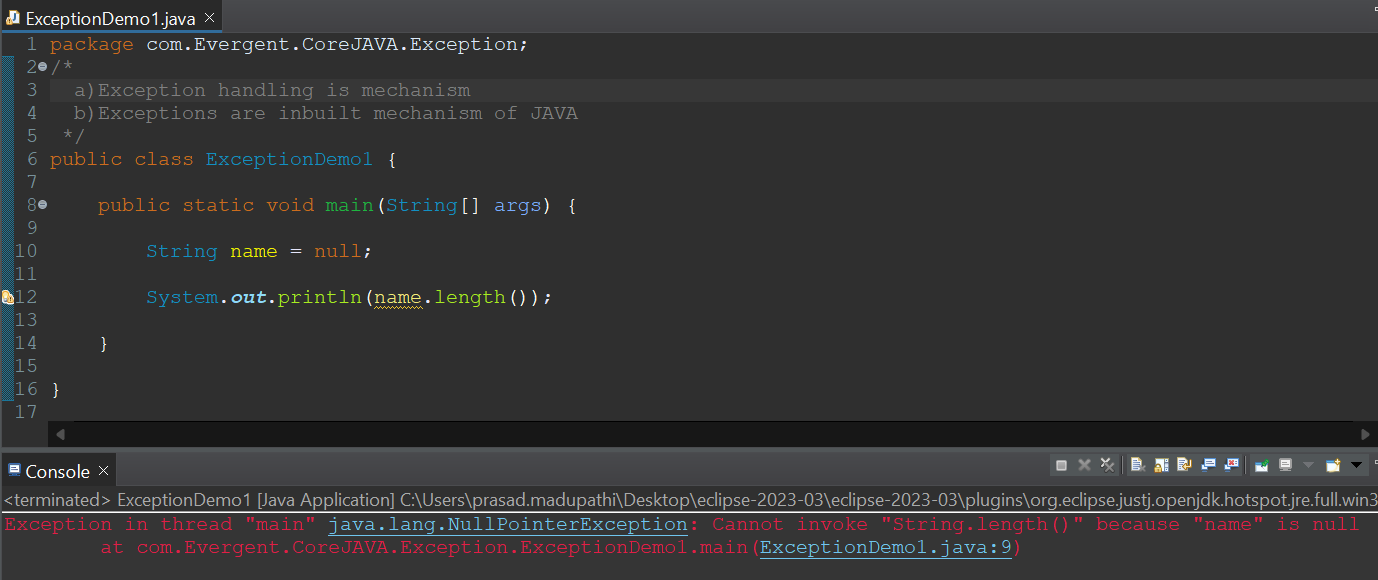


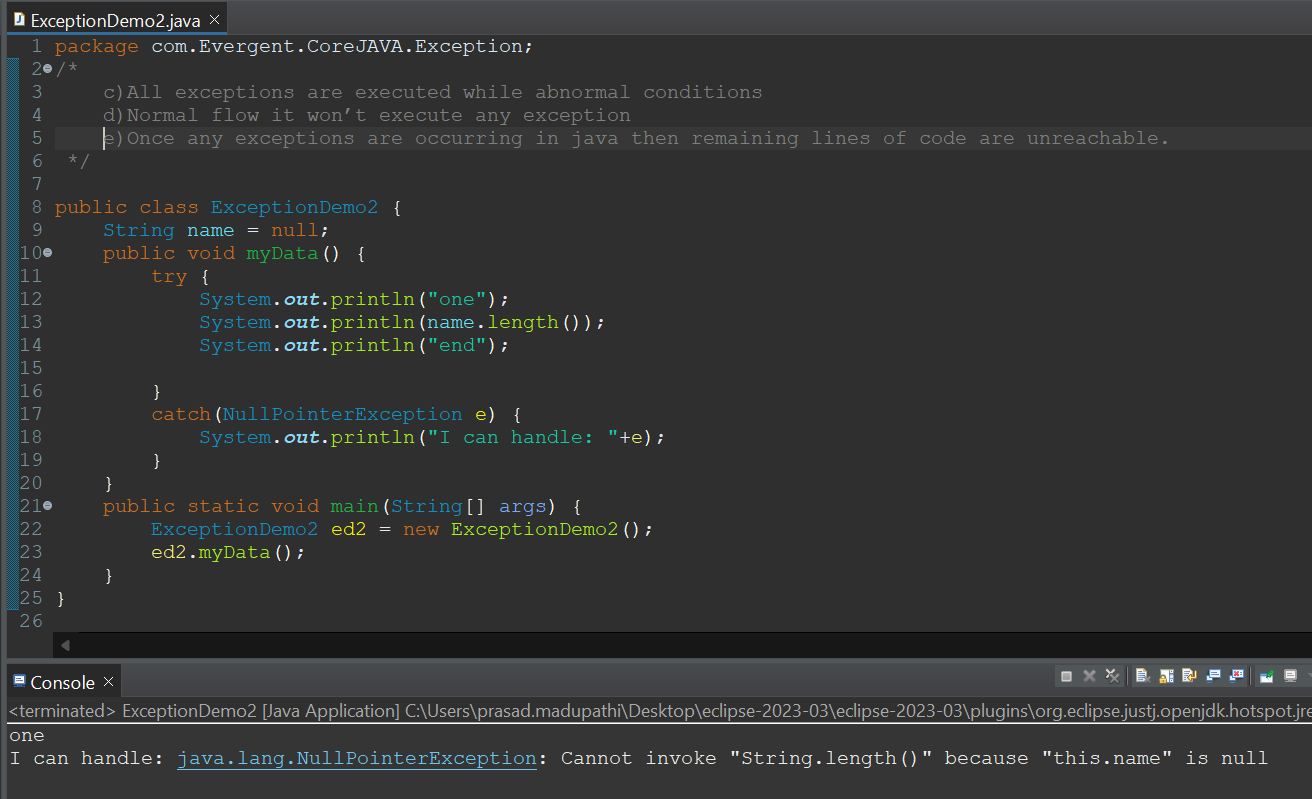
Design:

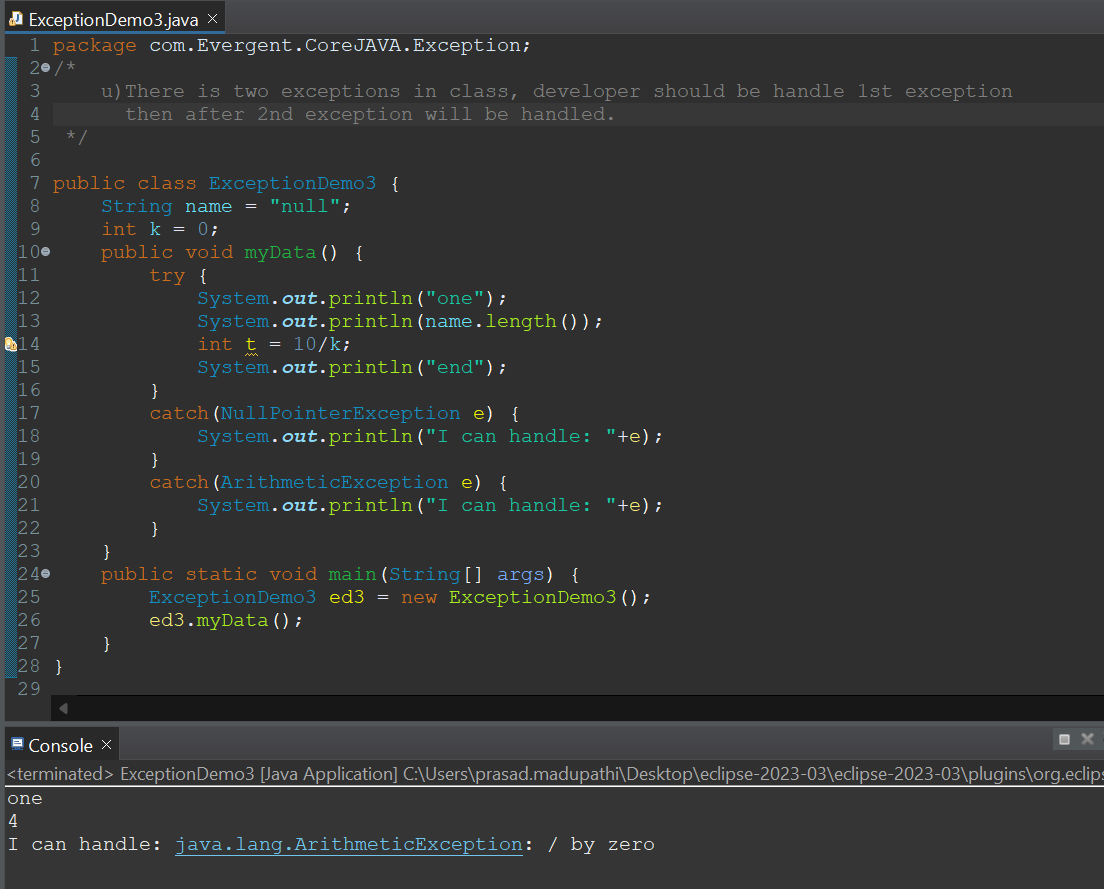


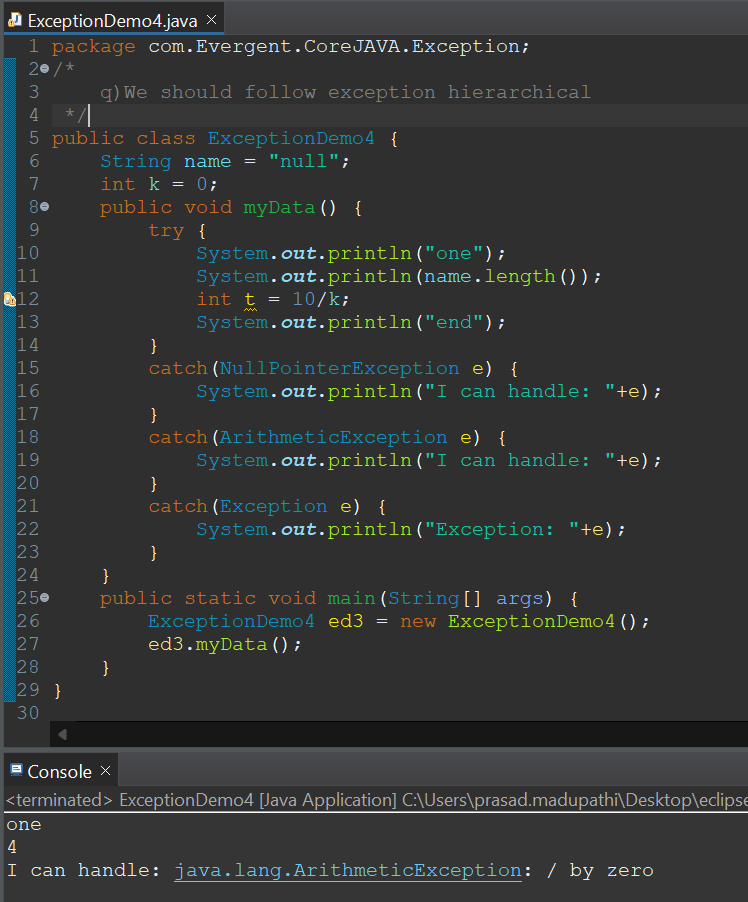
**Day 10: 19-Aug-2024 and Day 11: 20-Aug-2024:**

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

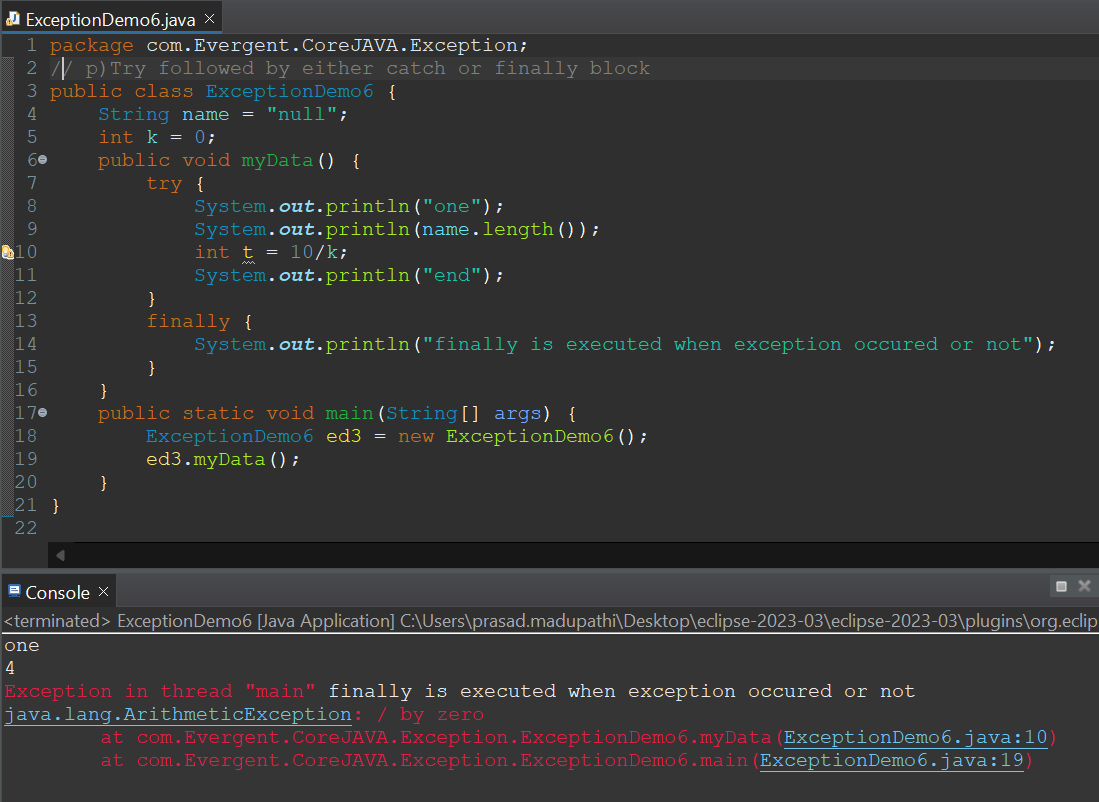


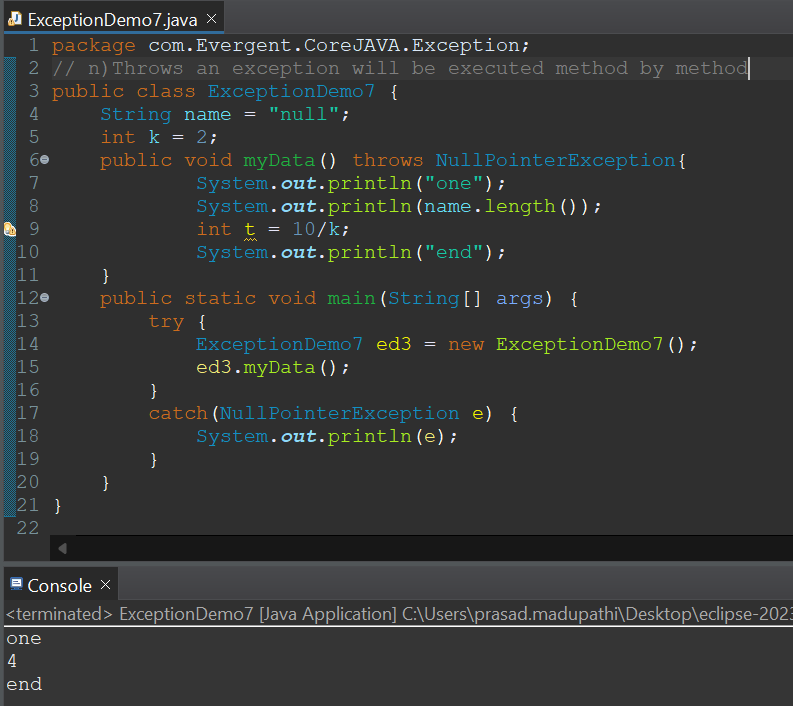


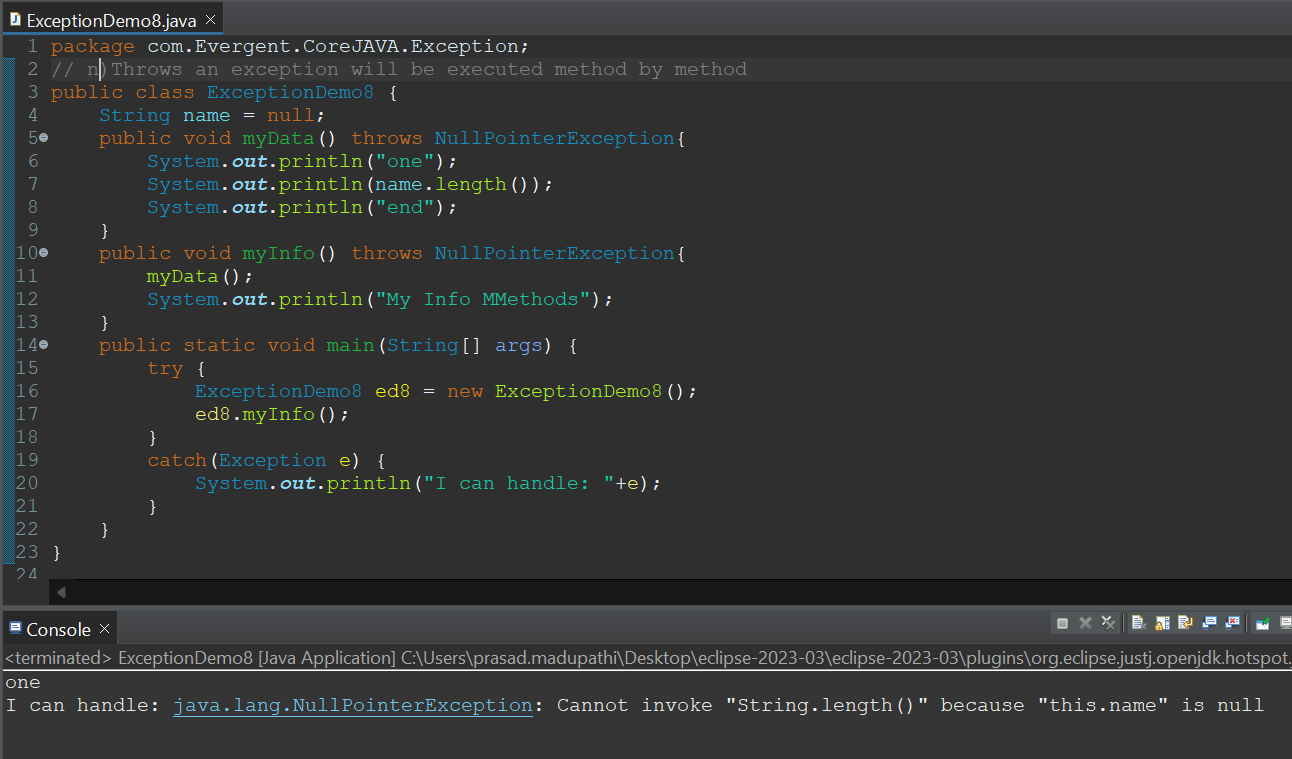


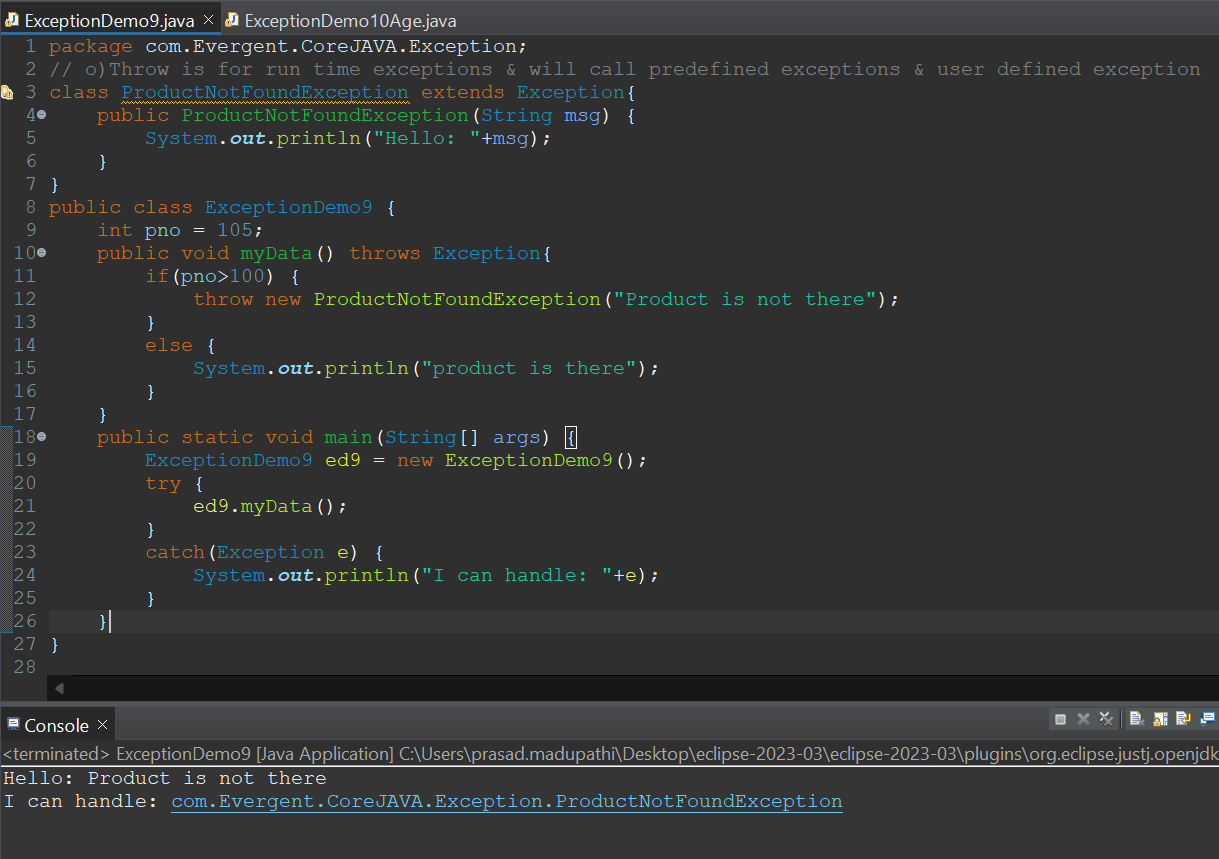




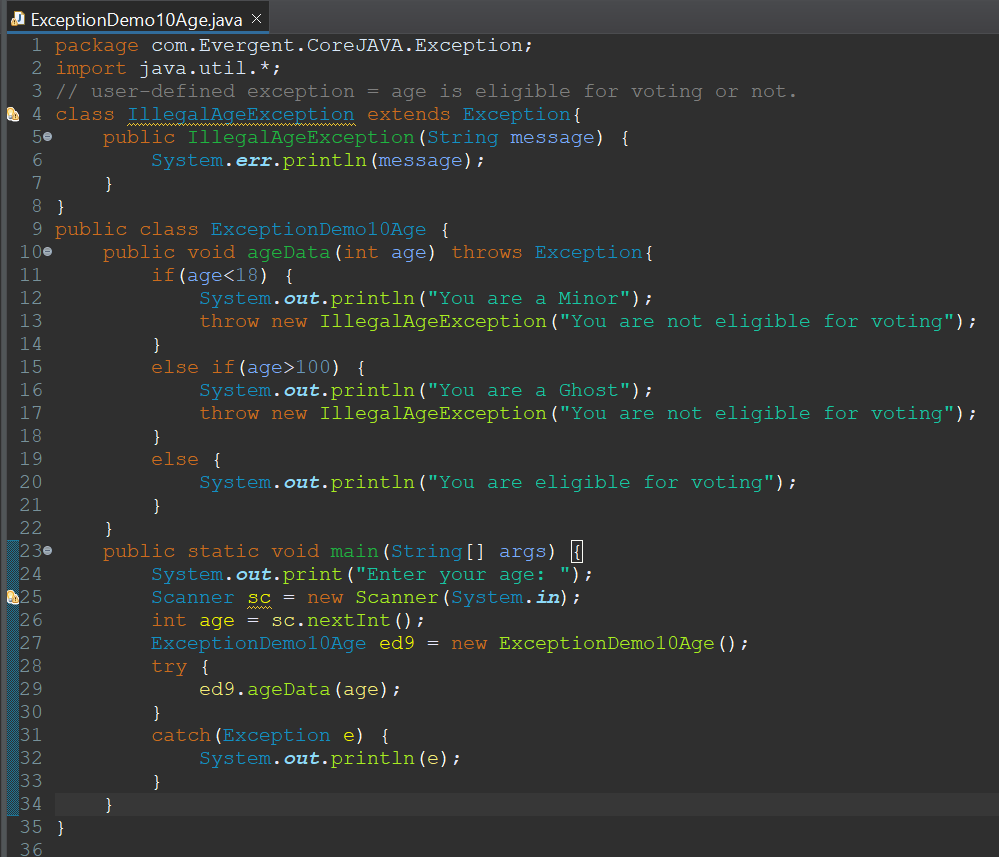




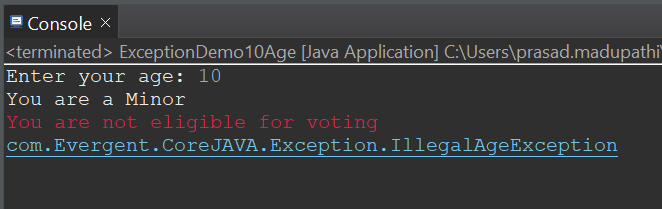




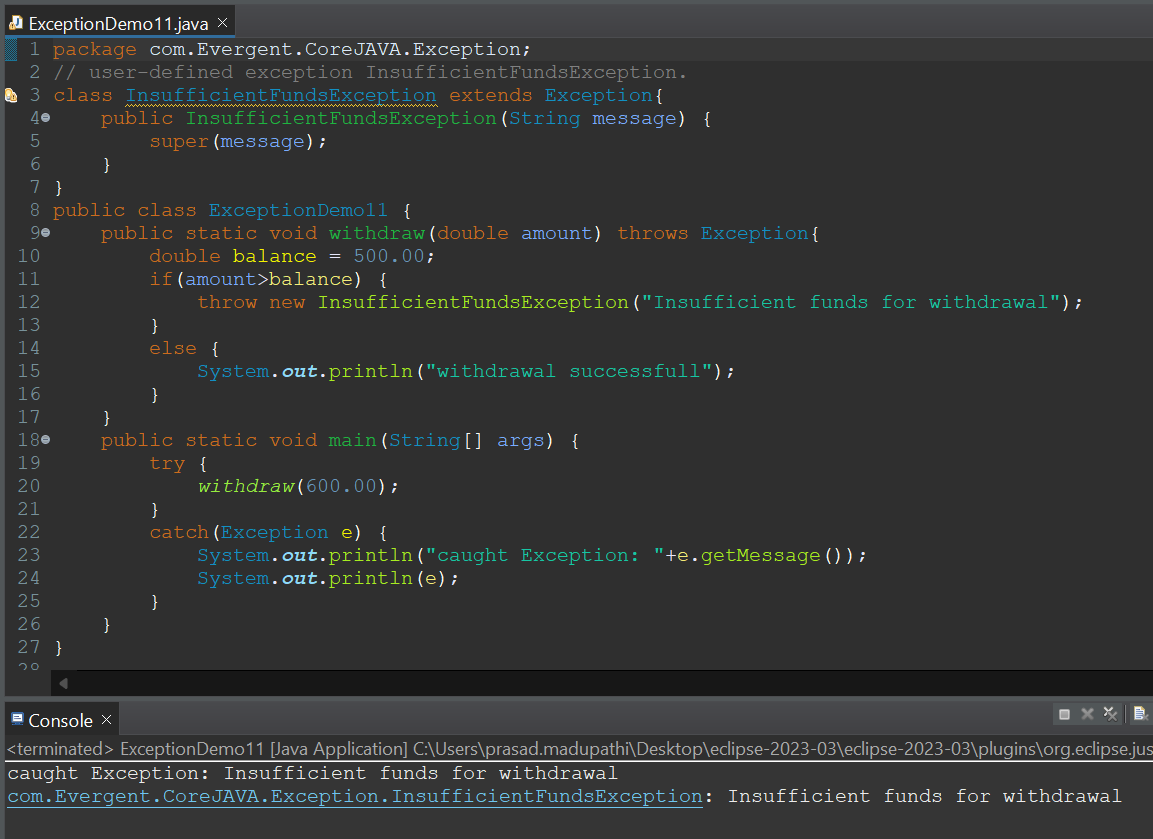
Code:

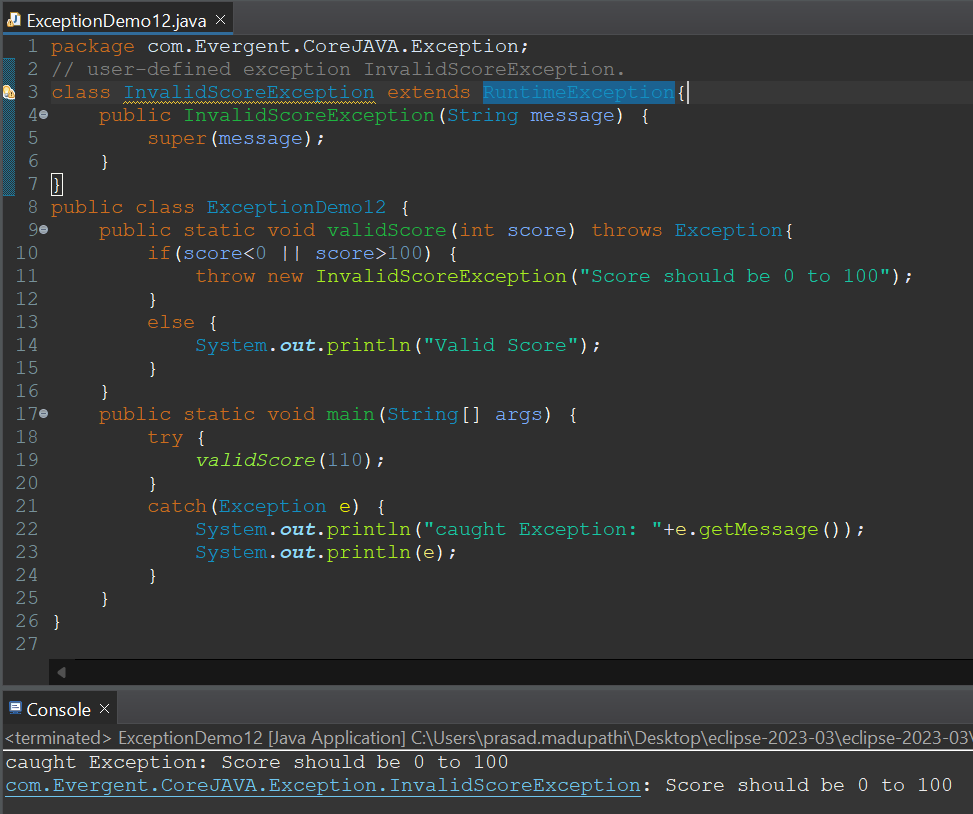


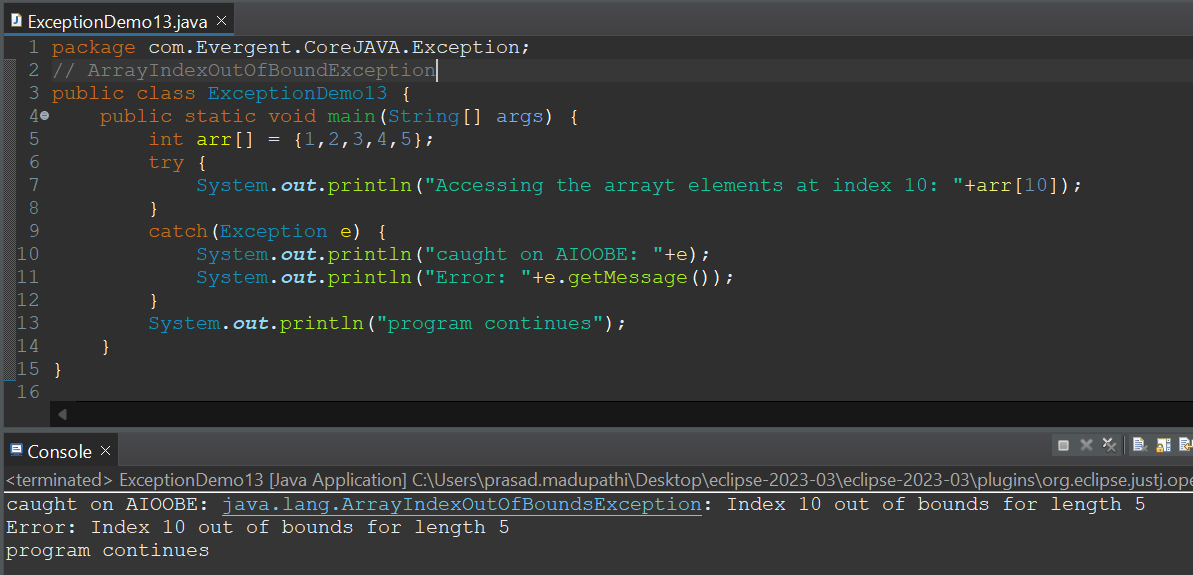
Output:

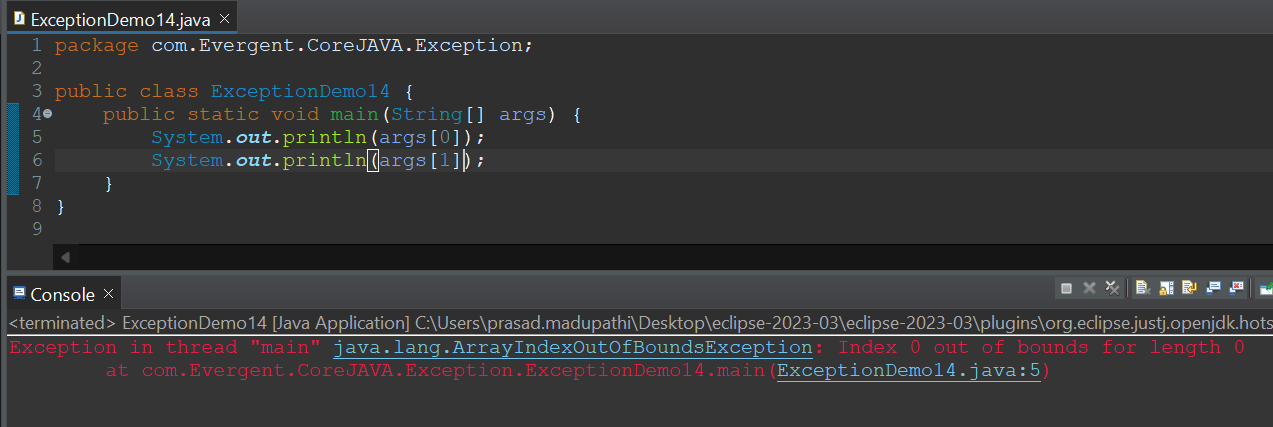


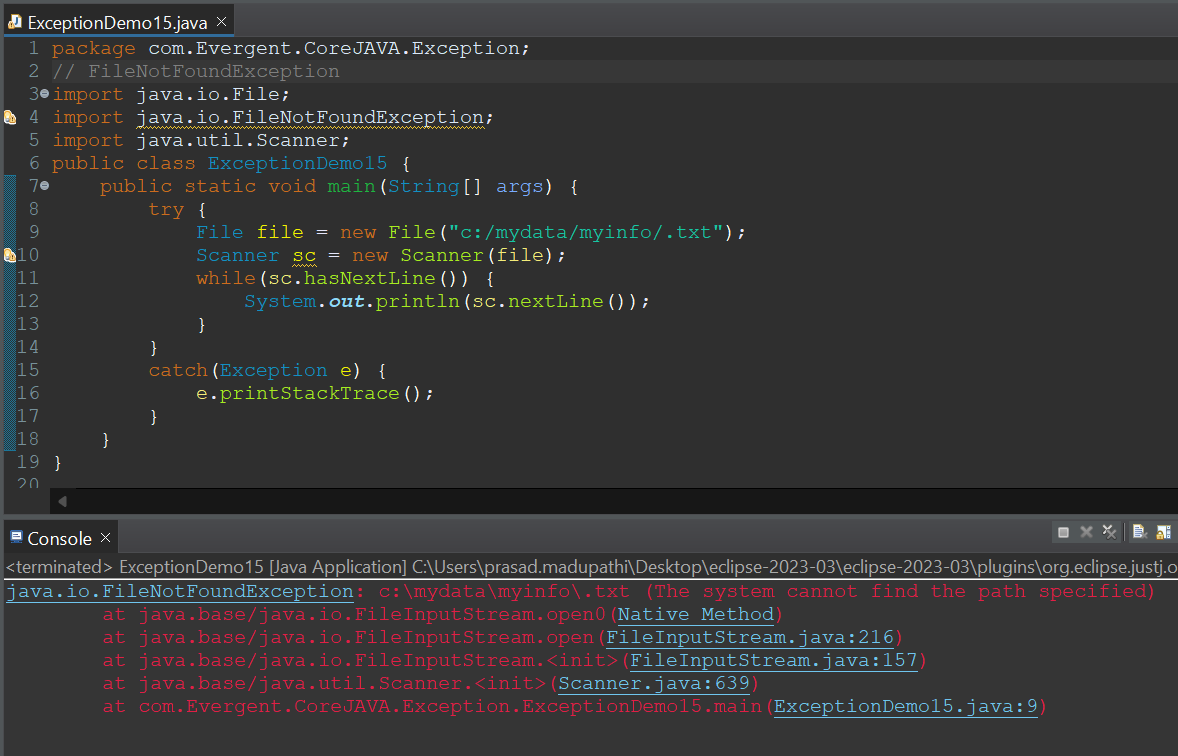
**Day 12: 21-Aug-2024:**







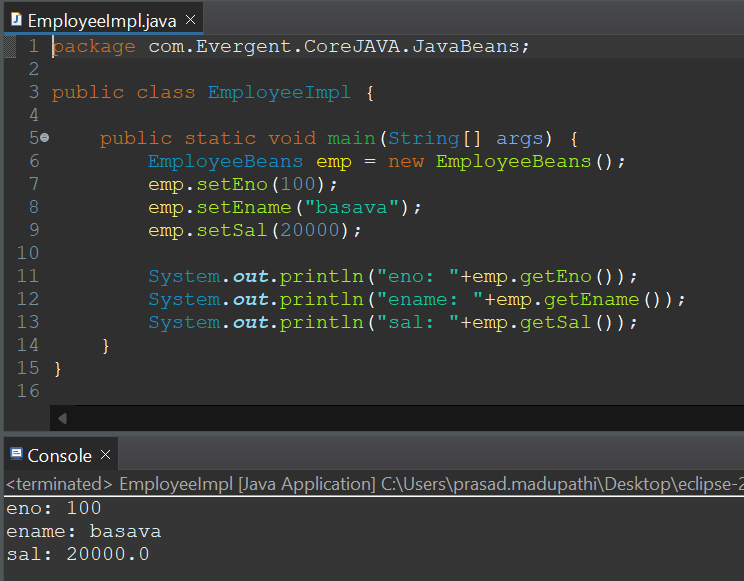




**JAVA Beans:**

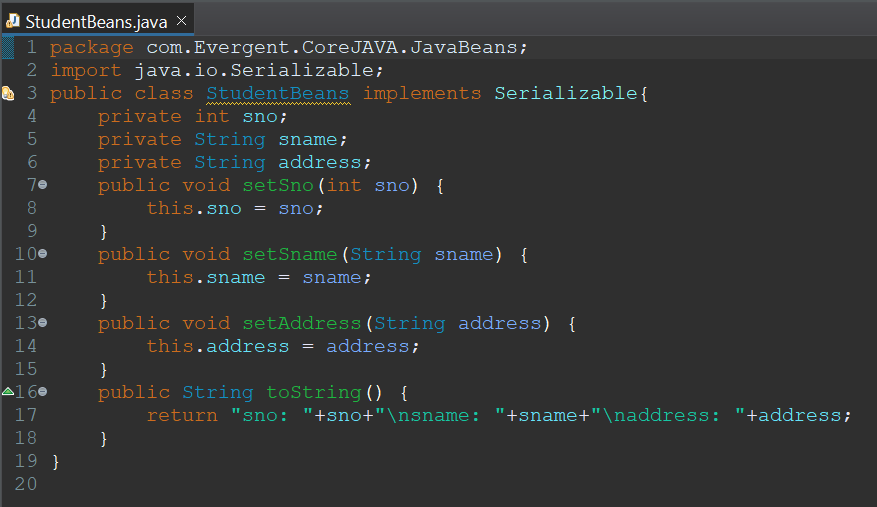
1. Java bean is a mechanism
2. Java bean is light weight
3. All attributes are private
4. get/set methods are public
5. Implements java.io.Serializable interface
6. We can achieve tightly encapsulation through java beans

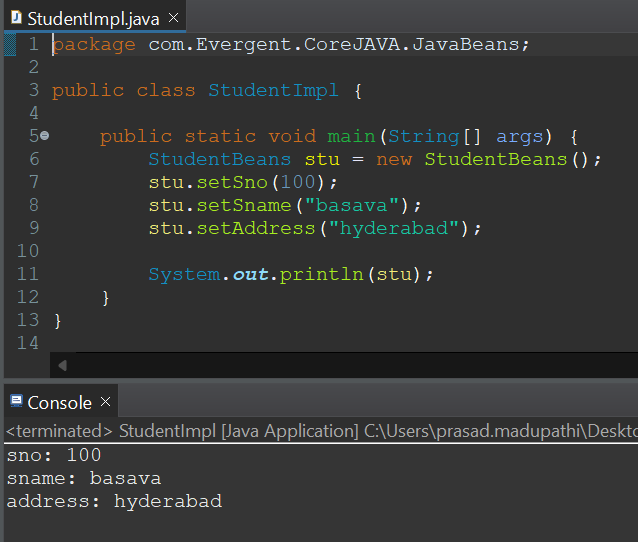




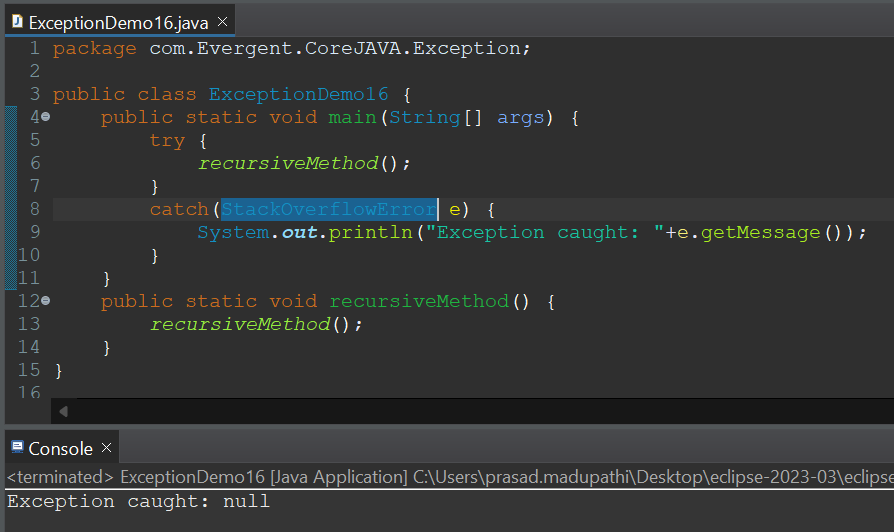


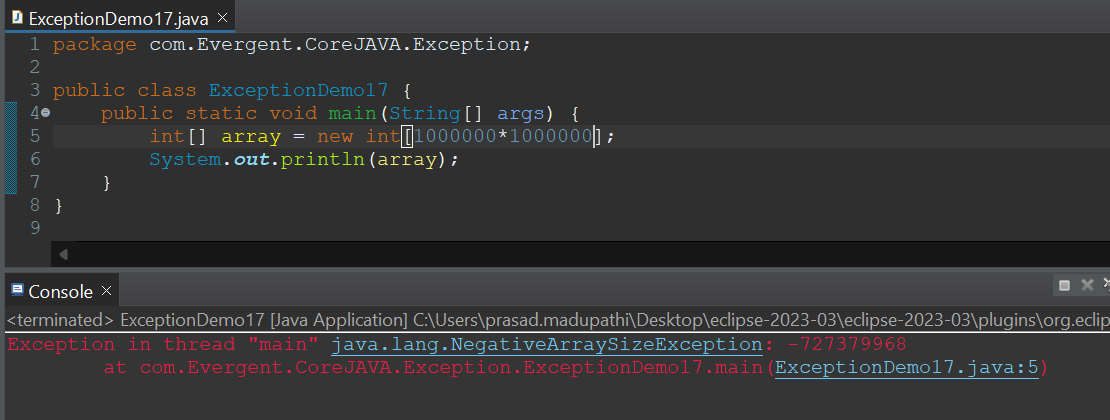






**Day 13: 22-Aug-2024:**



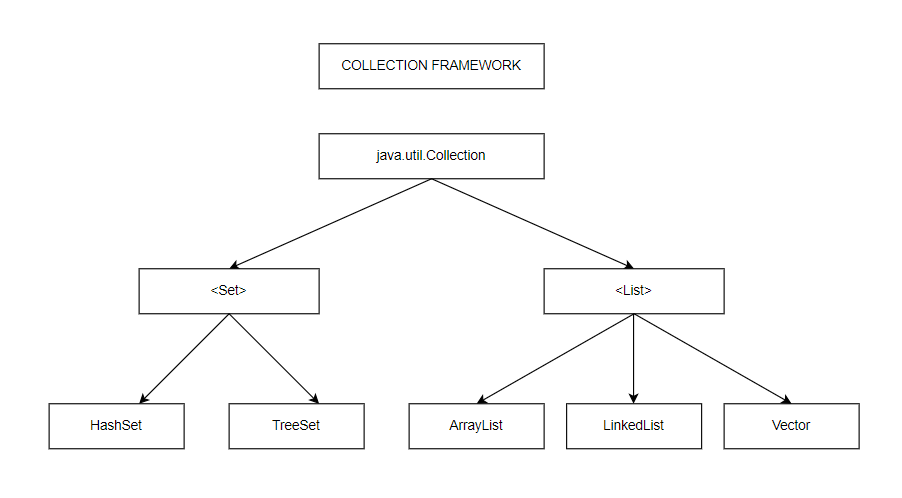


**Git and GitHub:**

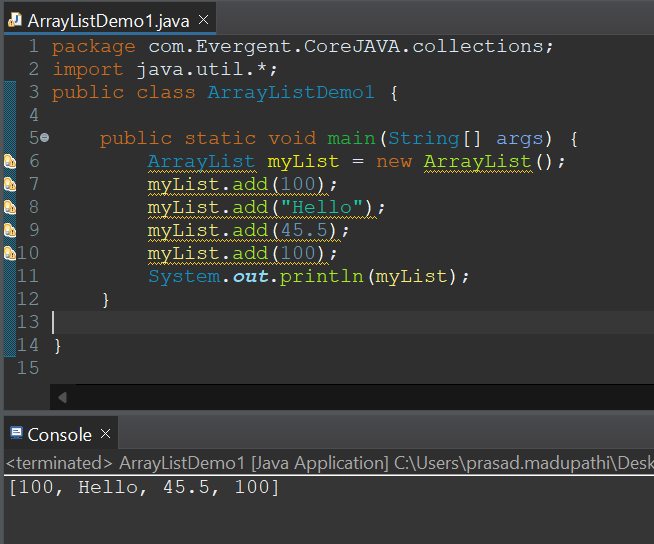
Git Commands:

1. Git init
2. Git status
3. Git add .
4. Git commit -m “Uploaded”
5. Git branch
6. Git remote add origin <https://github.com/Basavaprasad2001/CoreJAVA-Training.git>
7. Git push --force origin master

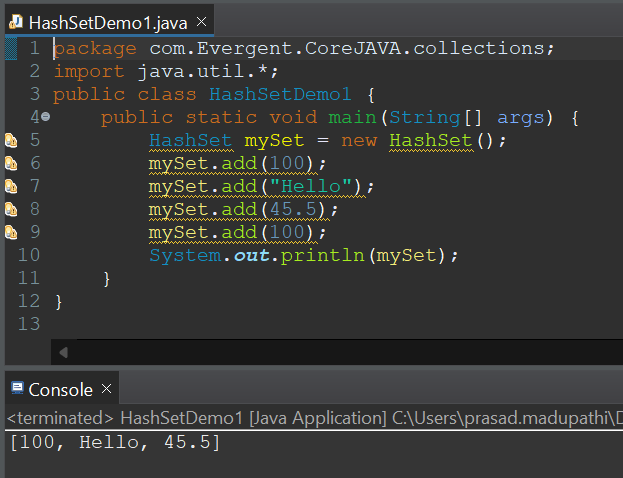
**COLLECTION FRAMEWORK**

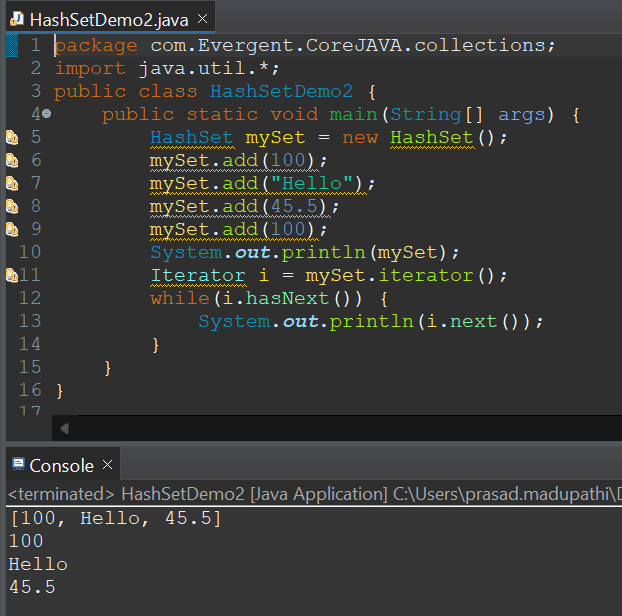


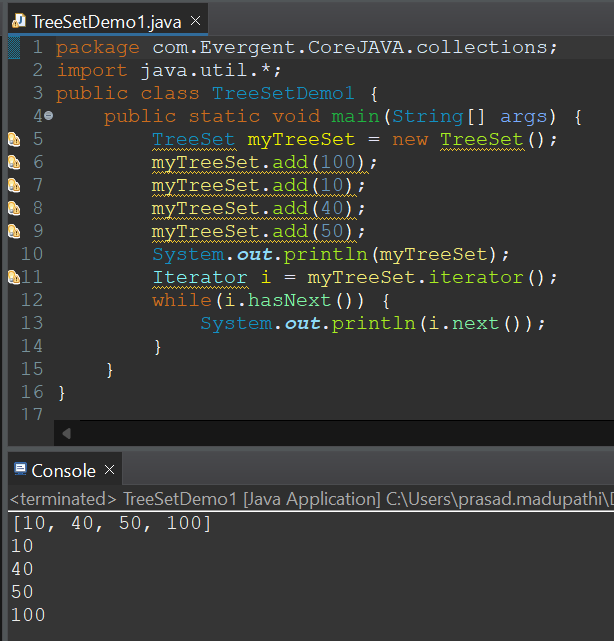
1. Dynamic size growth
2. Stores different data types
3. Contains methods
4. **Set.**
   1. Set items are stored randomly
   2. It doesn’t allow duplicates
5. **List.**
   1. List items are stored orderly
   2. It will allow duplicates







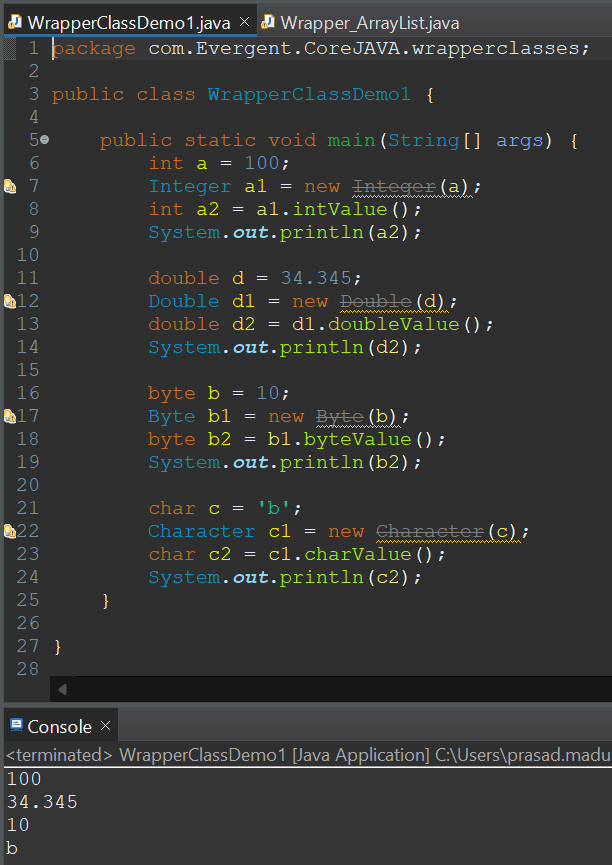


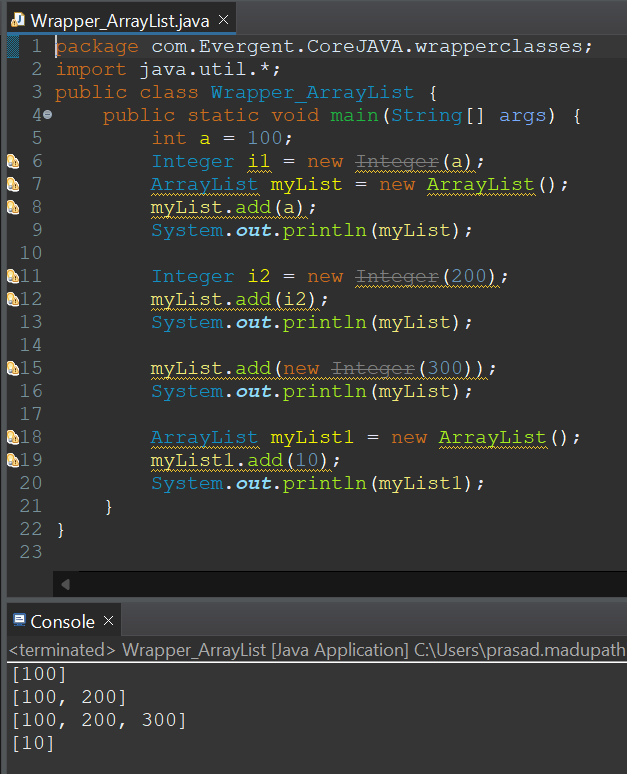


**Day 14: 23-Aug-2024:**

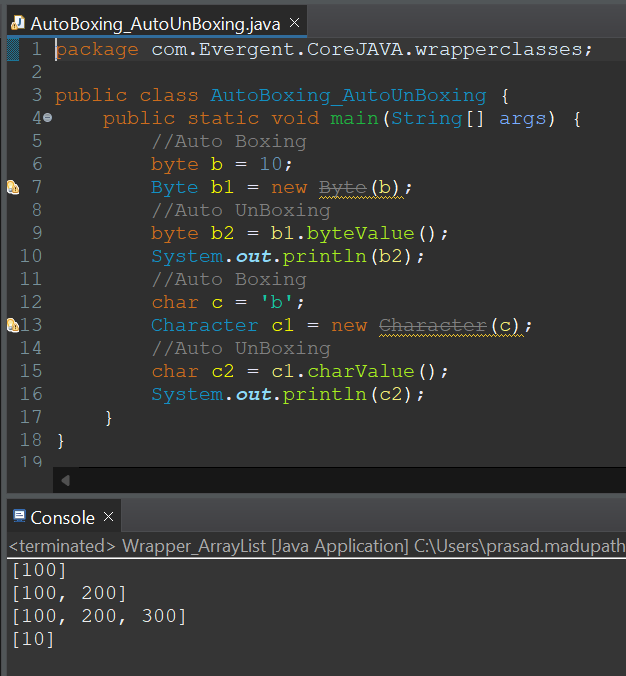
1. We can declare try and catch in loops also
2. If we extend RuntimeException the we don’t need throws
3. If we extend Exception then we must need throws

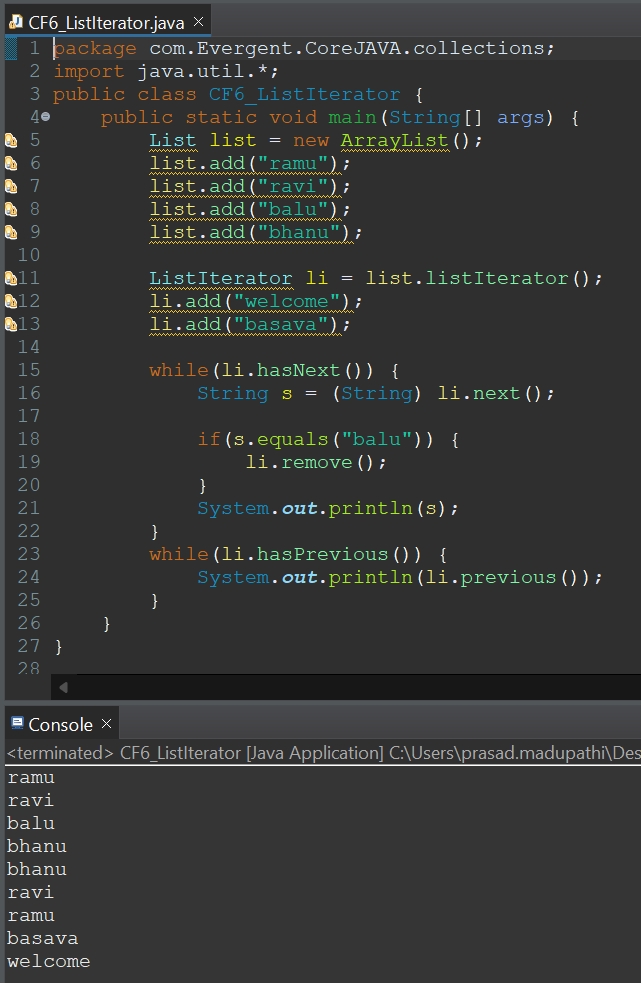
**Wrapper classes:**

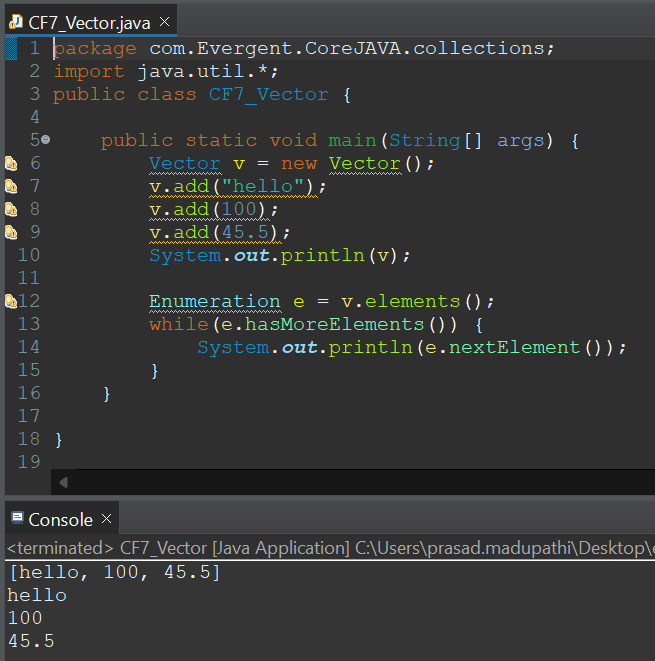


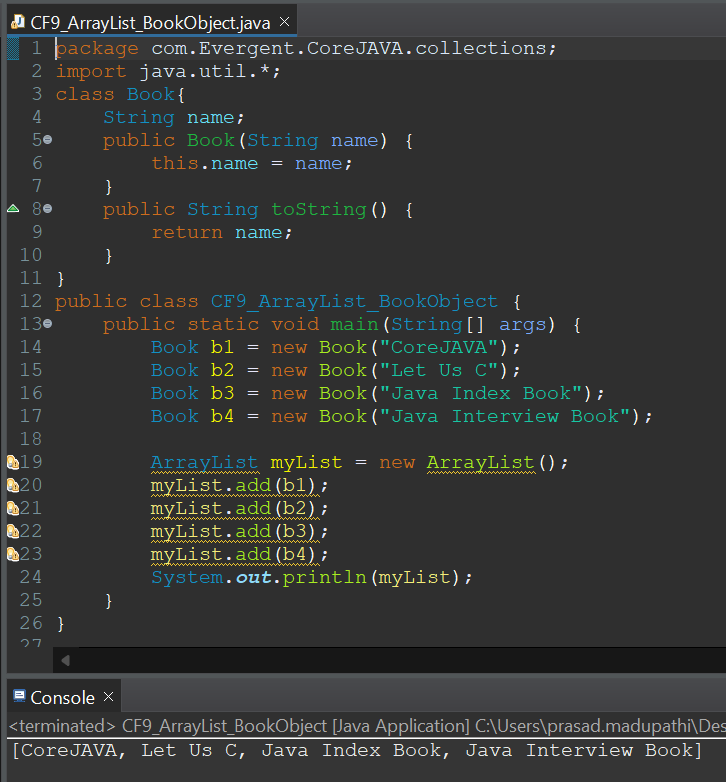


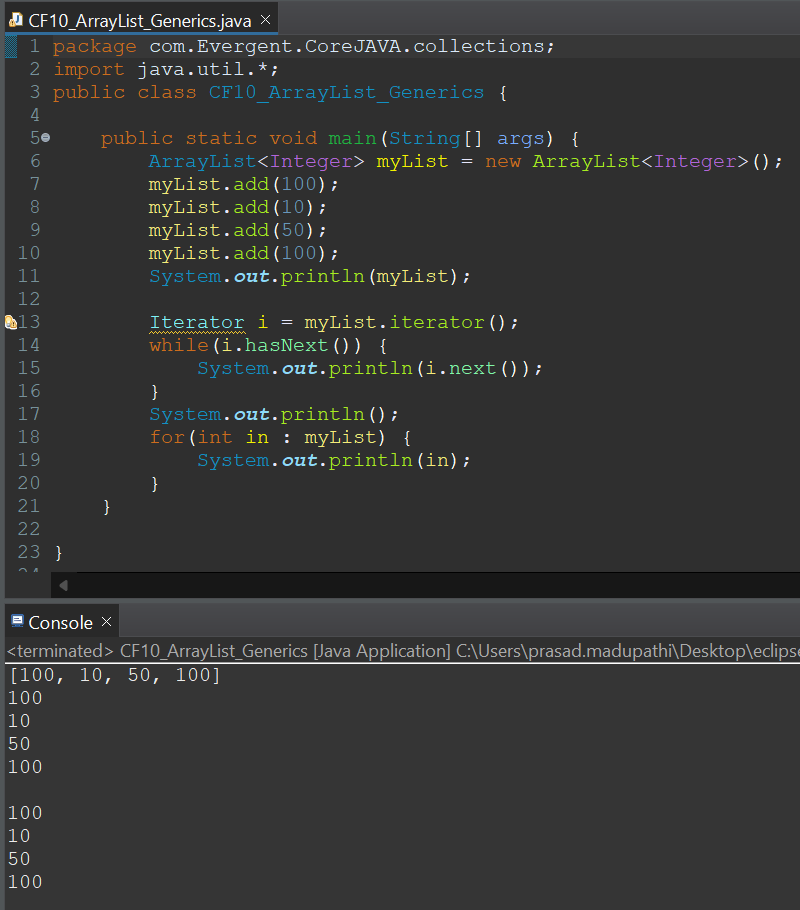
**AutoBoxing and AutoUnBoxing:**

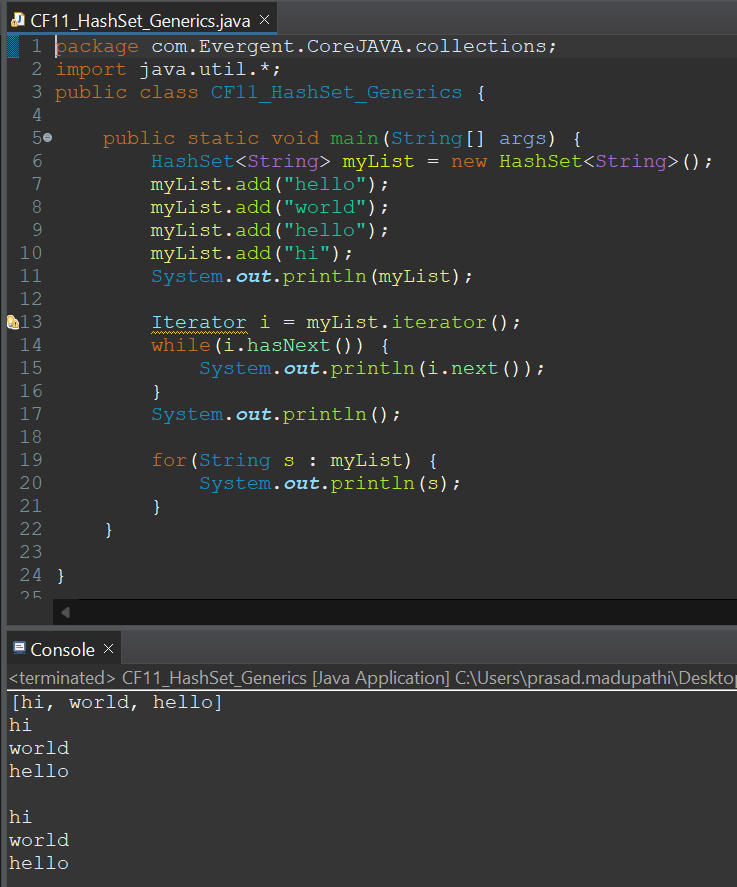
****

****

****

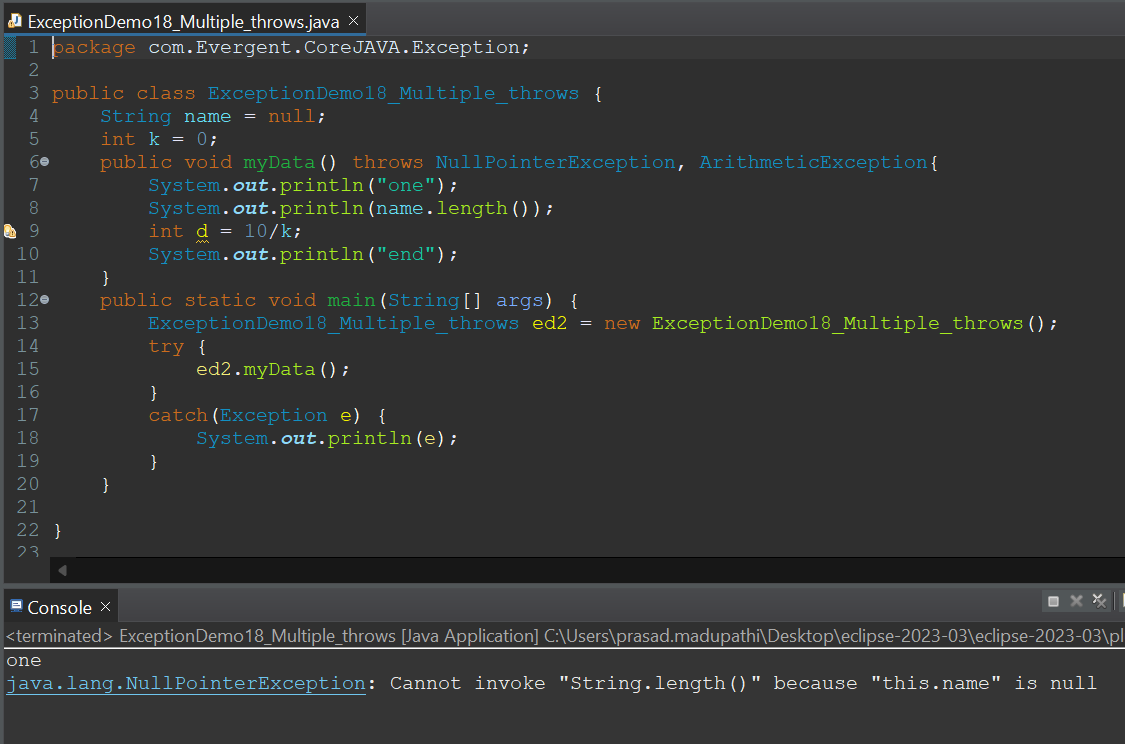
****

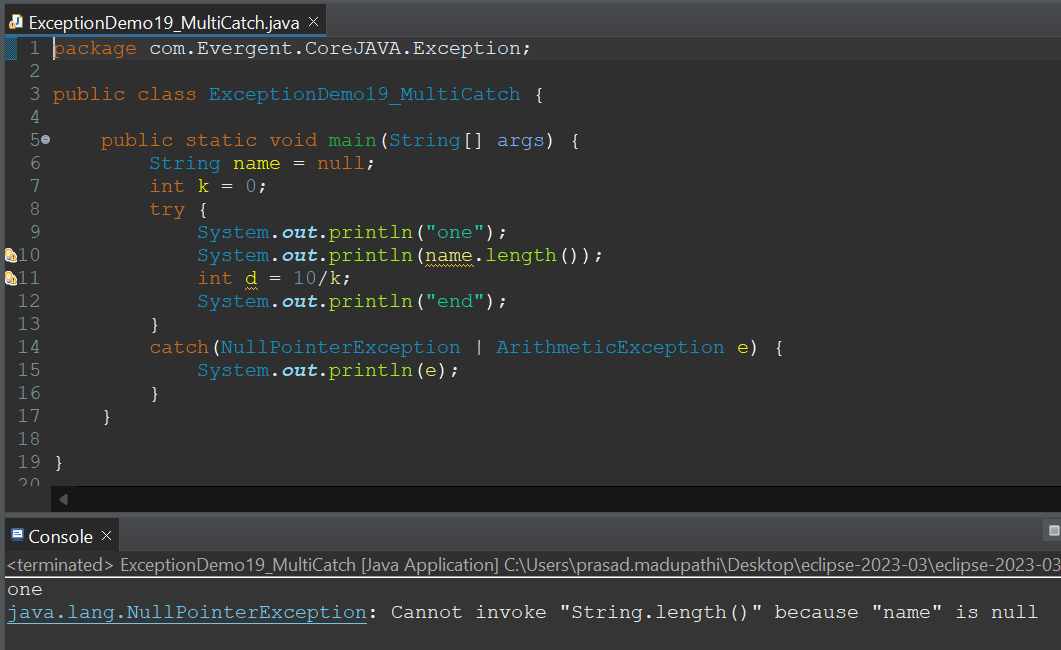
****

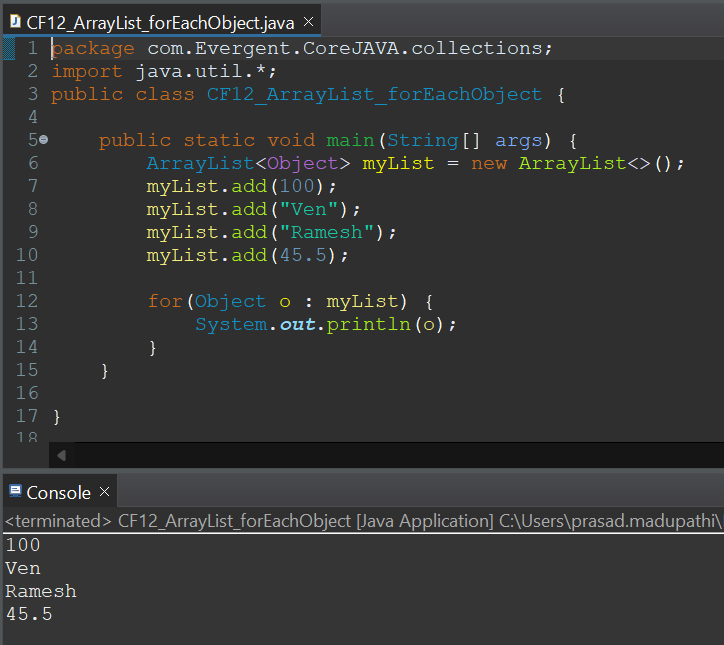
****

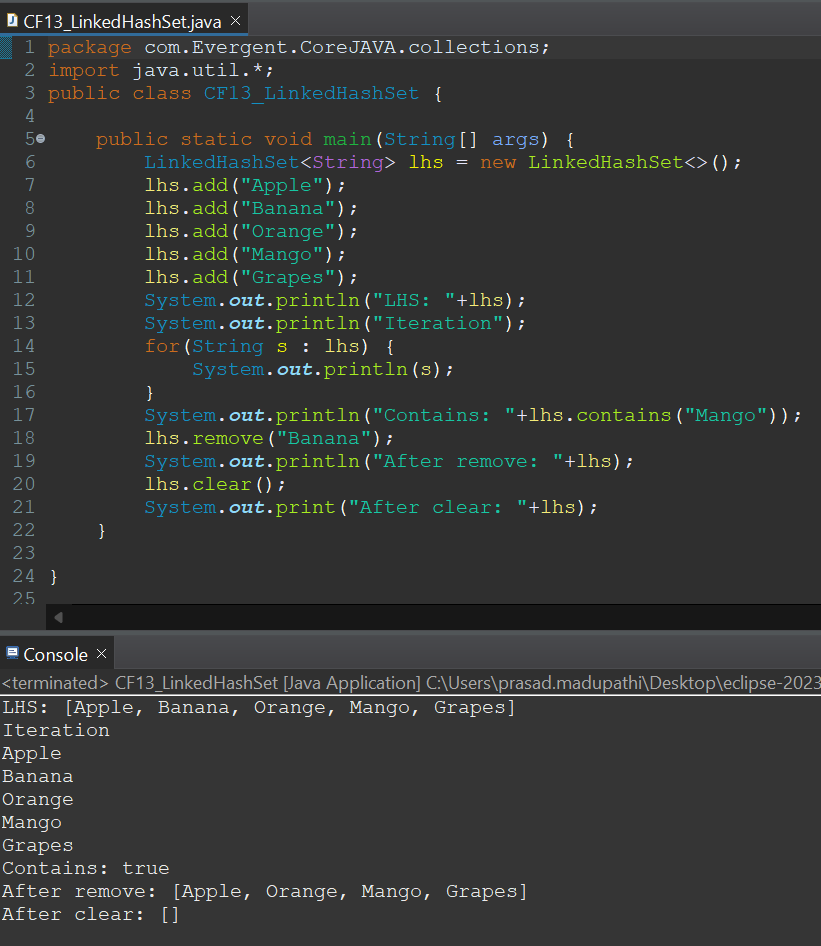
**Day 15: 26-Aug-2024:**

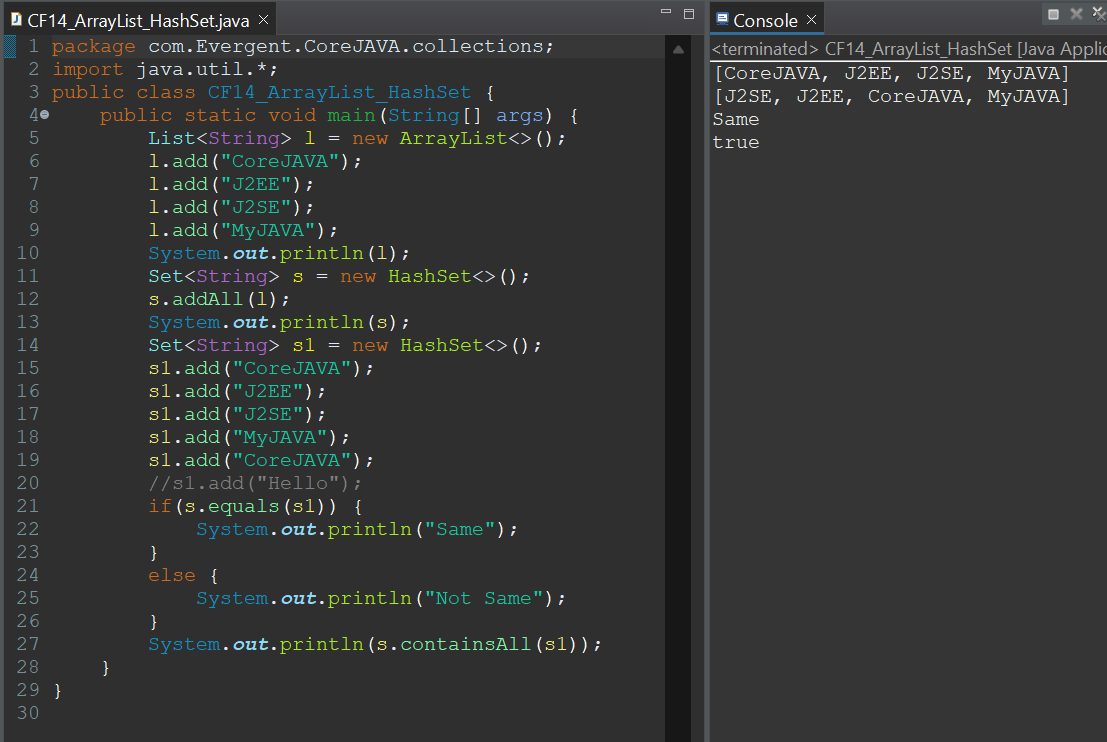
Exception Handling:



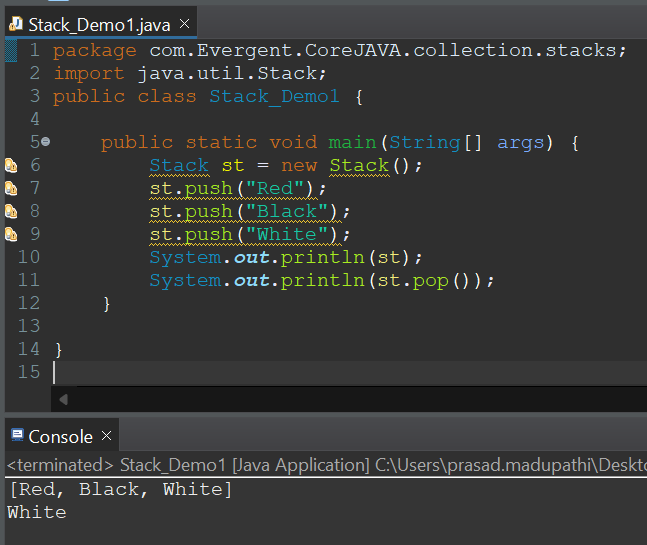


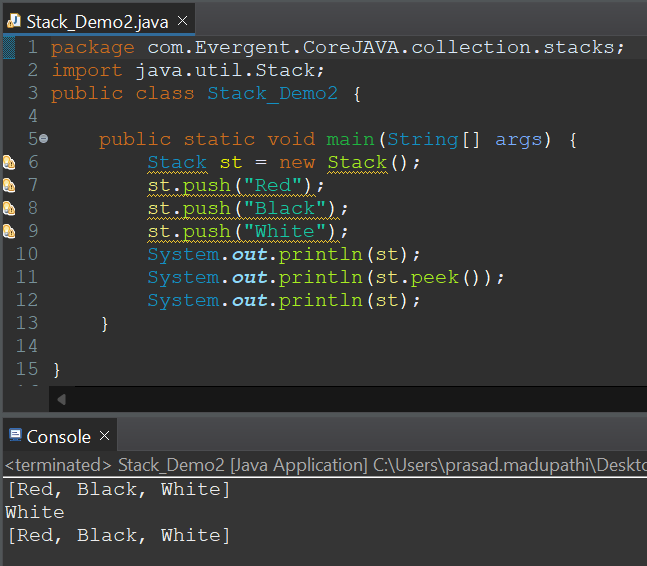


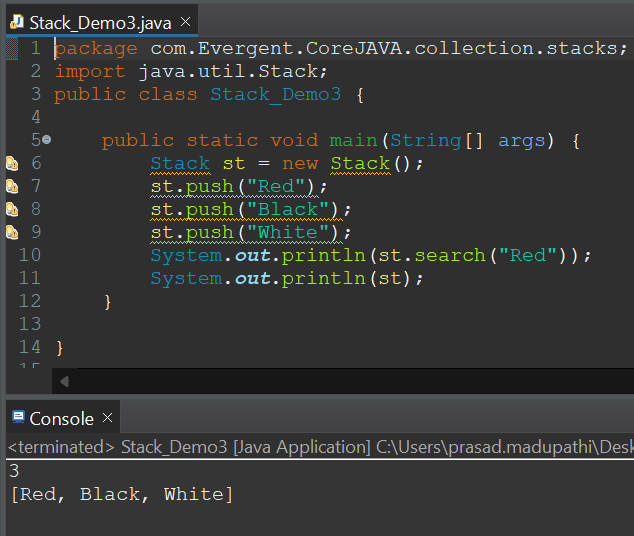
****

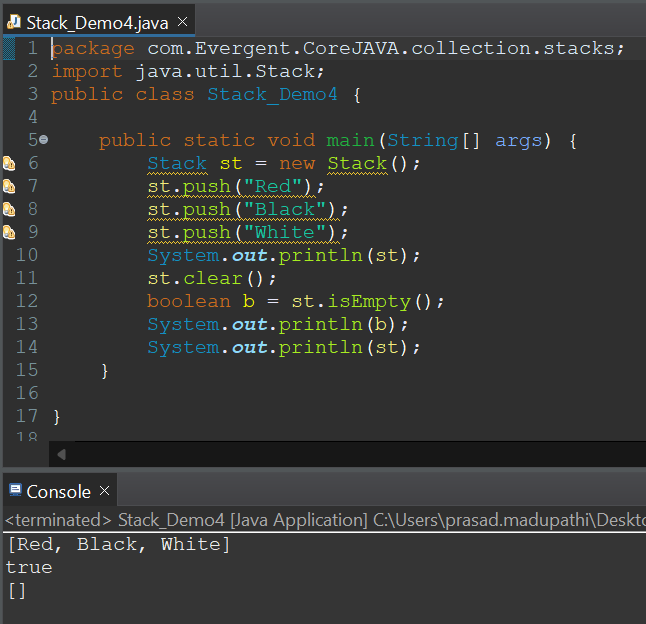
****

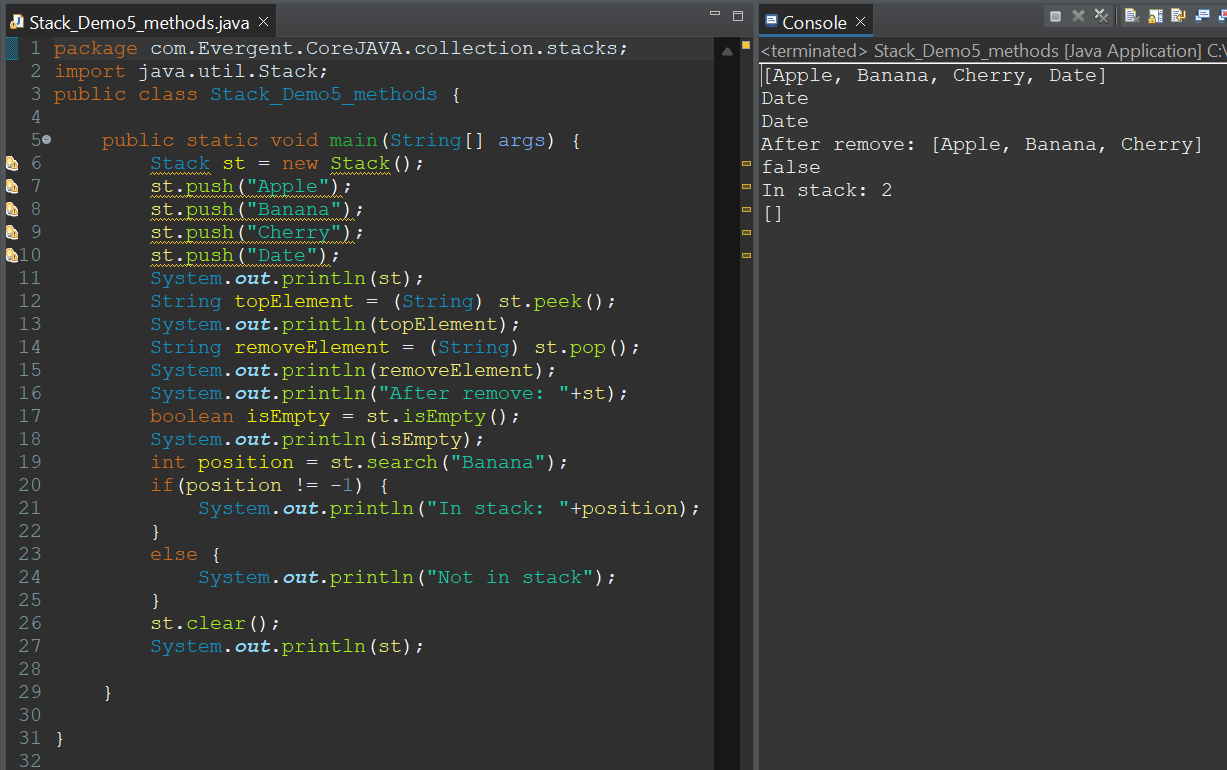
**Stack:**

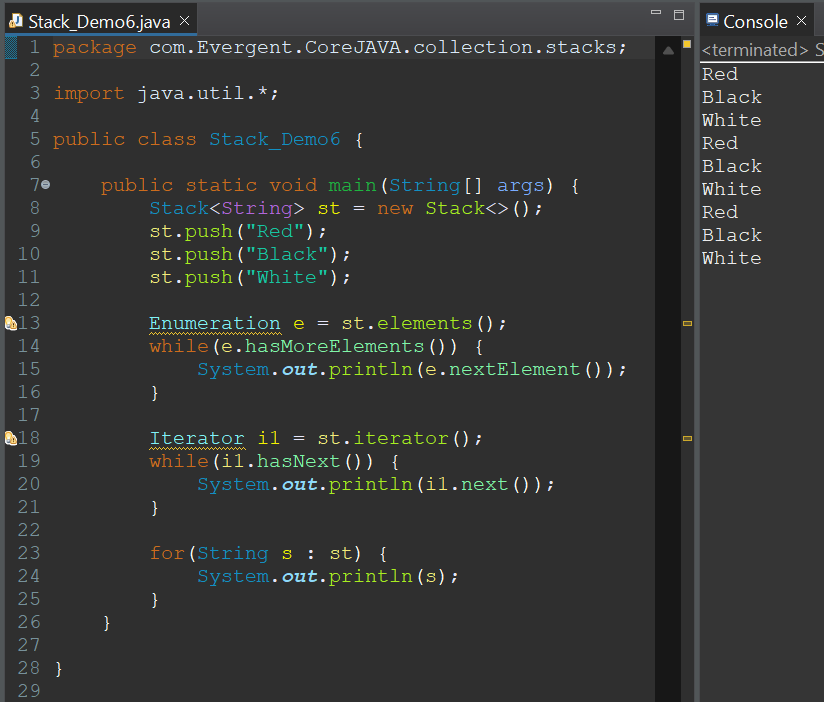
****

****

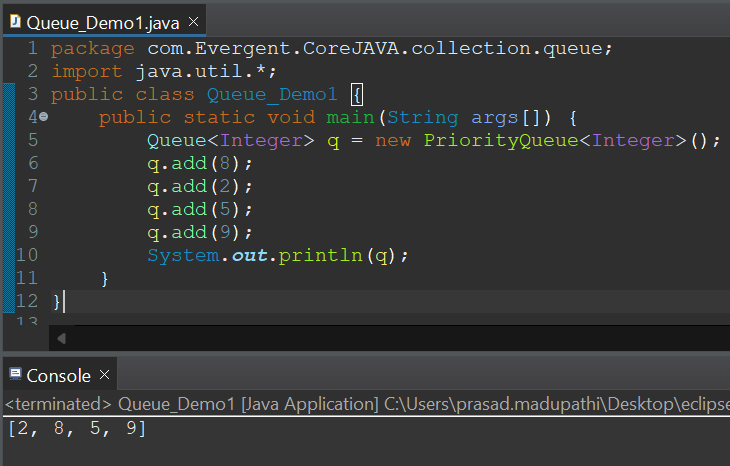
****

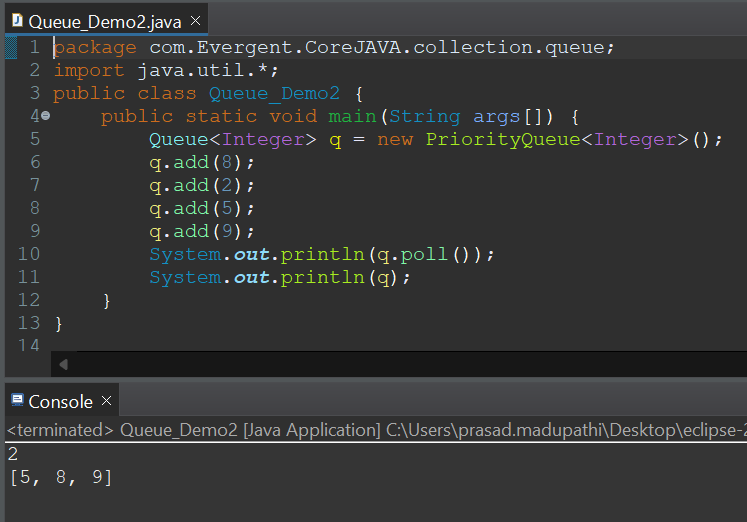
****

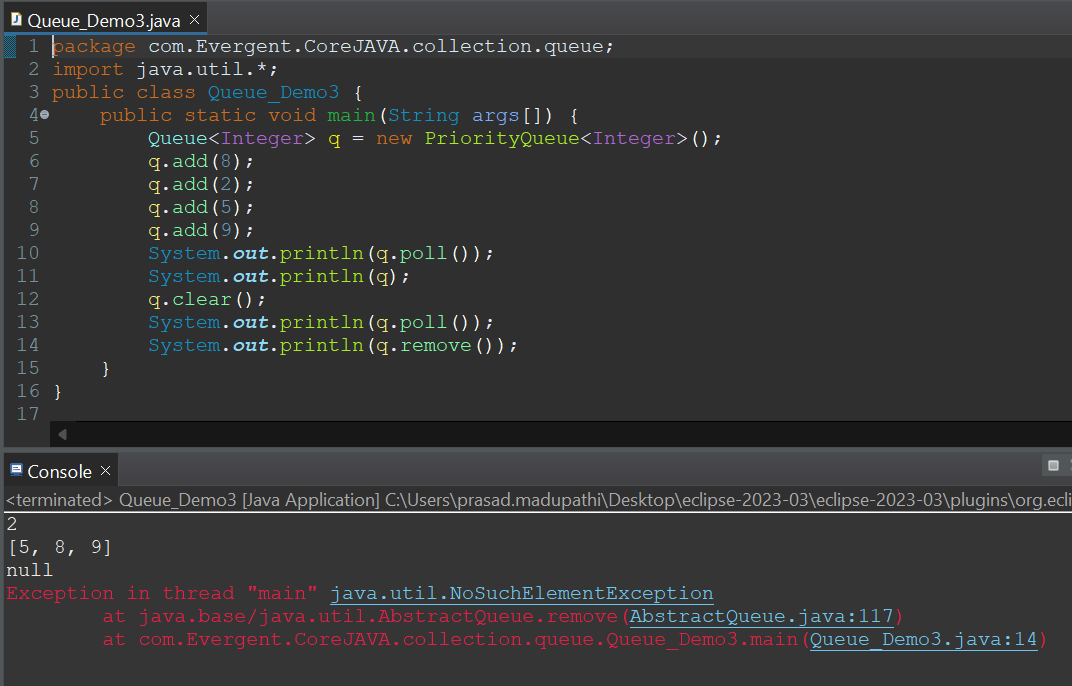
****

****

**Queue:**

****

****

****

****

**Day 16: 27-Aug-2024:**

