**Core Java Index :**

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

1. Language and Applications

2. Java Features

a. Why Java is Platform Independent

b. Oops

c. Exception Handling

d. Multi-Threading

e. Web applications

f. Open source

g. Security

h. Supports Networking

i. Memory Management

3. JDK, JRE, JVM

4. Conditional Statements ,Operators

5. Basic Java Programming

a. Largest of Two numbers

b. Largest of Three numbers

c. Swapping of two numbers with and without temp variable

6. Packages

7. 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

a. Calculations

b. PersonClass

c. MethodFlow

d. System.Out.Println()

3. Inheritance

4. Polymorphism

a. MethodOverloading

b. MethodOverriding

5. Abstract

6. IS-A Relation (inheritance )

7. HAS-A Relation(Object creation)

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

Morning Session:

* Contructors
* Concept of constructors which goes as:

1. The constructor name and class name should be same.
2. There are two types of constructors such as default and parameterized.
3. We can access constructors while creation of object.
4. Constructors are mainly for initializing.
5. Constructors doesn’t have any return type not even void.if there is void return type then the compiler will not consider as constructor but as method.
6. Every class needs at least one default constructor.
7. This,super keywords.
8. Constructors are always overloaded.
9. This is a keyword always pointing to the instance variables.

* Programs on constructor.

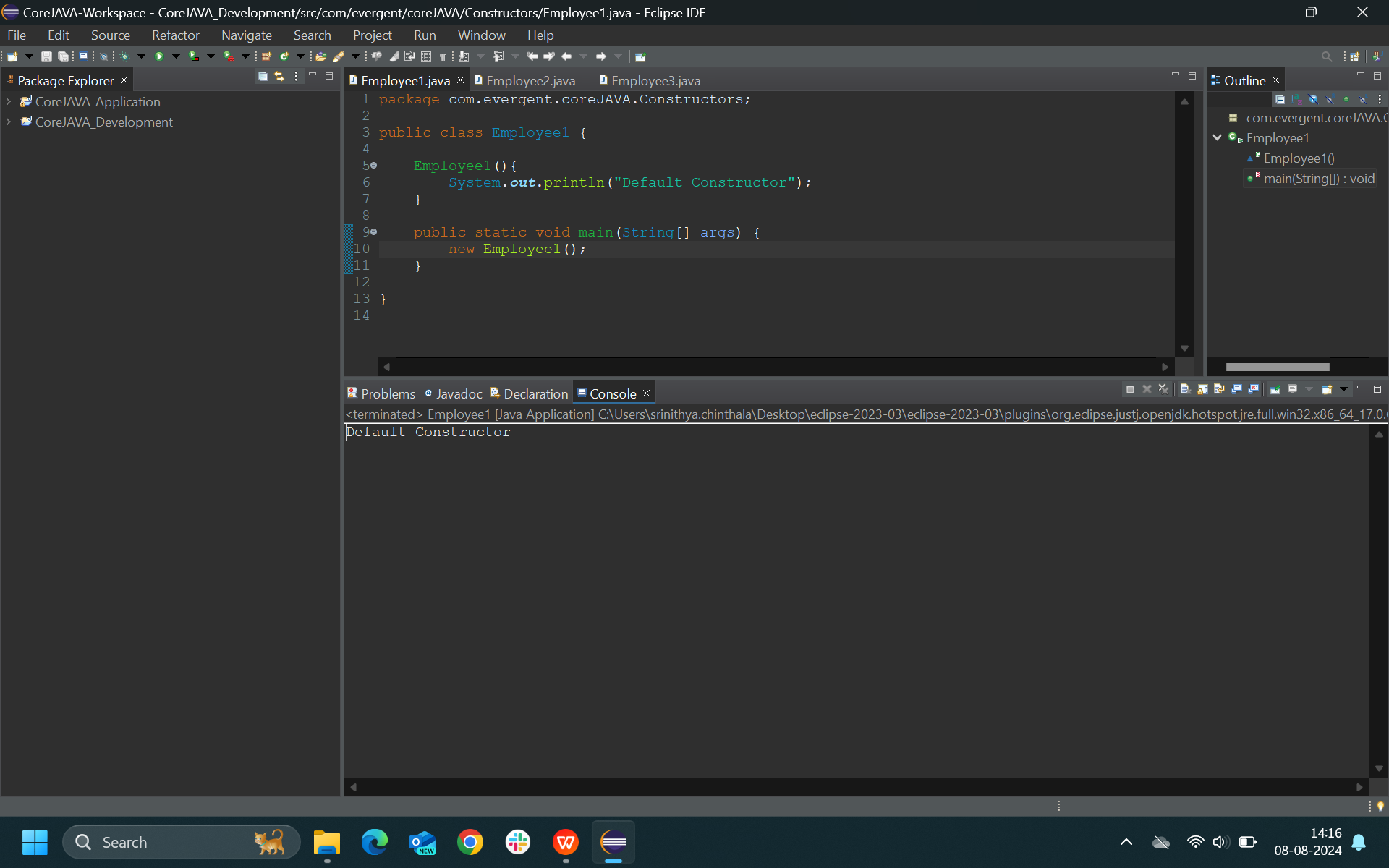
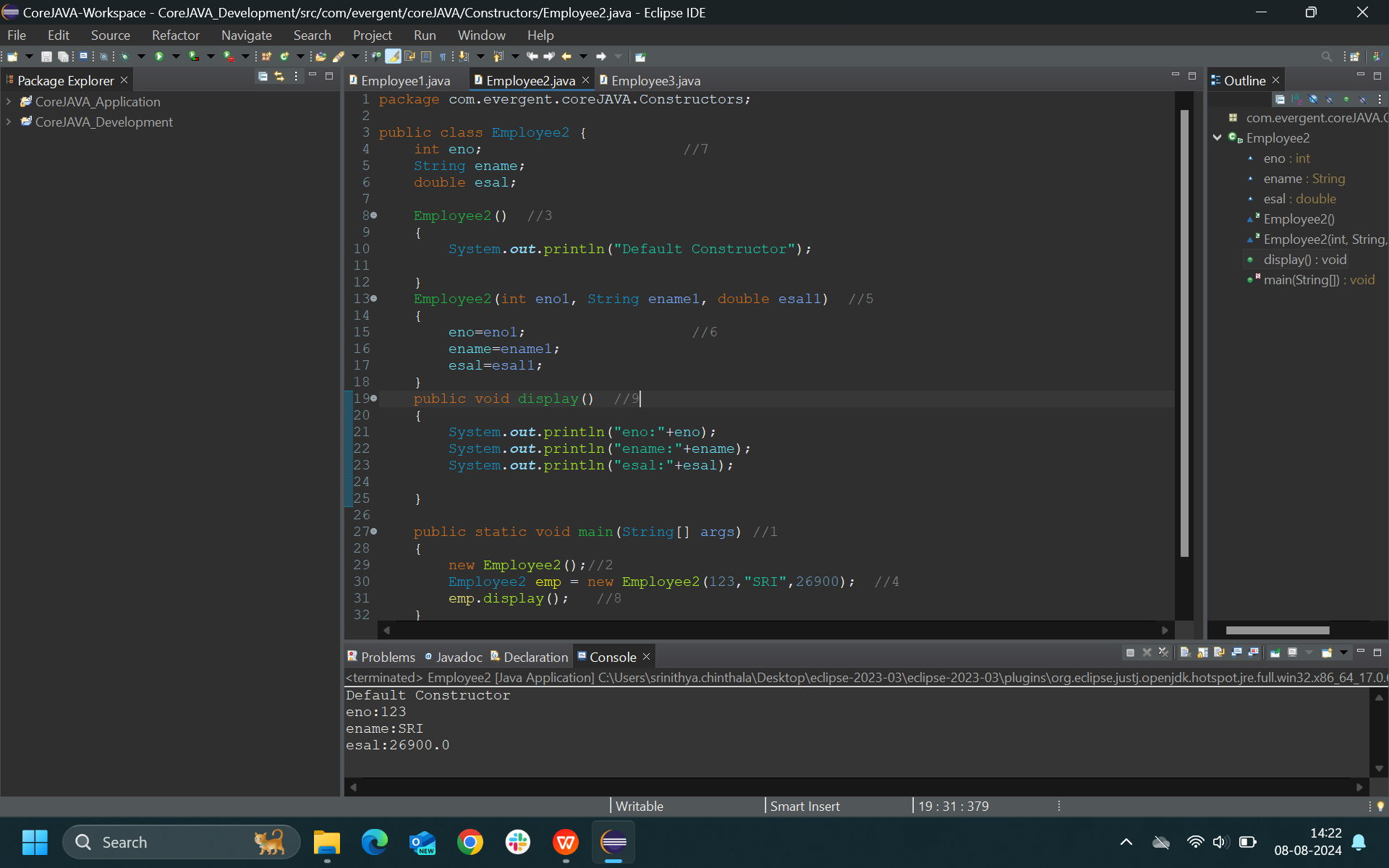
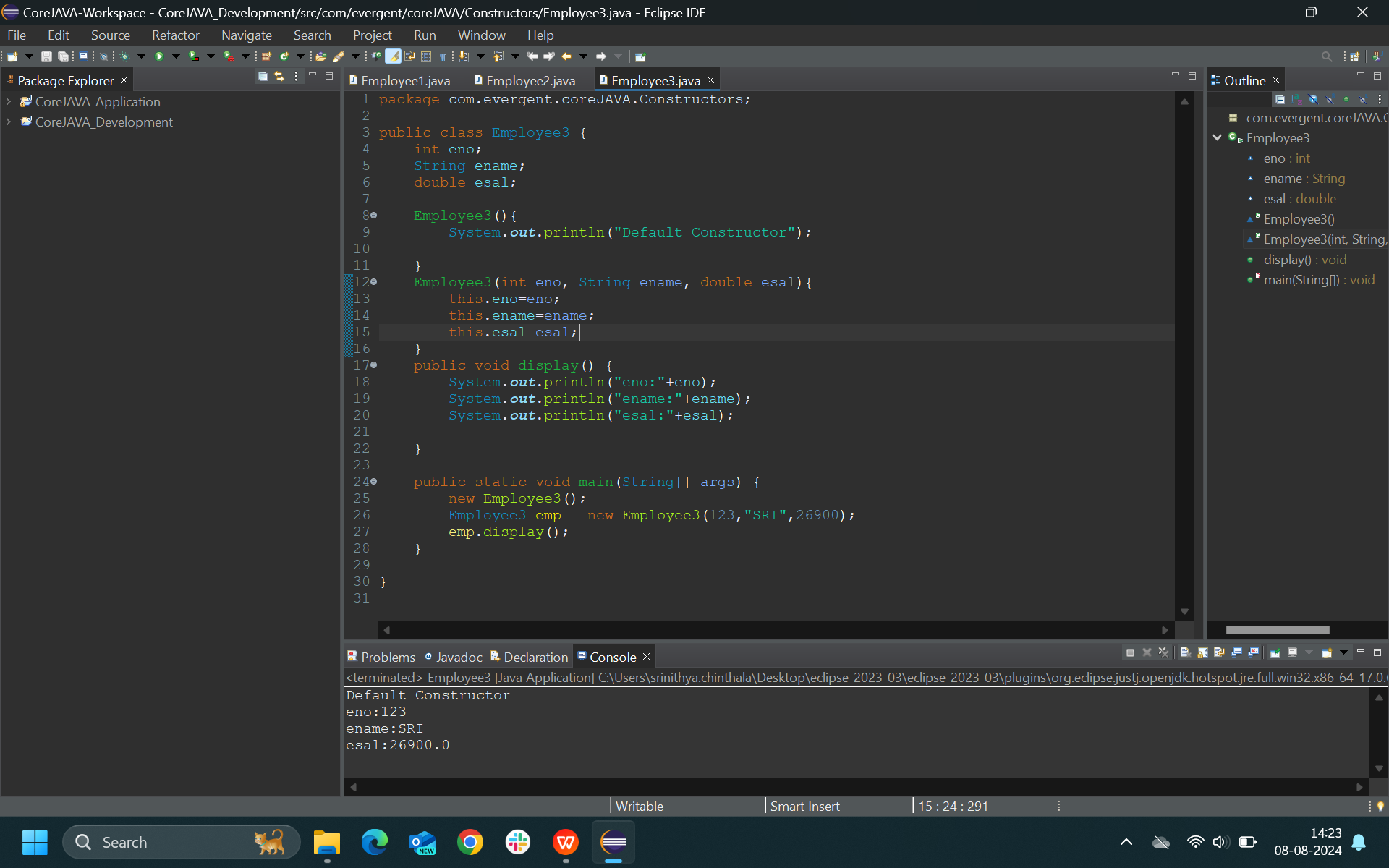
Afternoon Session:

* Super keyword.

Given Task:

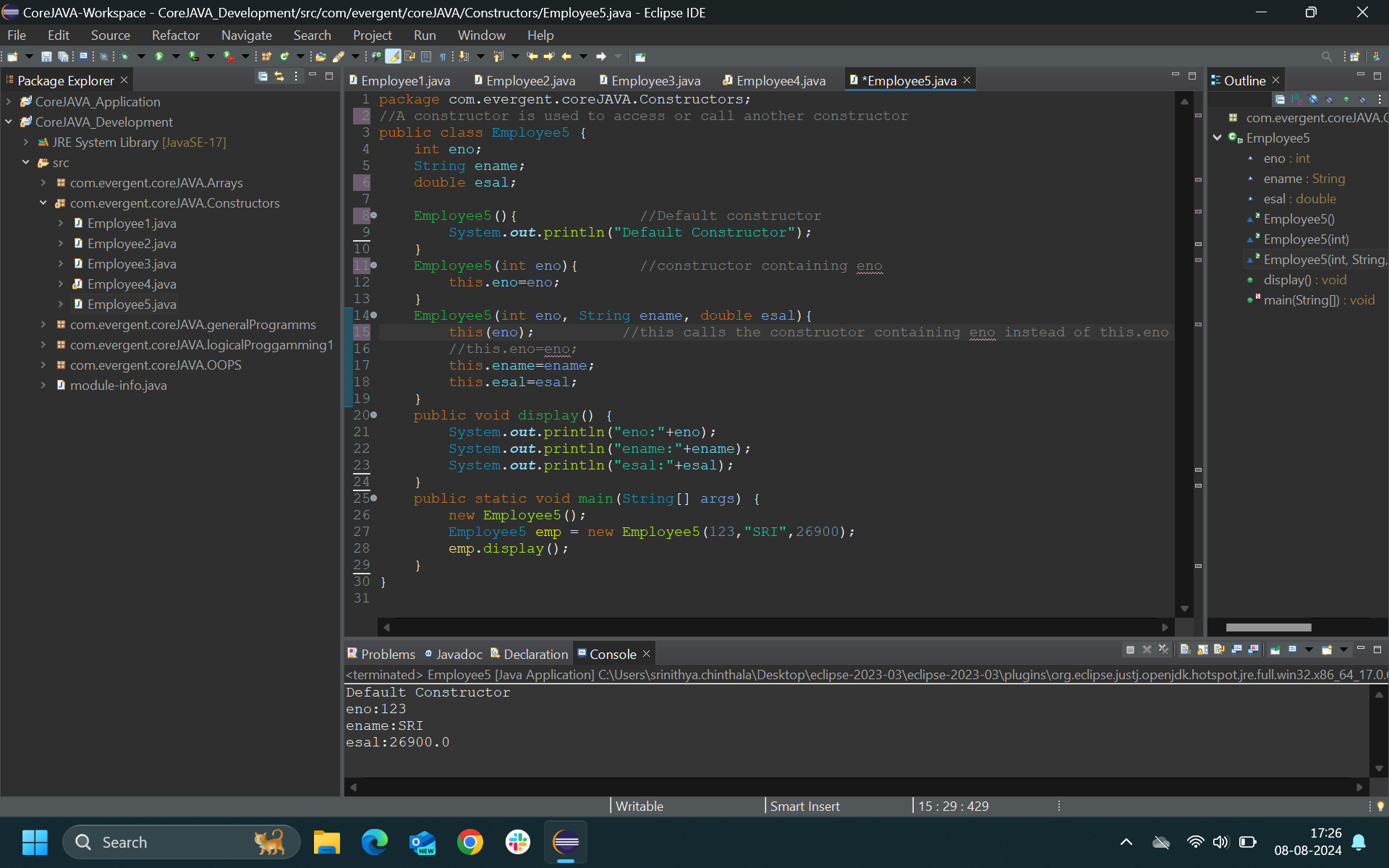
* To complete the given programs on default constructor and parameterized constructors:

1. Employee1(defalut constructor)
2. Employee2(parameterized constructor)
3. Employee3(parameterized constructor using “this” key word)

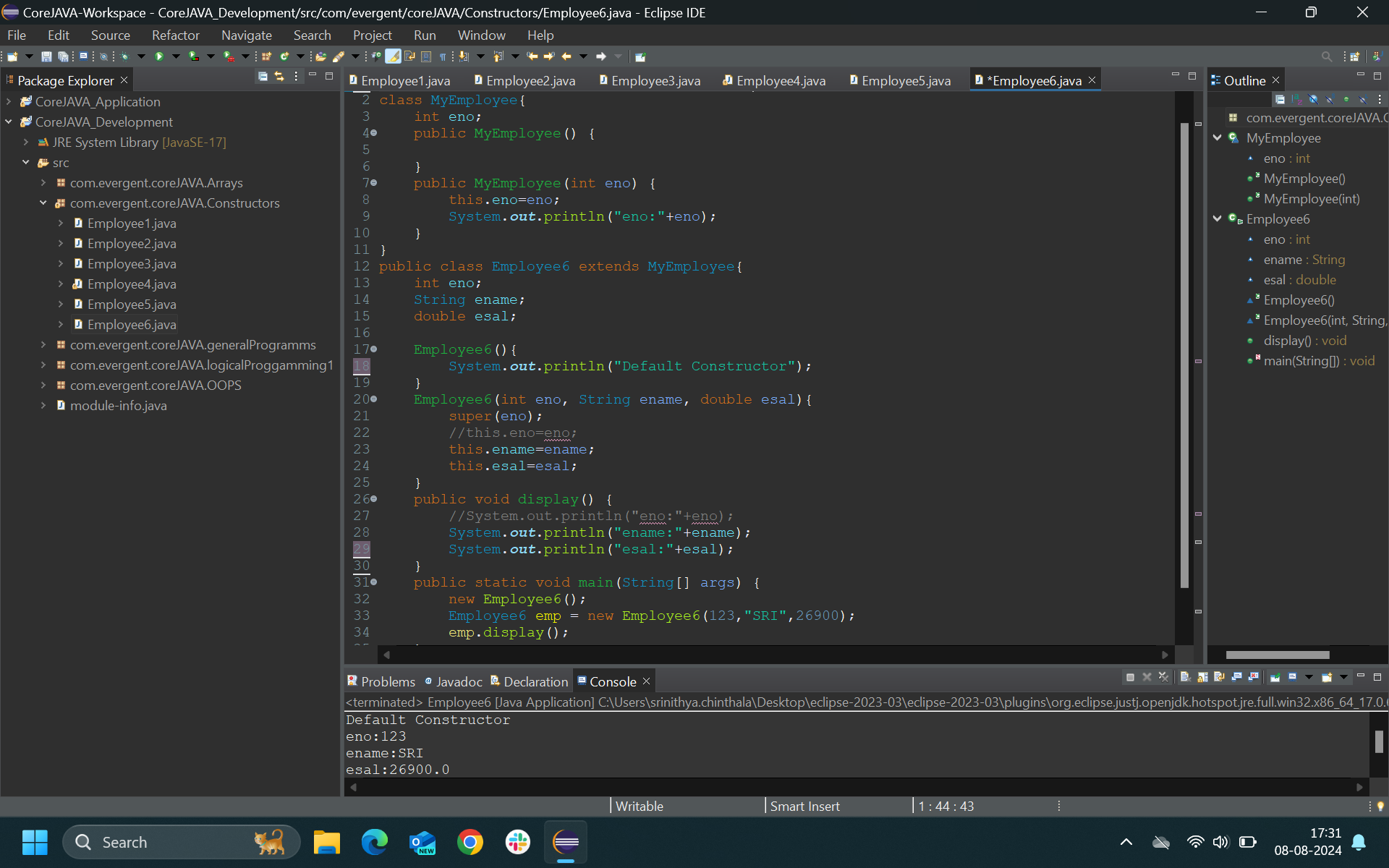
* ****Employee1*:*
* ****Employee2:
* **Employee3:
* Employee4:

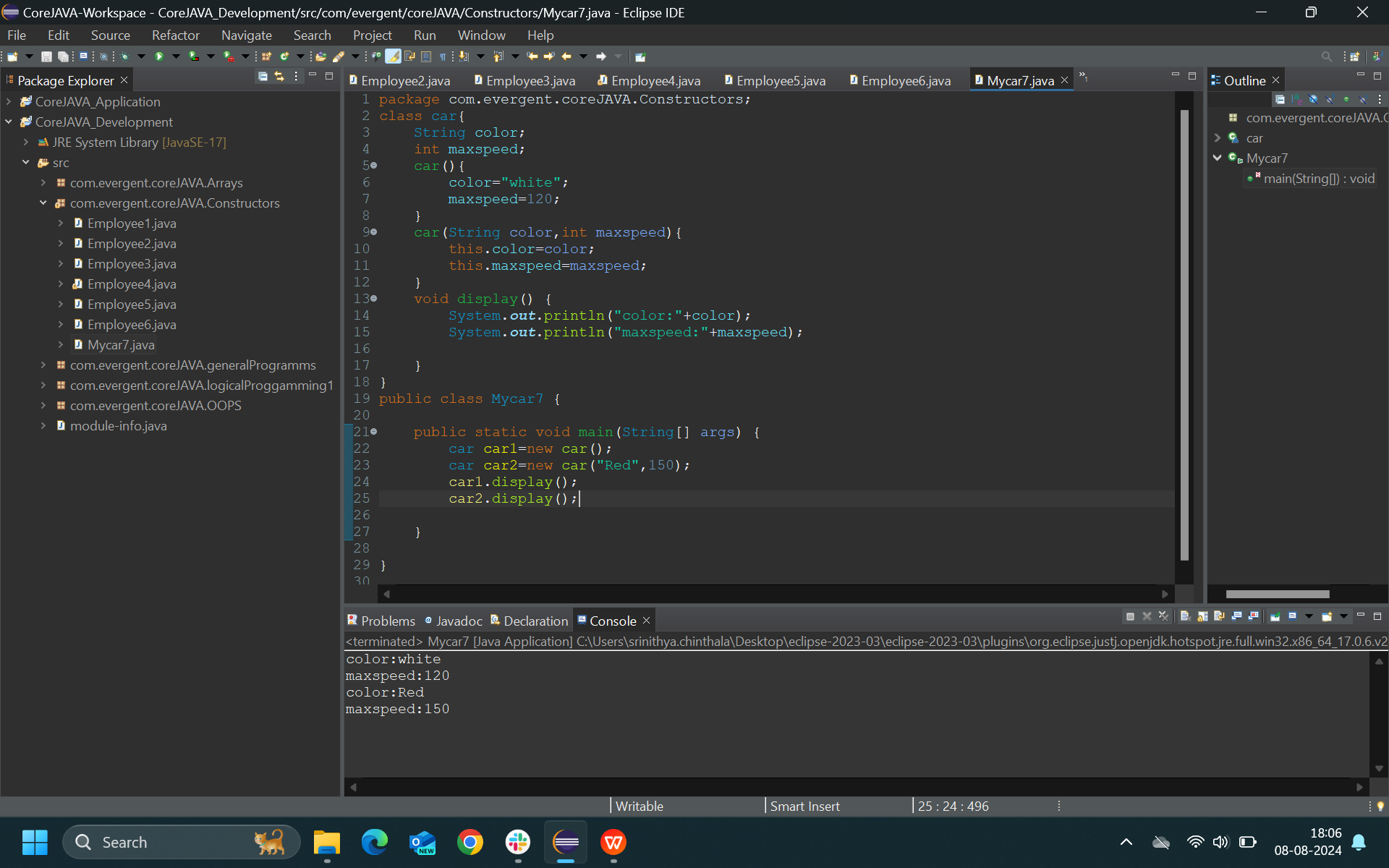
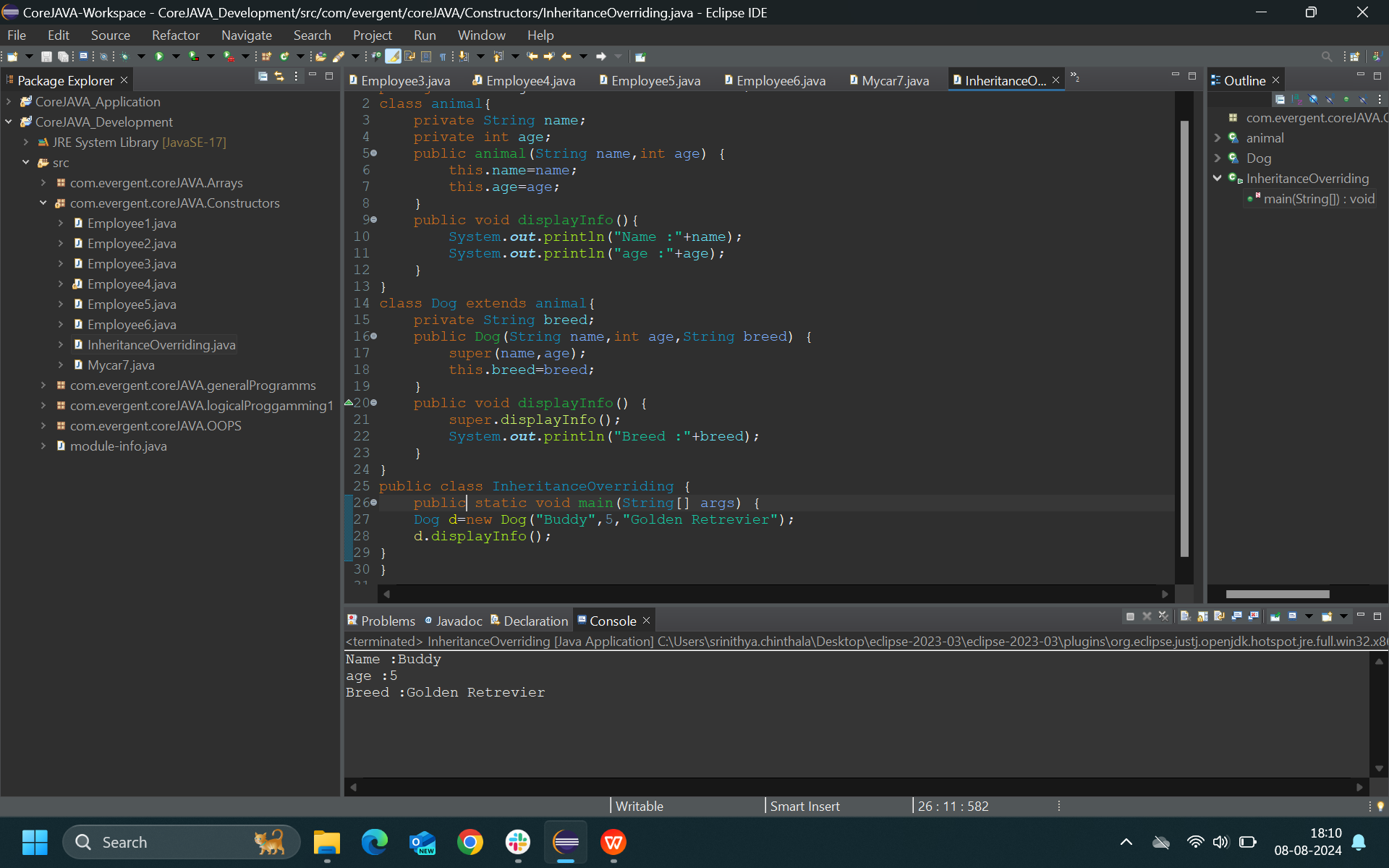


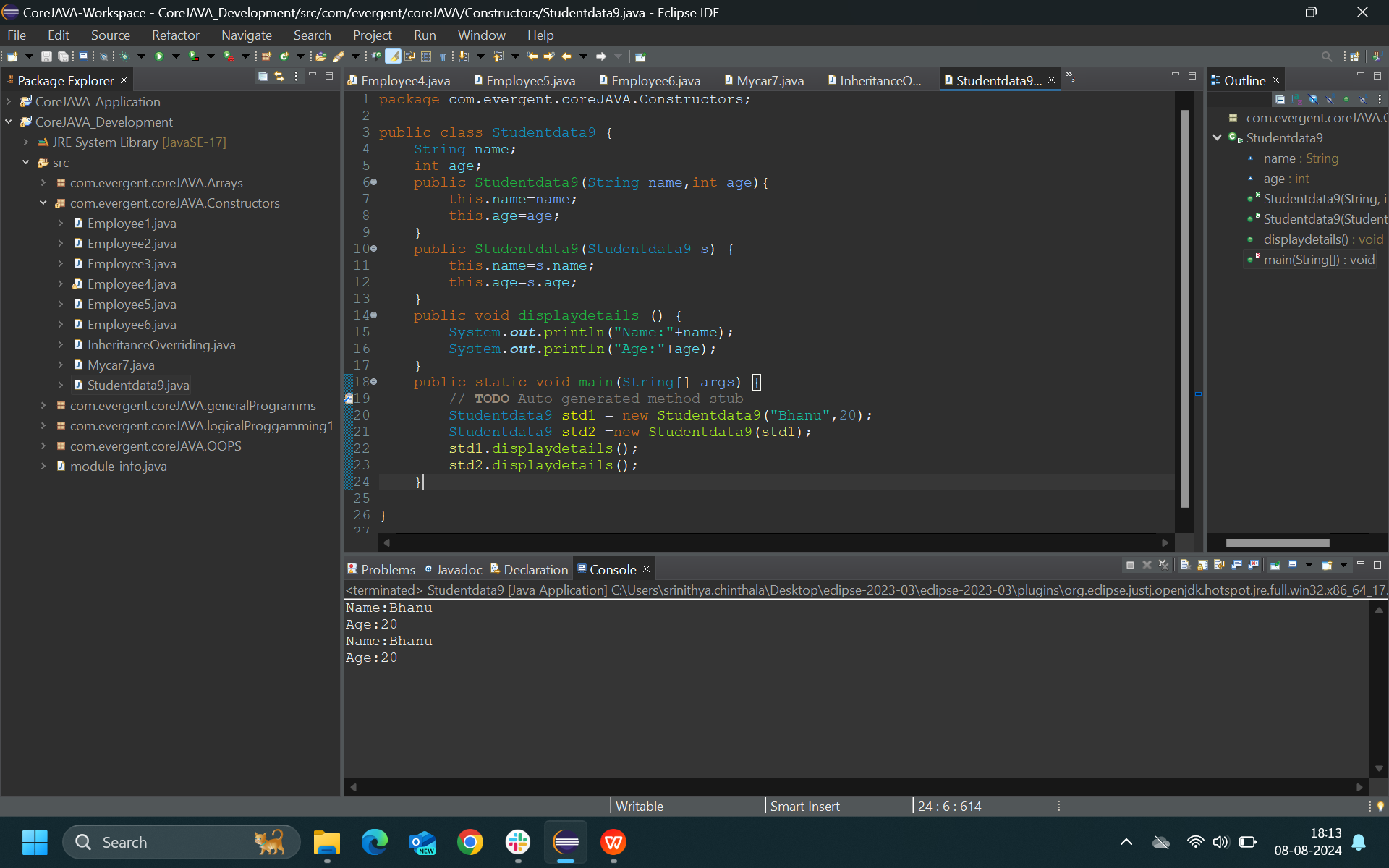
* Employee5:



* Employee6:



* MyCar7:
* Inheritance Overriding:
* Studentdata9:



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

Morning Session:

* Static Keyword

1. Static is a keyword.
2. We can declare static as variable & method.
3. We can access static variable & method directly as classname.method name & classname.variable name.
4. Static methods can access static variables & methods only.
5. Static methods can not access non static variables & methods.
6. Non Static methods can access static variables & methods.
7. Static Block:when ever class is loaded in JVM the static blocks are initiated.

* Final keyword

1. Final is a keyword.
2. We can declare final as variable & method & class.
3. Final variables cannot be modified.
4. Final methods cannot be override.
5. Final class cannot get inherited.

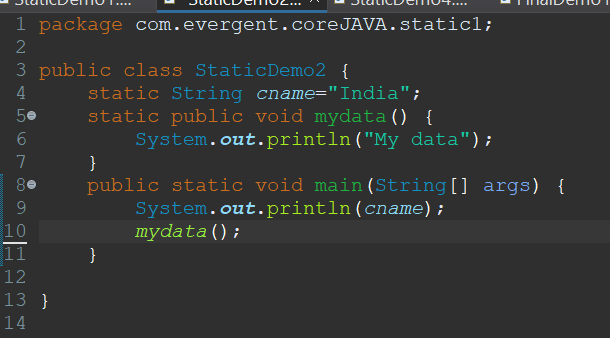
* Programs on Static and final keyword .

Tasks:

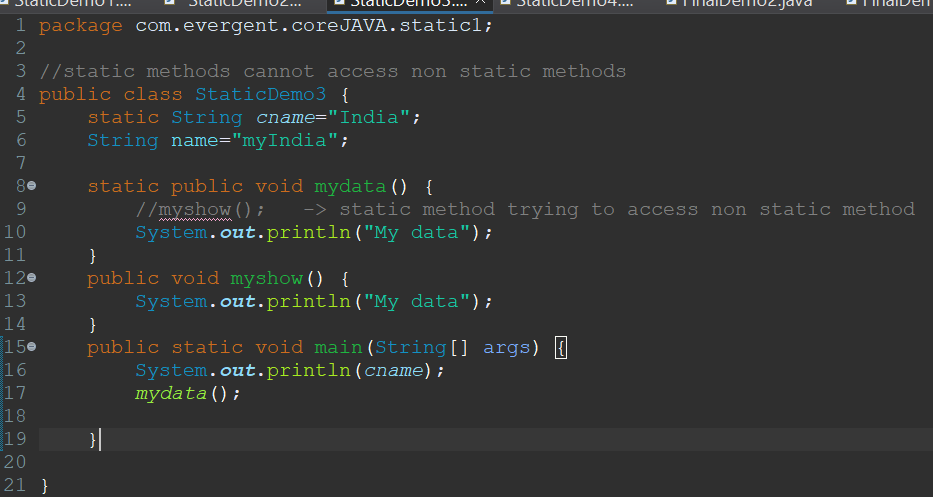
* StaticDemo1*:*



StaticDemo2*:*



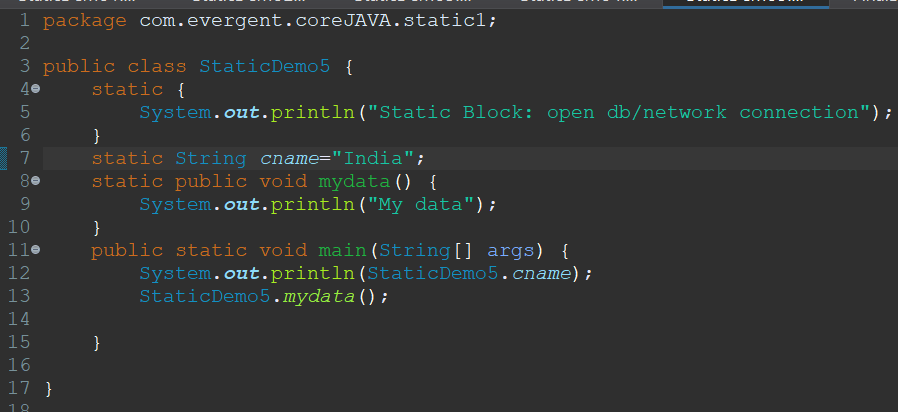
* StaticDemo3:



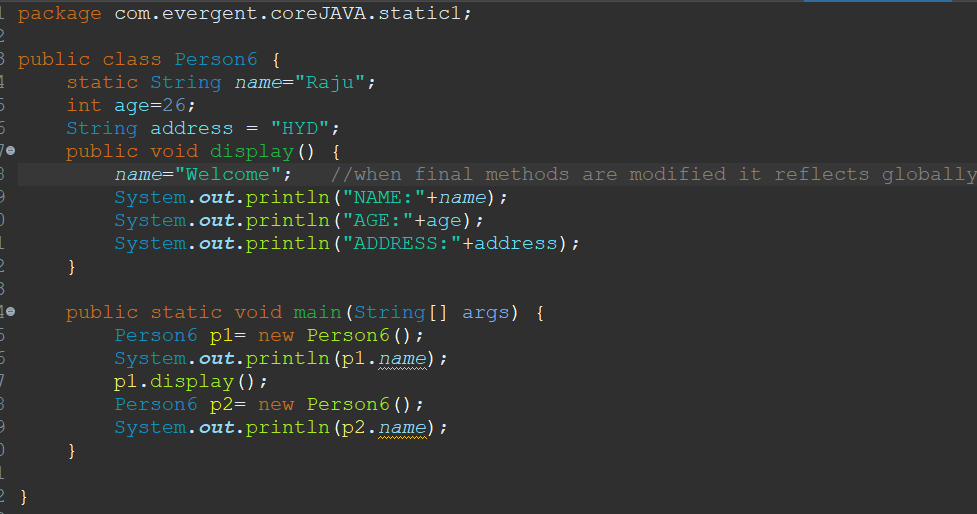
* StaticDemo4:



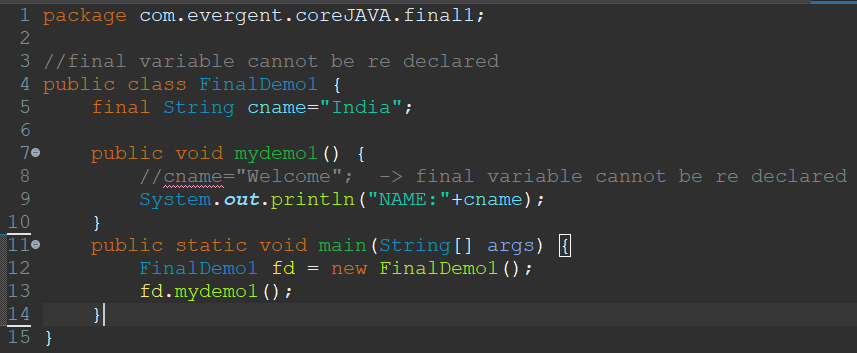
* StaticDemo5:



* Person6:



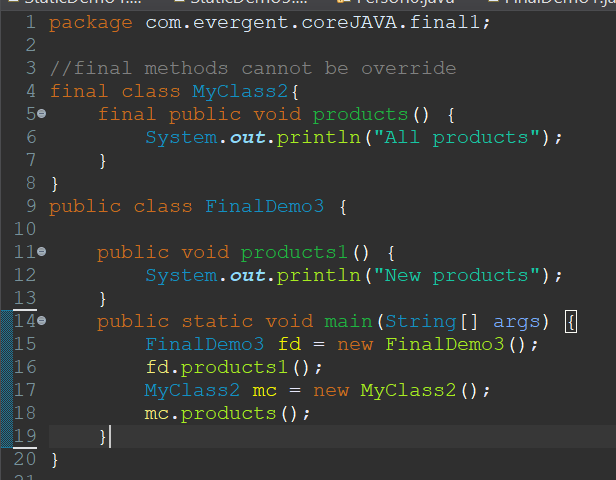
* FinalDemo1:



* FinalDemo2:



* FinalDemo3:



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

**Morning Session**:

* STRING

1. String class
2. String buffer
3. String builder

* String Class

1. String is a final class.
2. String class is immutable : Once we declare any String object it is constant.If we are trying to modify any existing string then the it will create other memory location.Existing object is eligible for garbage collections.
3. String class having methods.
4. All String class methods are non synchronized.

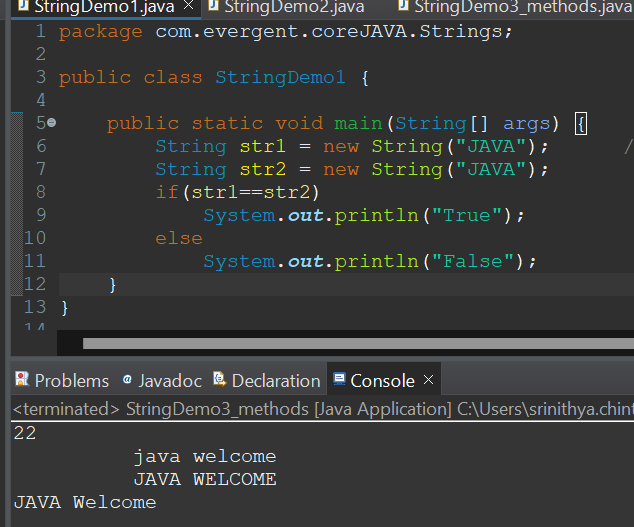
* String Buffer

1. String is a final class.
2. String buffer is mutable.
3. String buffer having methods.
4. All String buffer methods are synchronized.
5. String buffer is not recommended to use in development but is still in API.

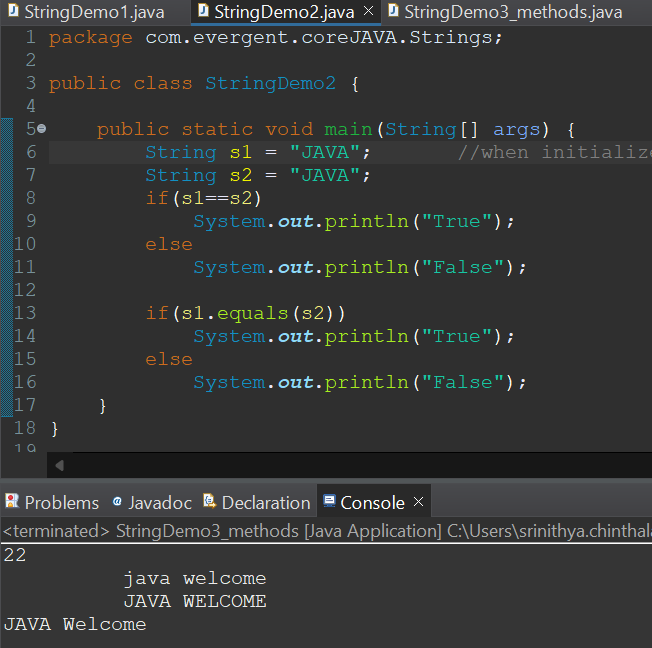
* String Builder

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

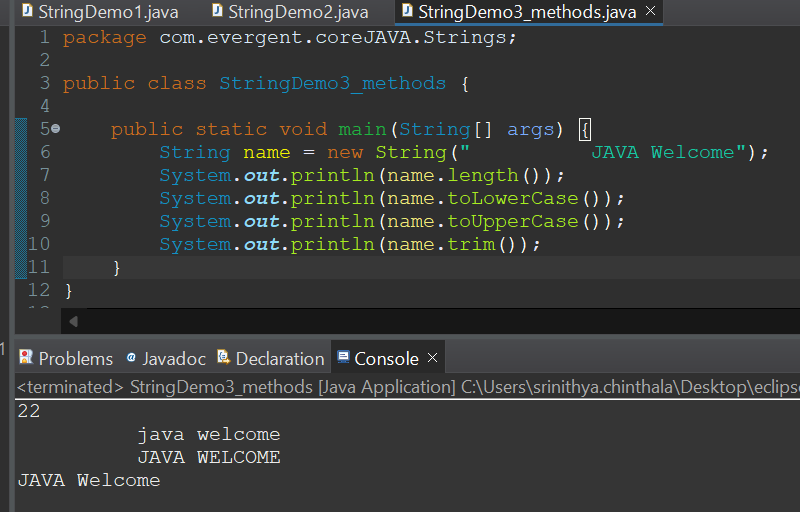
* Programs on String class
* StringDemo1*:*



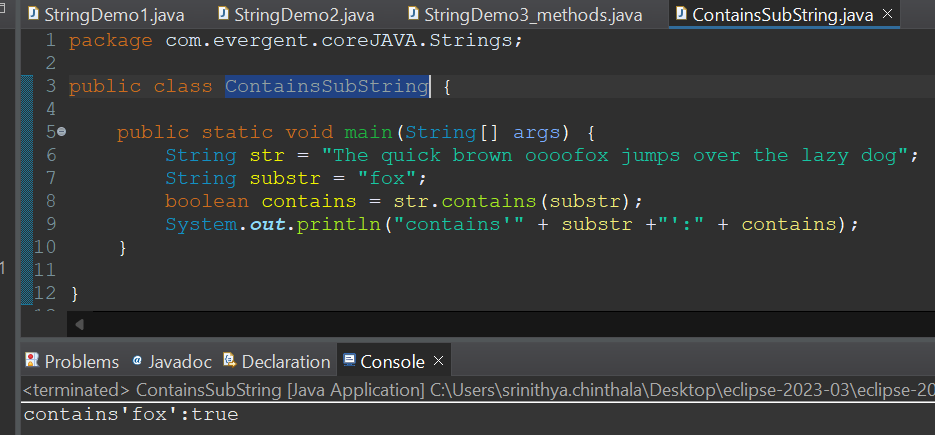
* StringDemo2:



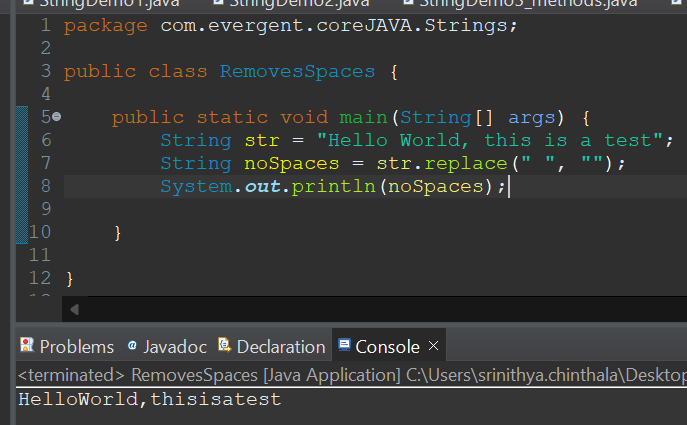
* String Demo3:



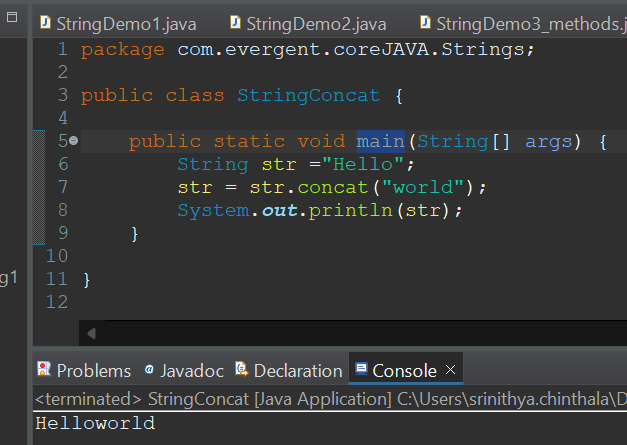
* Contains substring:



* No spaces:



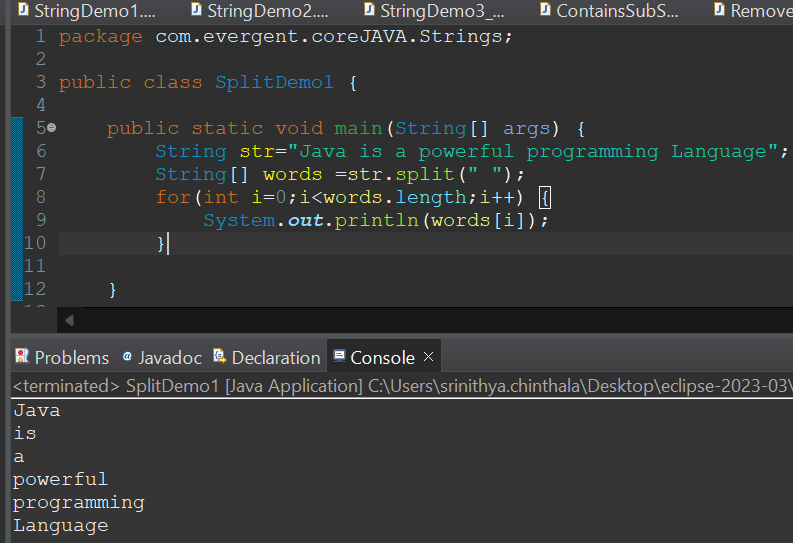
* Concat string:



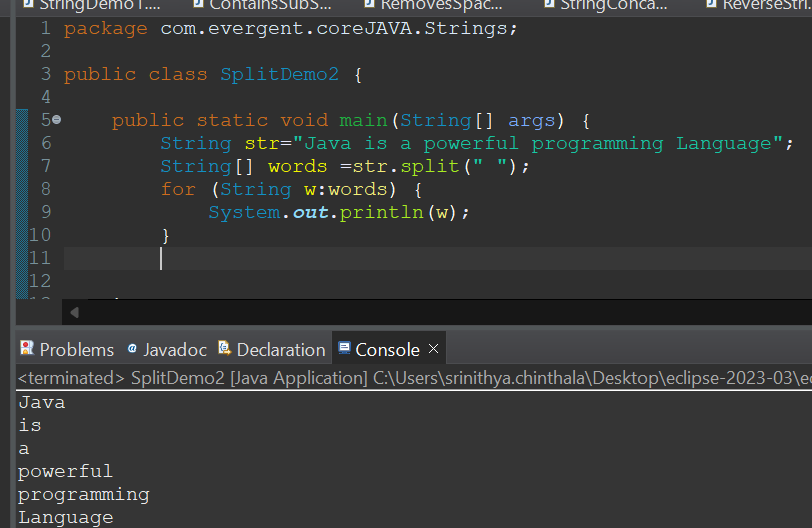
* Reverse String:



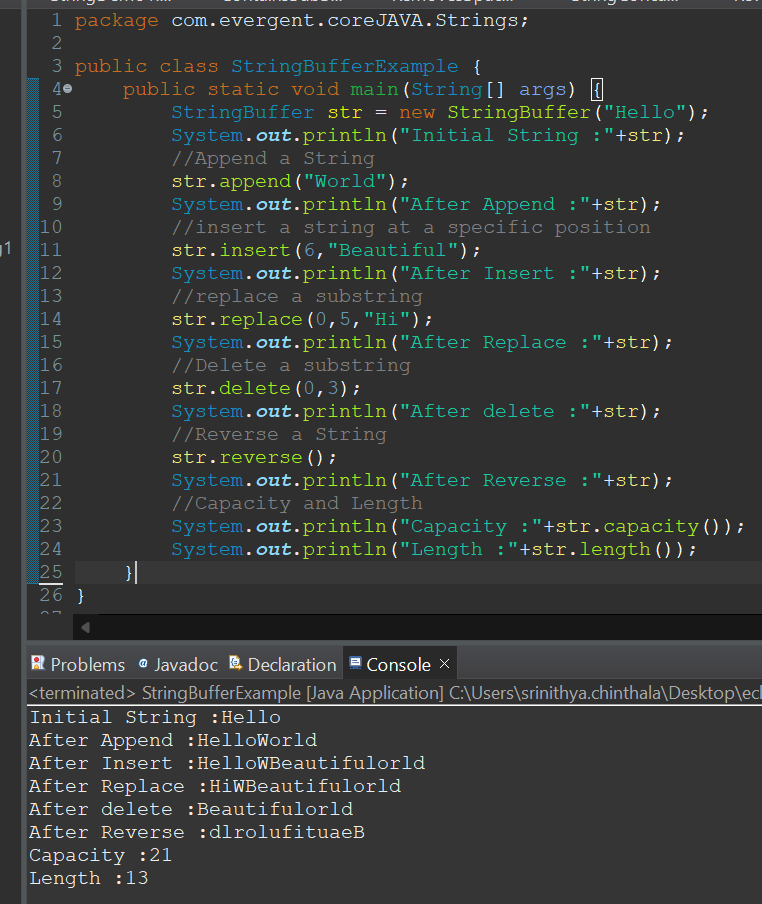
* Split demo1:



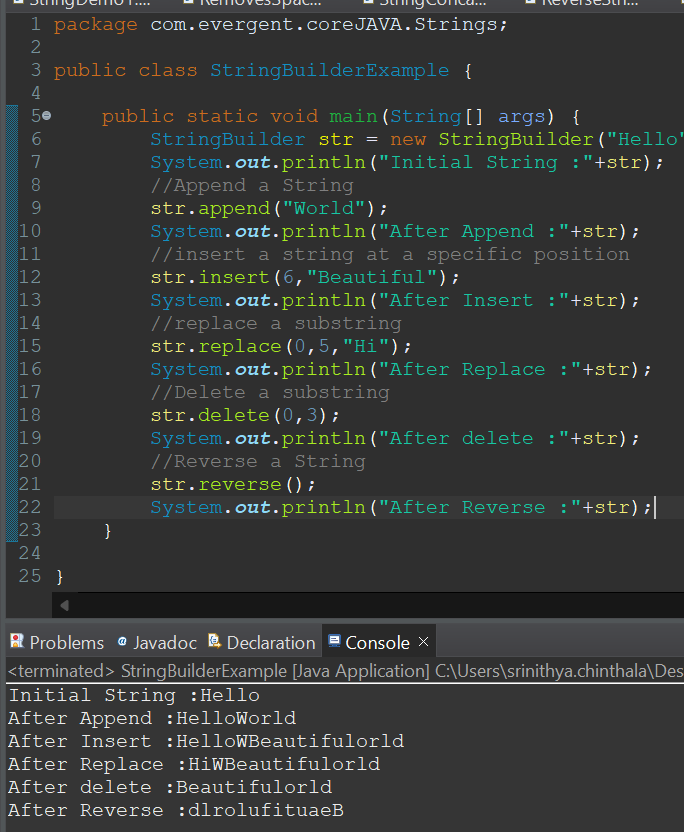
* Split demo2:



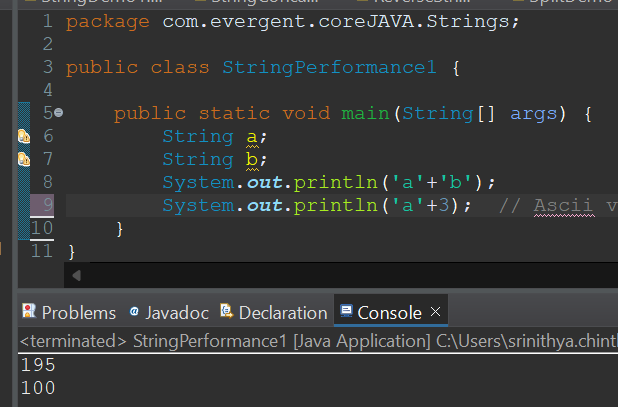
* String Buffer example:



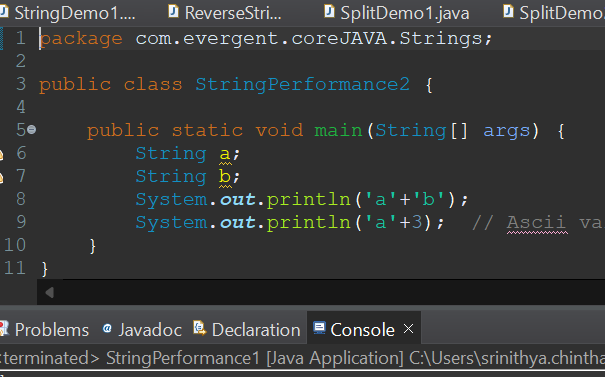
* String Builder example:



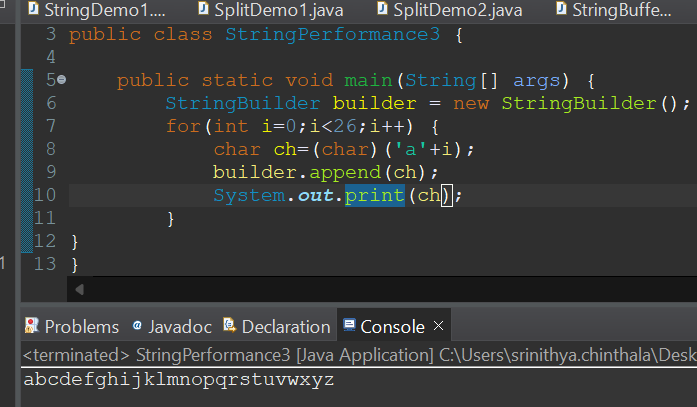
* String Performance 1:



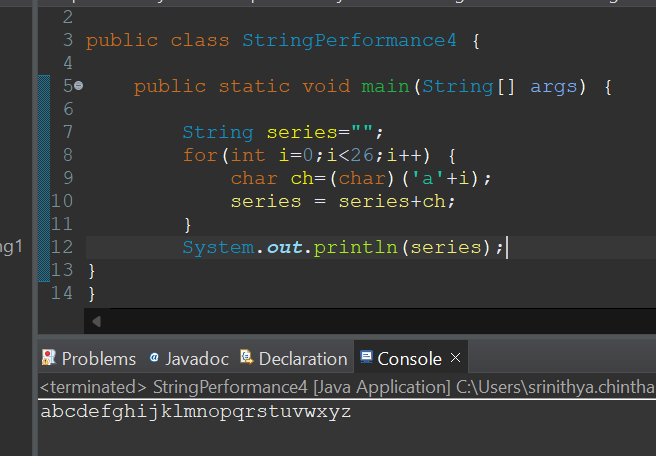
* String Performance 2:



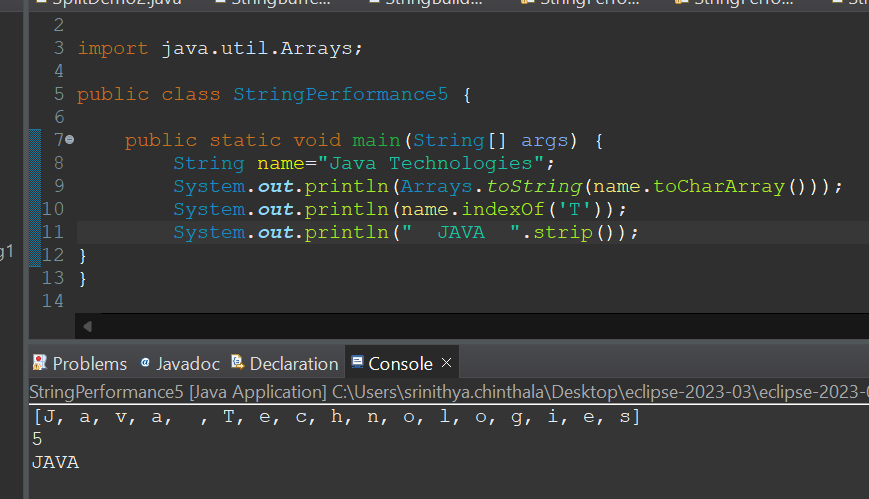
* String Performance 3:



* String Performance 4:



* String Performance 5:



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

**Morning Session:**

* Immutable Strings
* We can create our own immutable class?

1. We can declare class as final.
2. The class is declared as final so that it cannot be sub classed.
3. Private final fields:
   1. The fields name and age are private and final constructor.
4. Constructor:
   1. The constructor initializes the final fields when the person object is created.

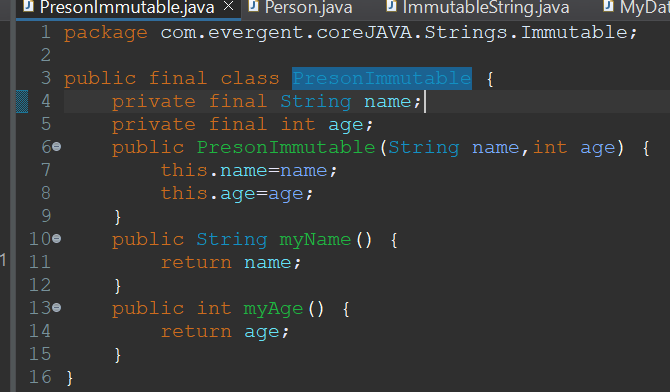
* Programs on Immutable Strings.
* **Afternoon Session:**
* Interface

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 over-ride all the interface methods otherwise the class will be showing compile time error.
5. We cannot create objects to interface but we can create reference to interface.
6. We can declare variables inside interface but all are public static final.
7. Java will support Multiple inheritance through inheritance.
8. One class can implement more than one interface.
9. One interface can extend another interface.

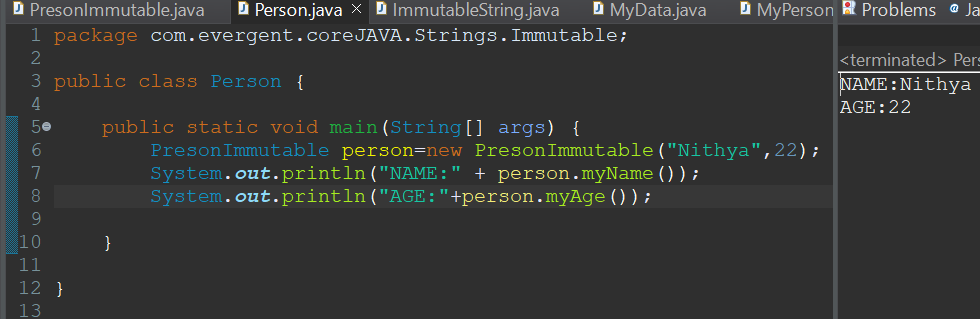
* Programs on interface

Programs:

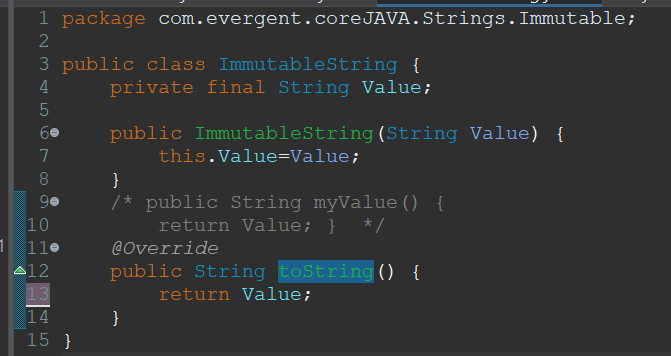
* Person Immutable:



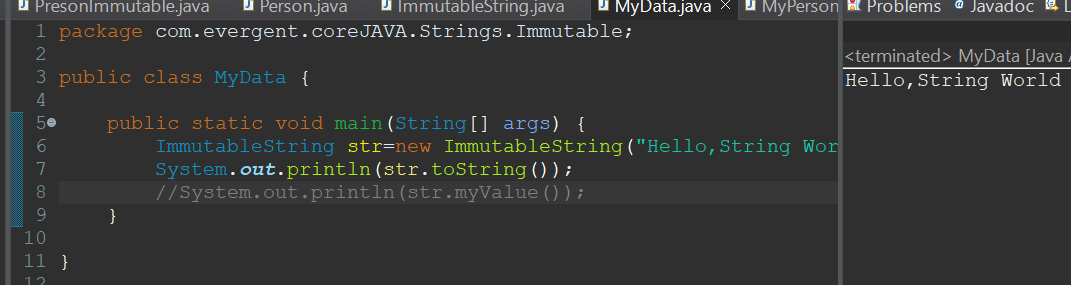
* Person:(main method of Person Immutable):



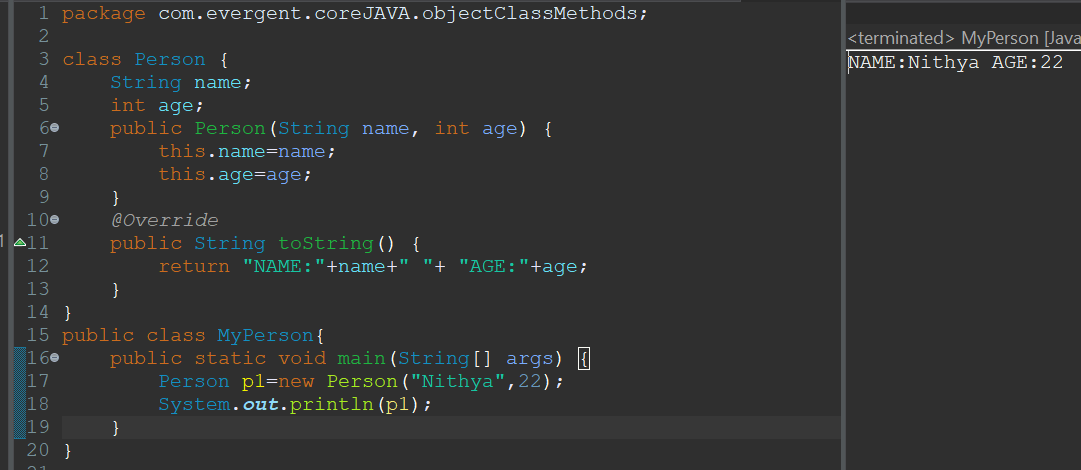
* Immutable String:



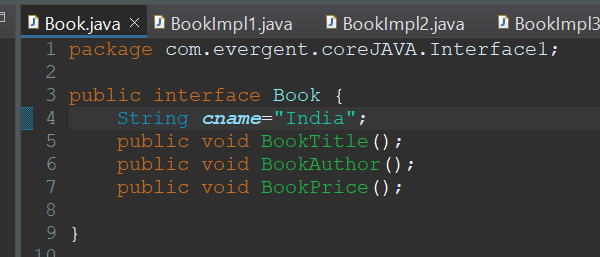
* My Data(main method of Immutable String):



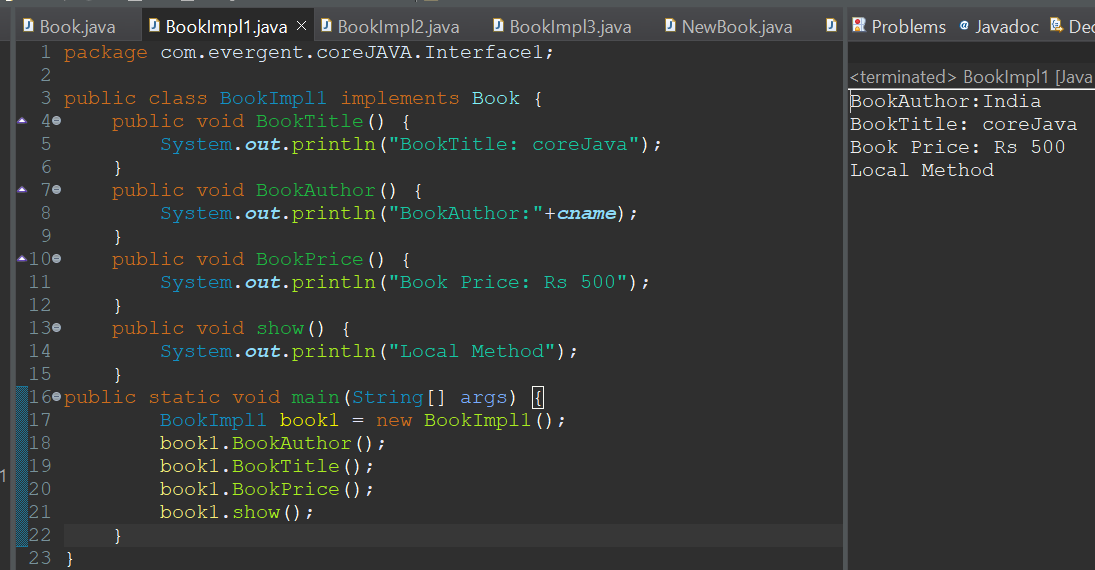
* Person:



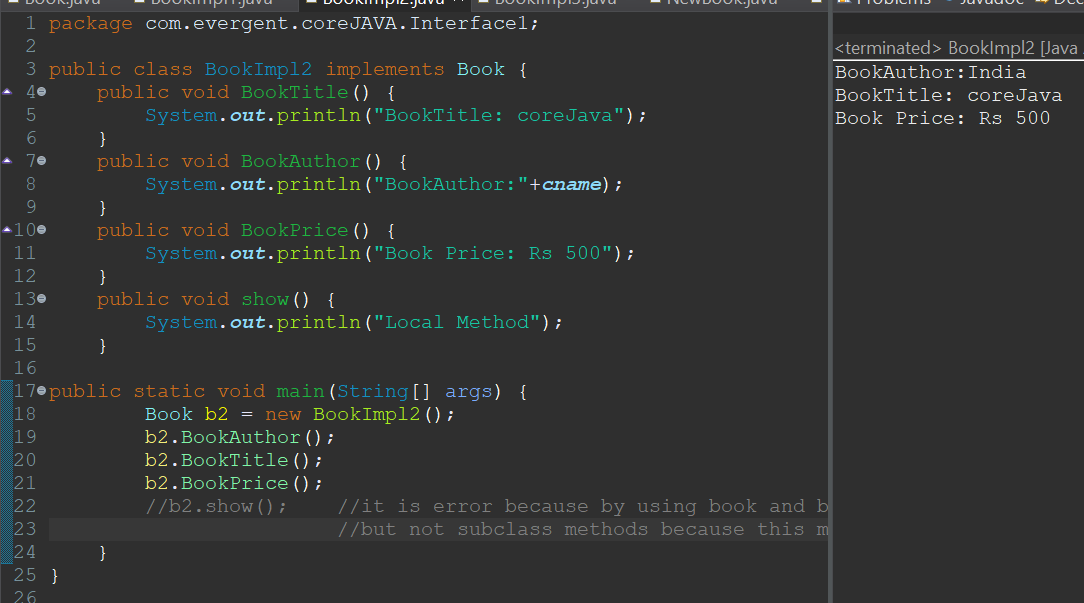
* Book(Interface1):



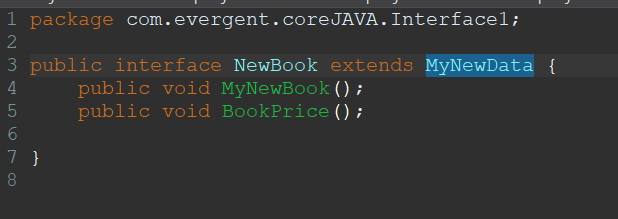
* BookImpl1(class1):



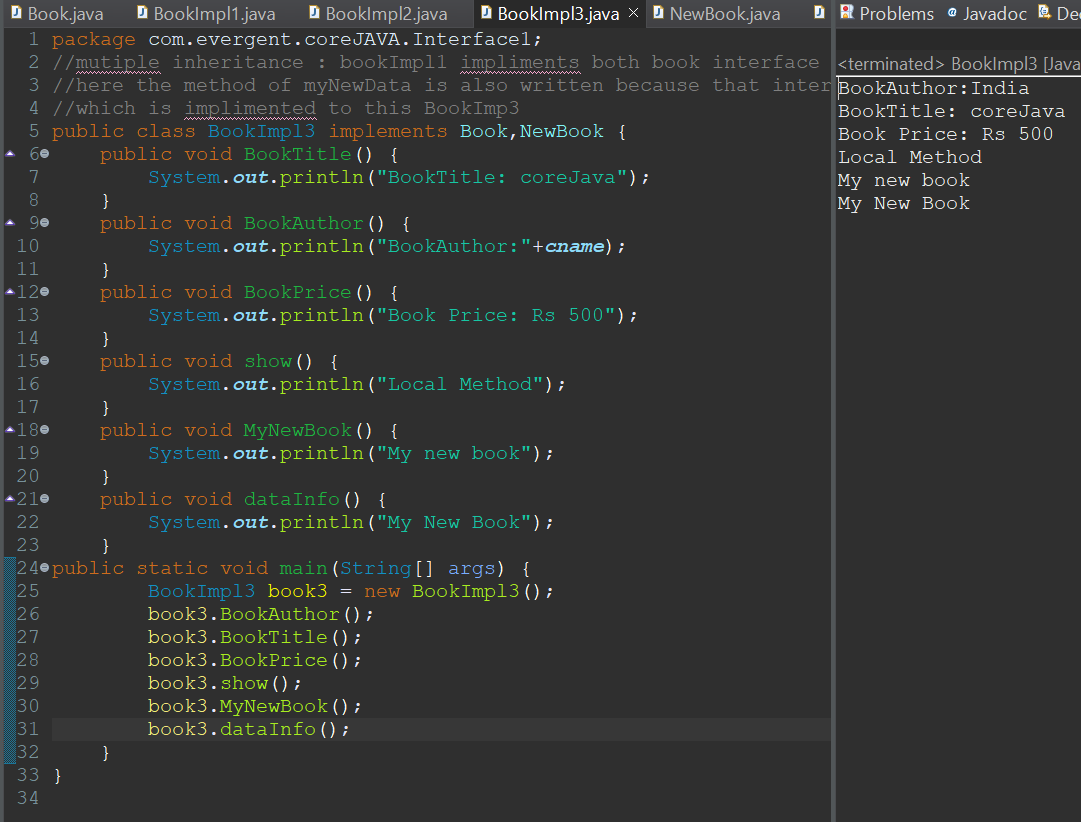
* BookImpl2(class2):



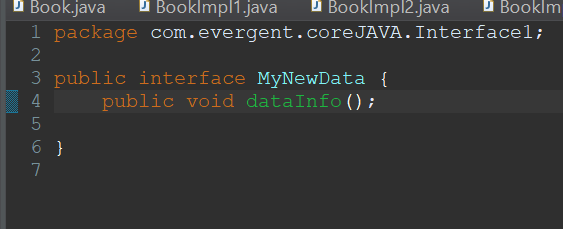
* New Book(interface2):(extends interface 3)



* BookImpl3(class3):



* My New Data(interface3):



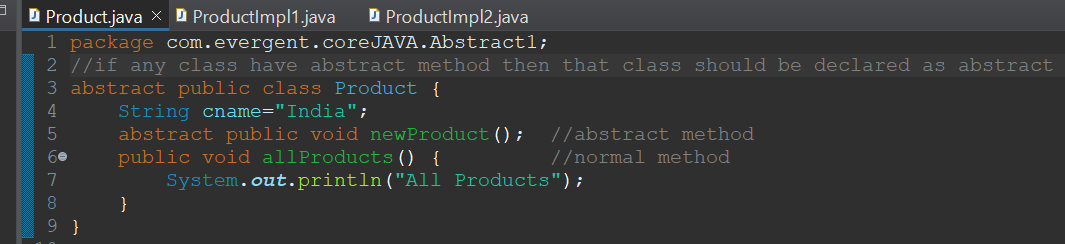
**Day 8(14-08-2024)**

**Morning Session:**

* Abstract

1. Abstract is a key word.
2. Abstract class having abstract methods and concrete methods (implemented).
3. If any class having abstract method that class should be declared with abstract keyword otherwise it will show compilation error.
4. If any class extends abstract class then that class should override all abstract methods otherwise the class shows compilation error.
5. We cant create object to abstract class we can only create reference to abstract class.

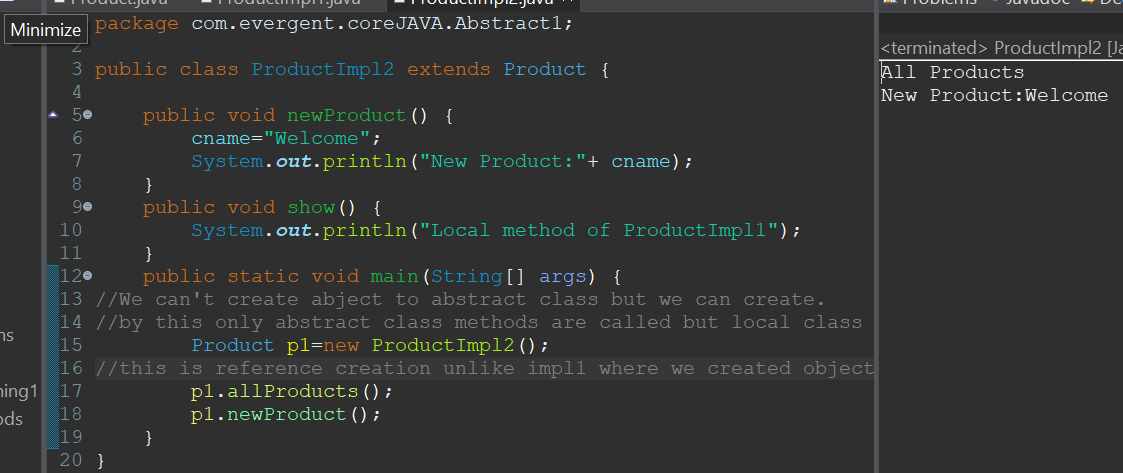
* Product:



* ProductImpl1:



* ProductImpl2:



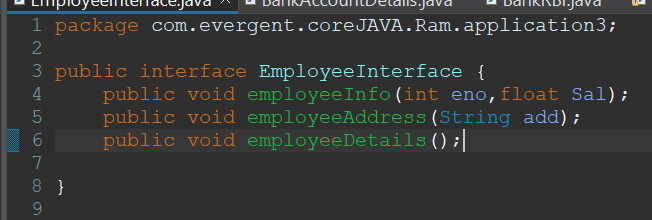
**Day 9(16-08-2024)**

**Morning Session:**

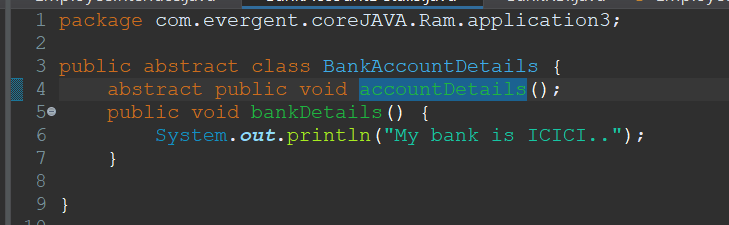
* Assessment

**Afternoon Session:**

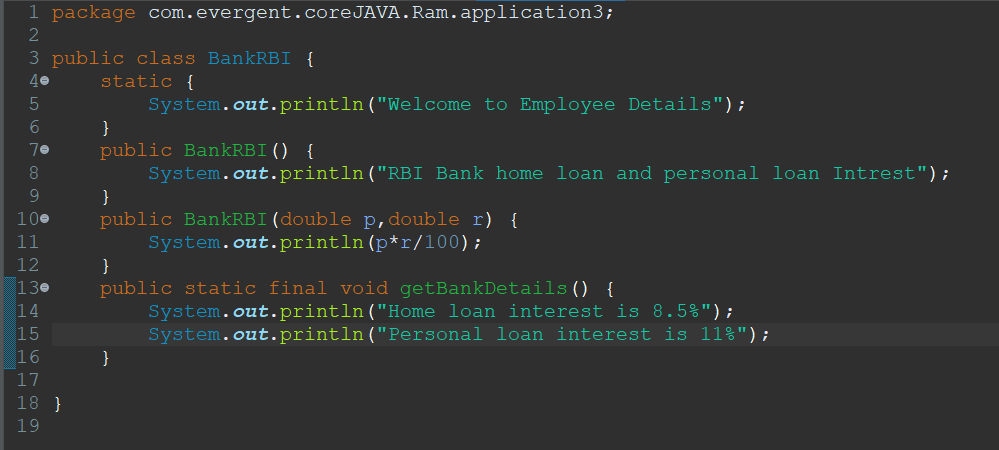
* Explained about an application and explained about some interview questions.
* Employee Interface (interface):



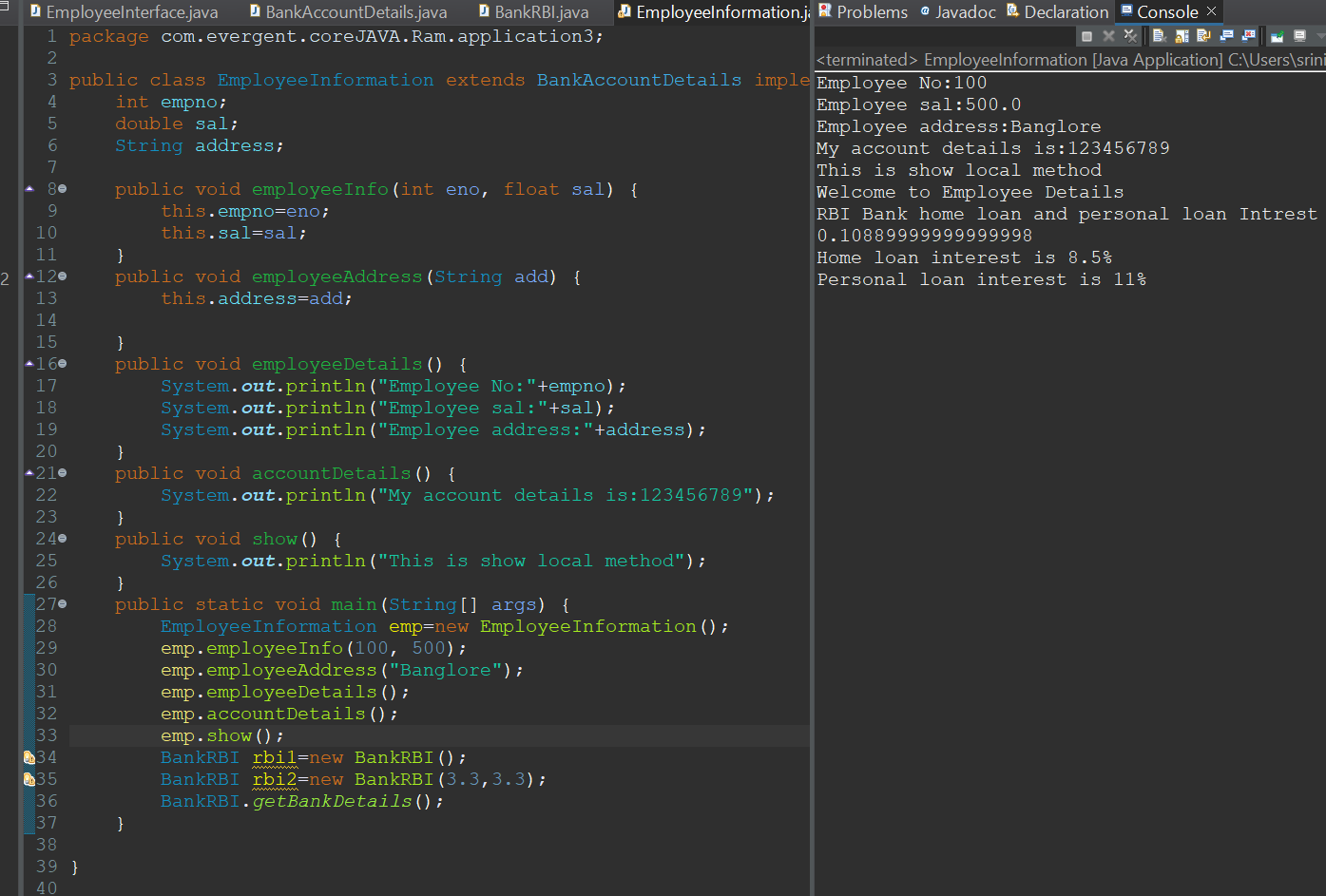
* Bank account details (abstract class):



* BankRBI:



* Employee Information:



**Day 10(19-08-2024)**

**Morning Session:**

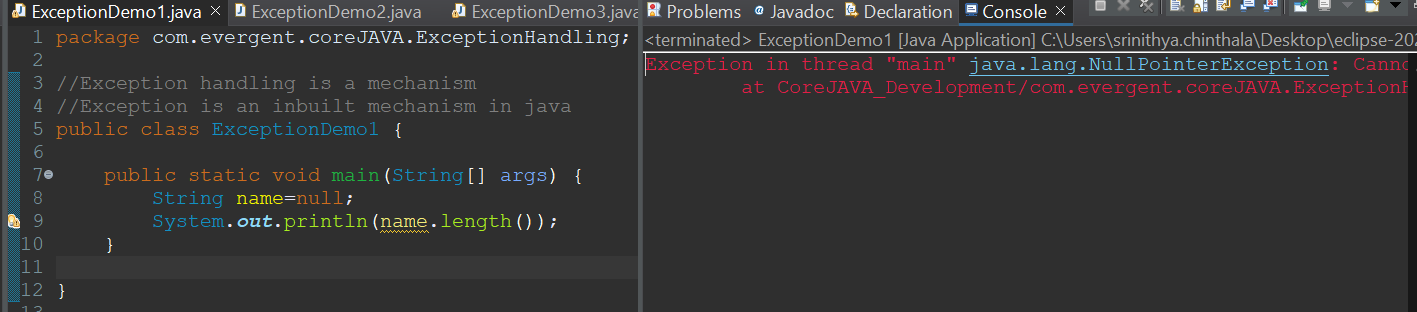
* Revision of completed topics: oops, constructors, static, final, strings, interfaces, abstract classes.

**Afternoon session:**

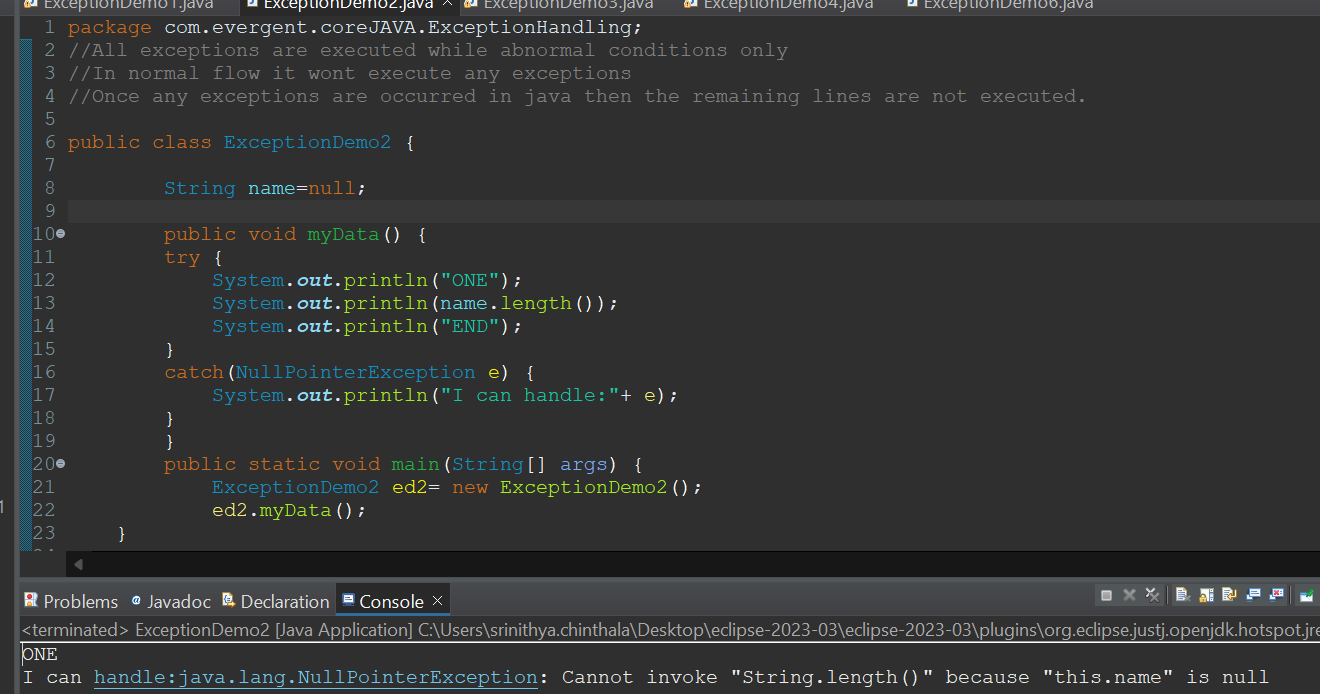
* Exception Handling:

1. Exception Handling is a Mechanism.
2. Exceptions are inbuilt Mechanism of JAVA.
3. All Exceptions are executed during abnormal conditions only.
4. Normal flow it won’t execute any exceptions.
5. Once any exceptions are occurring in java code, then remaining lines of code is unreachable.
6. Java.lang.Throwable is super class for exception and errors.
7. There are two types of Exceptions in JAVA: a.Checked Exceptions b.Unchecked Exceptions
8. All checked Exceptions are compile time exceptions.
9. All unchecked Exceptions are run time exceptions.
10. There are 5 keywords in Exception Handling: a.Try b.Catch() c.Finally() d.Throws() e.Throw
11. Try is for business logic.Catch for handling exceptions.
12. Catch is used to handle exceptions
13. Finally is a block, if exceptions occurs or not finally block will be executed.
14. Throws an exception will be executed method by method
15. Throw is for runtime exceptions and will call predefined exceptions or user defined exceptions.
16. Try followed by either catch block or finally block.
17. We should follow exceptions hierarchical.
18. We can create our own (User Defined) exceptions.
19. Our own exceptions extends exception or runtime exception.
20. All Exceptions classes are in to java.lang package.
21. If there are 2 exceptions in try, developer should handle 1st exception after that 2nd exception will be handled.
22. Developer can’t handle errors.

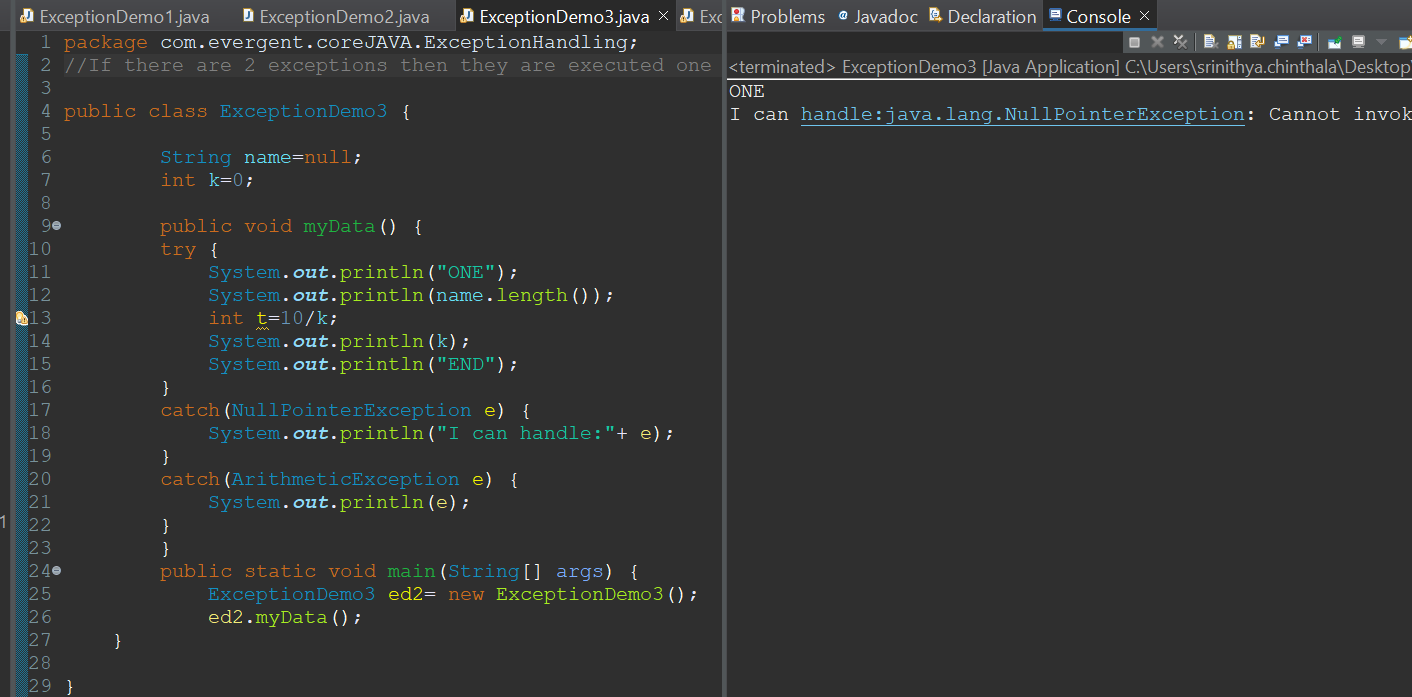
* ExceptionDemo1:



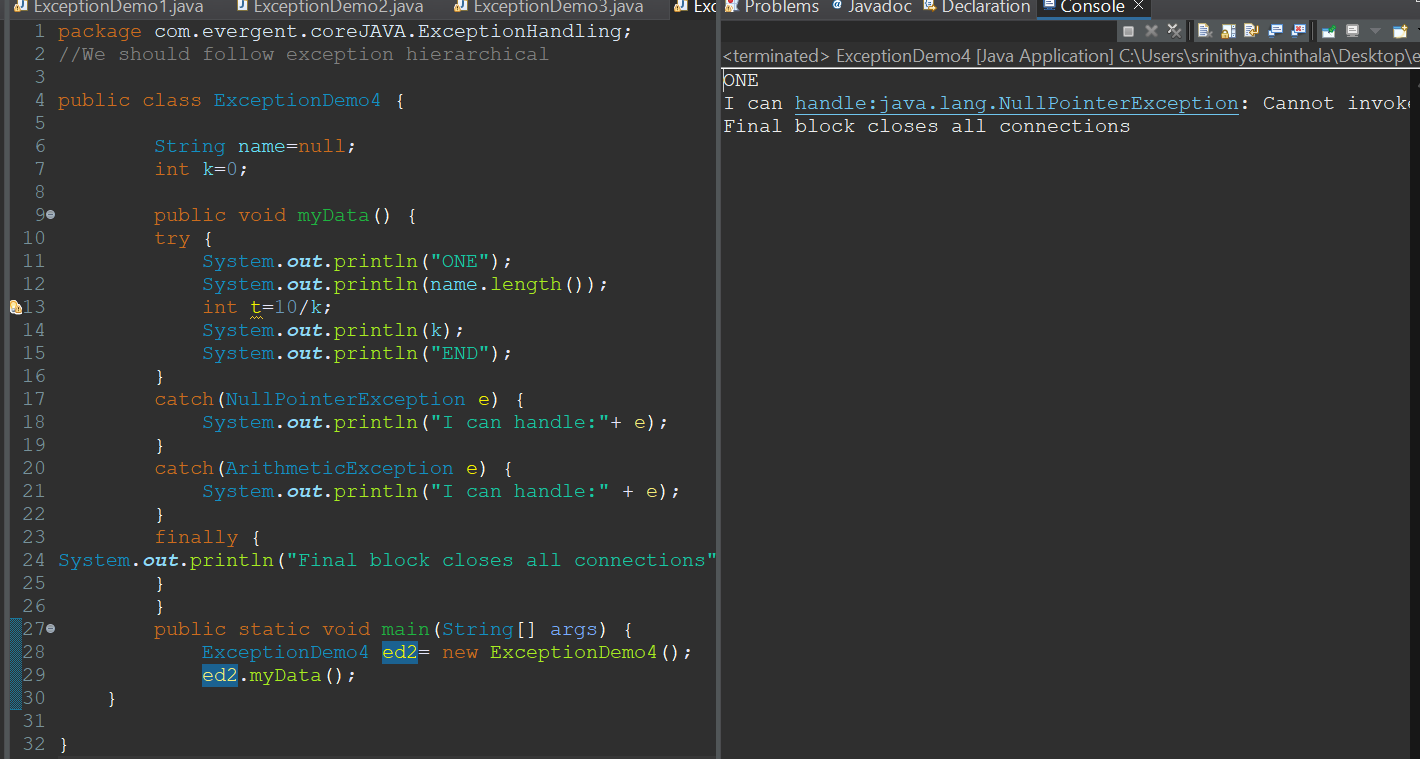
* ExceptionDemo2:



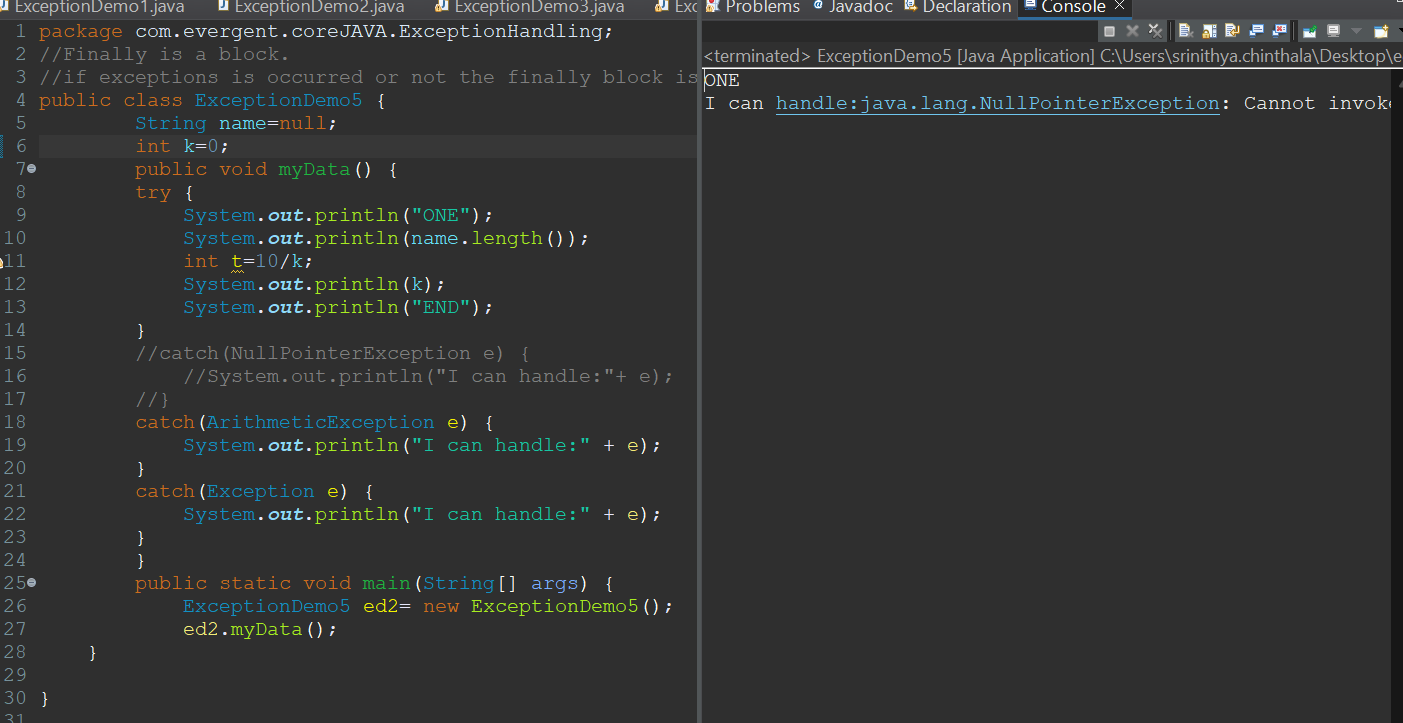
* ExceptionDemo3:



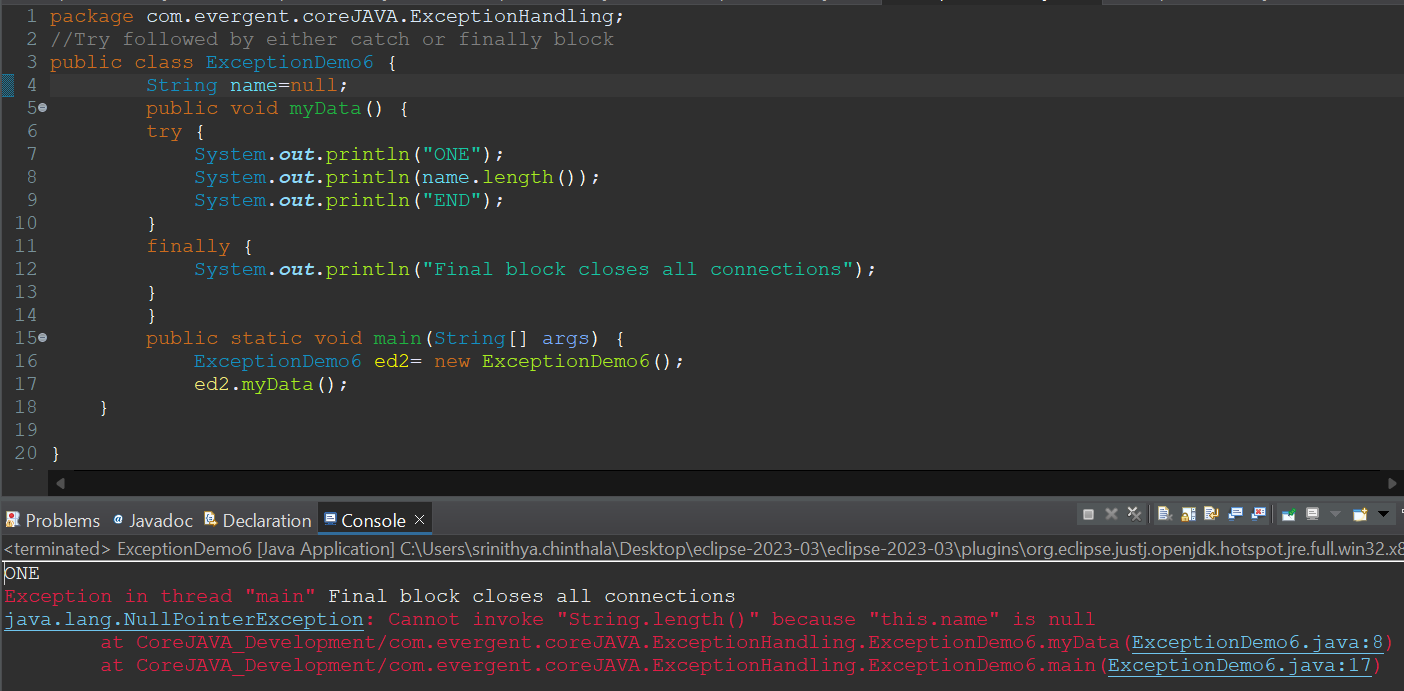
* ExceptionDemo4:



* ExceptionDemo5:



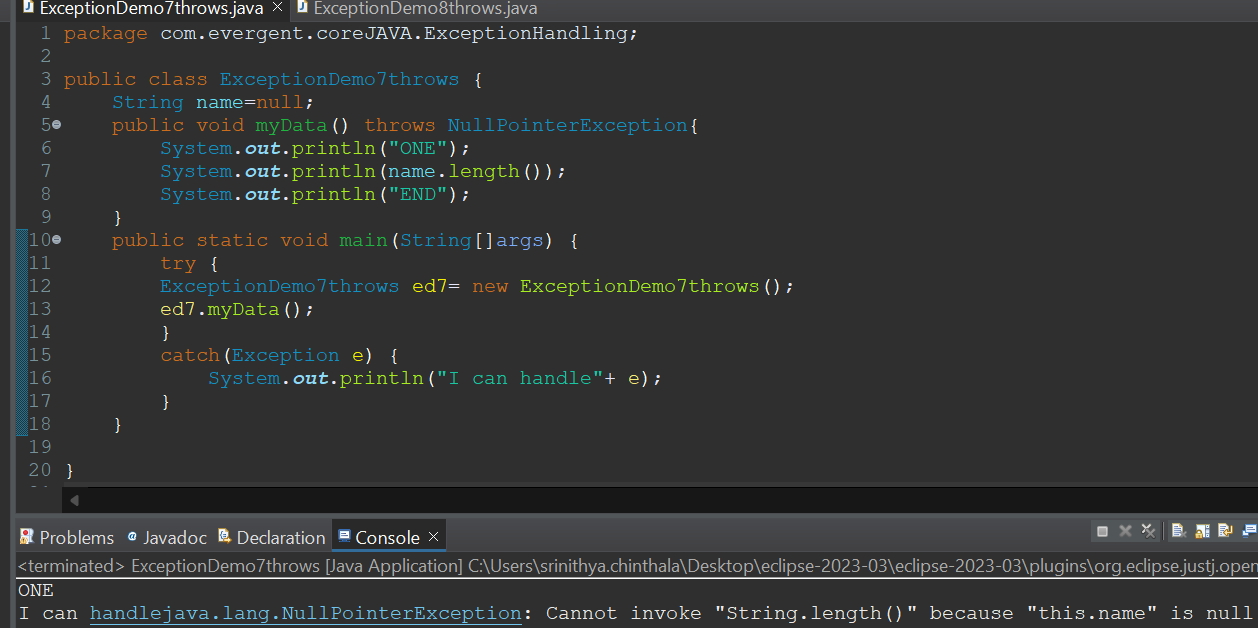
* ExceptionDemo6:



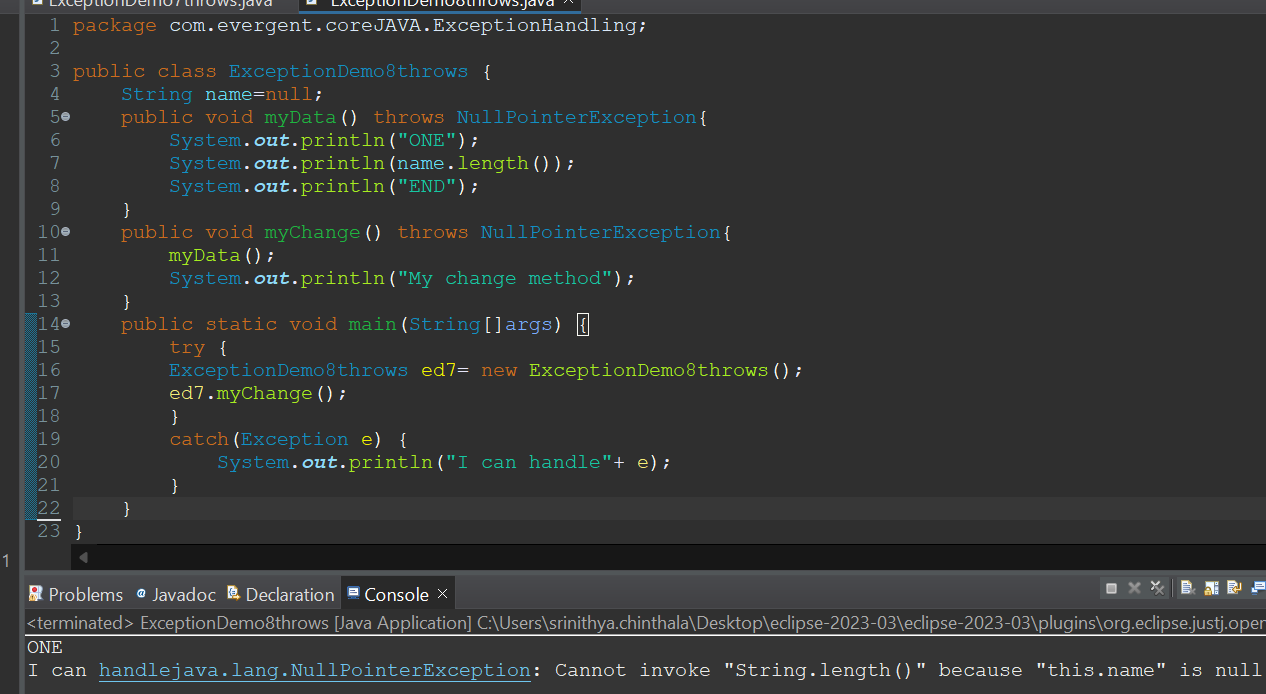
**Day 11(20-08-2024)**

**Morning Session:**

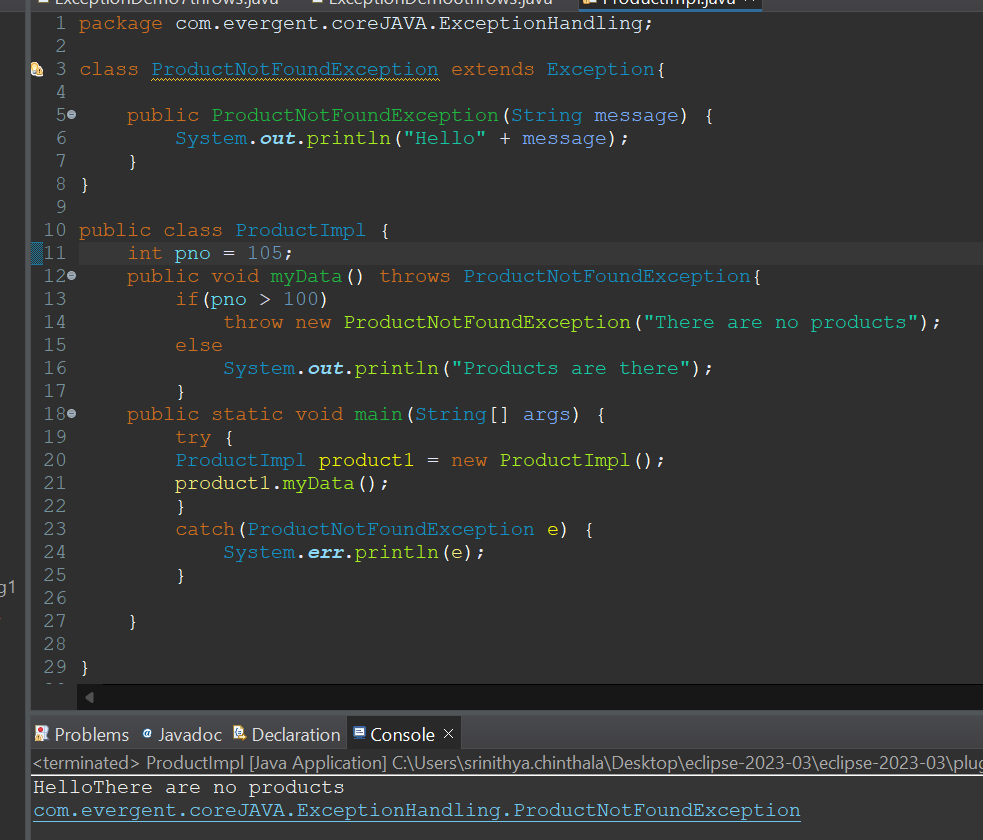
* Continuation of exception handling.
* Some more programs on throws.
* ExceptionDemo7throws:



* ExceptionDemo8throws:



* ProductImpl:(throw)



**Day 11(21-08-2024)**

**Morning Session:**

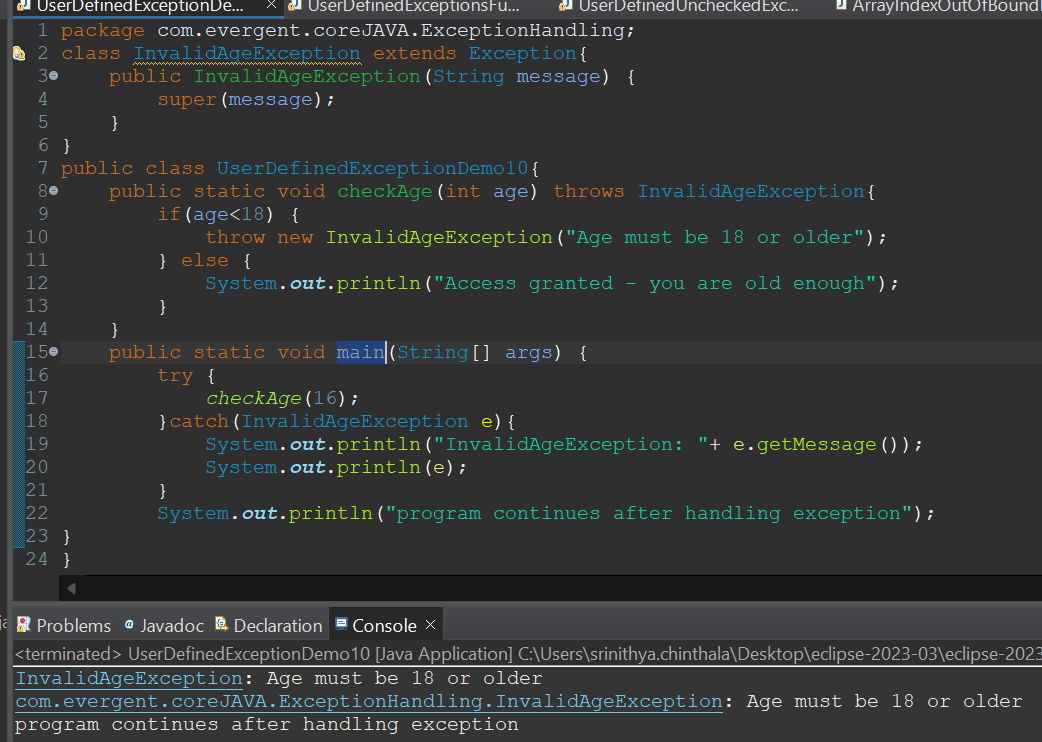
* Continuation of exception handling.
* User defined exception handling.
* User defined unchecked exception handling.

**Afternoon Session:**

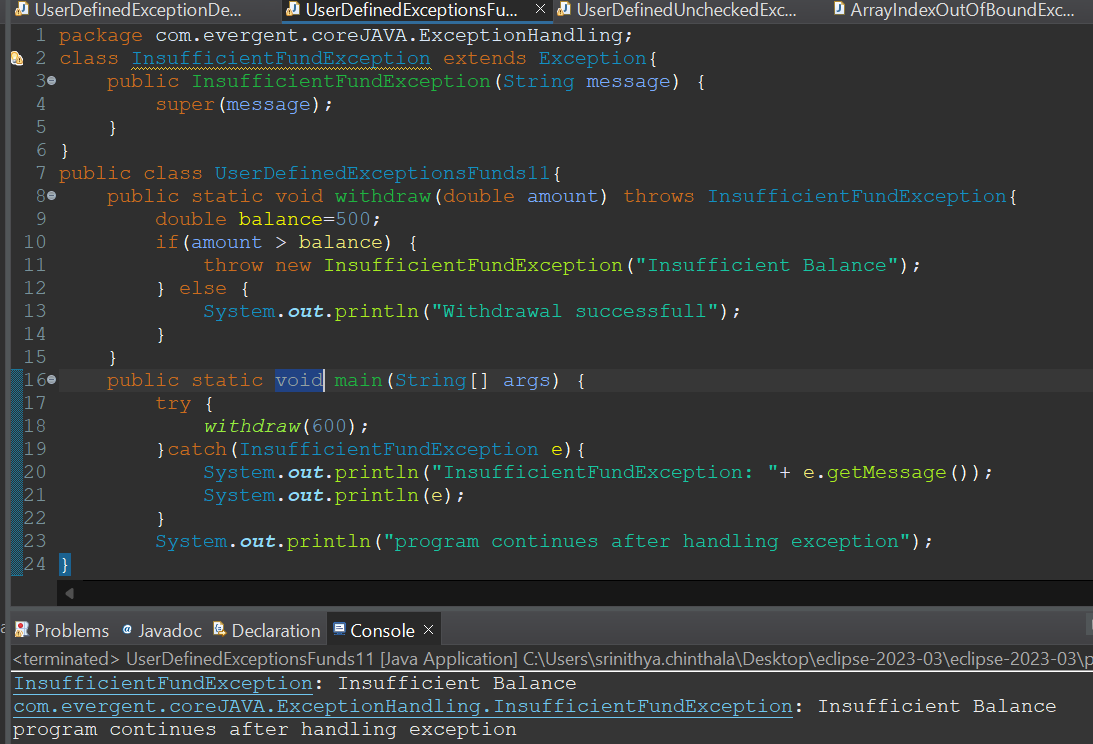
* JAVA BEANS

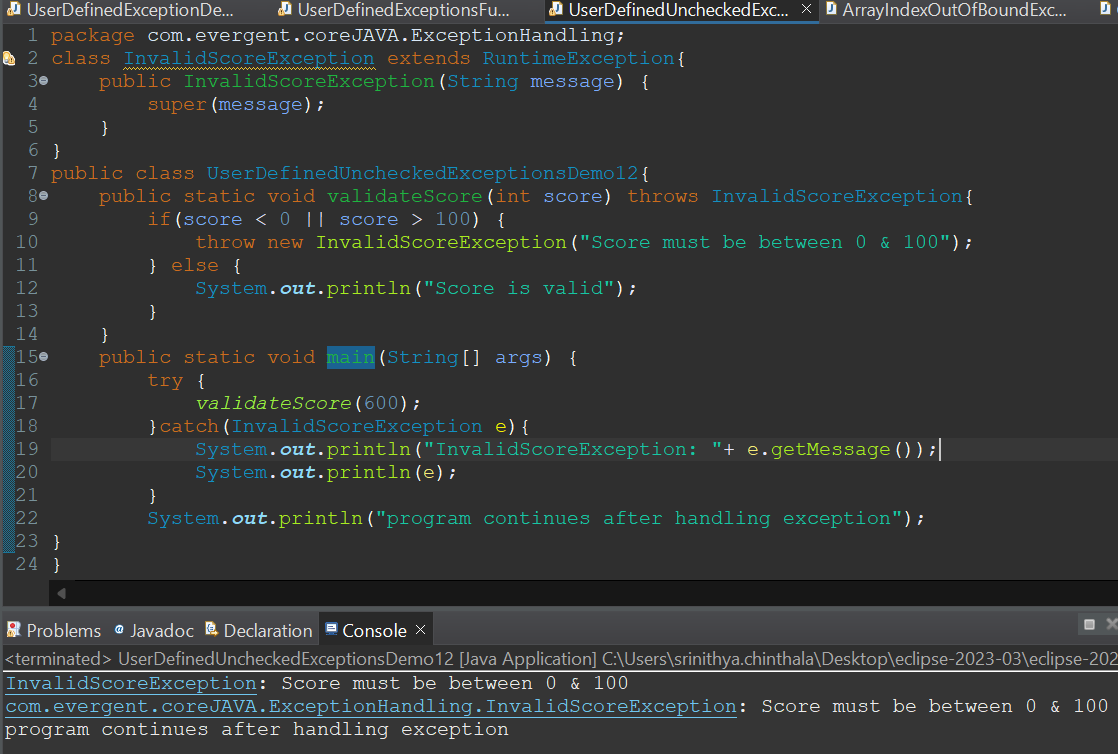
1. Java beans is a mechanism.
2. Java beans is light weight.
3. All attributes are private ans set/get methods are public implements serializable interface.
4. We can achieve tightly encapsulation through java beans.

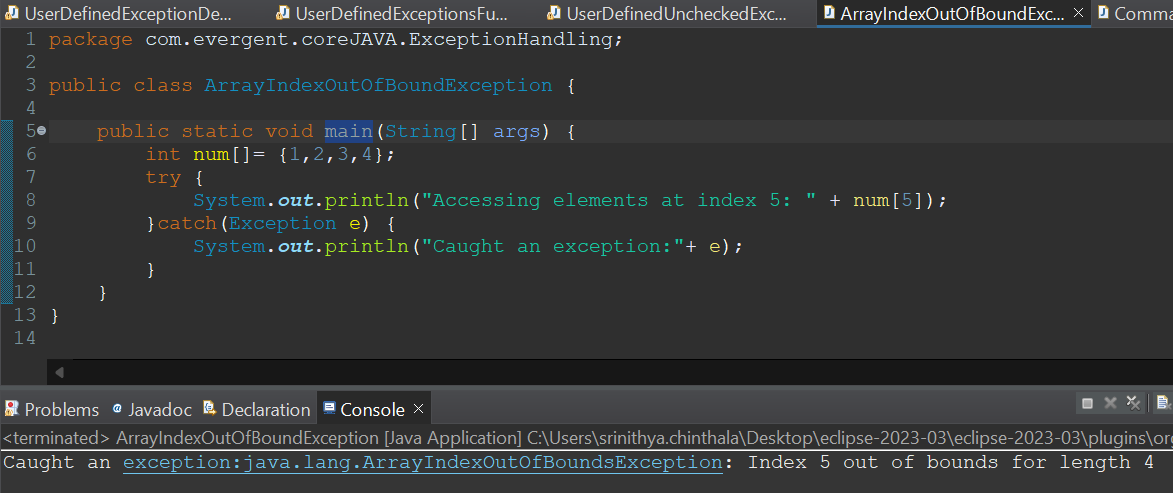
* UserDefinedExceptionDemo10:



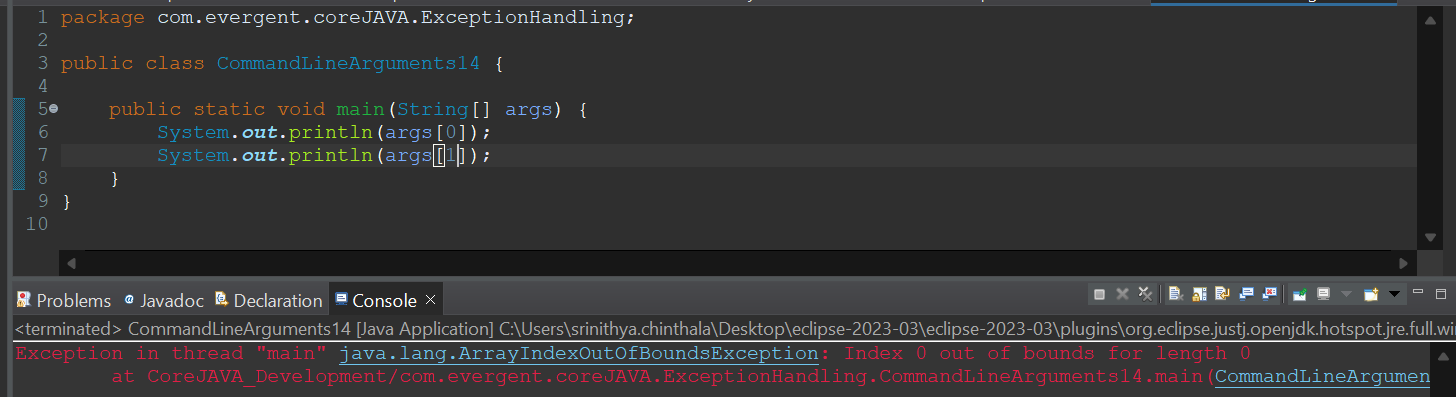
* UserDefinedExceptionsFunds11:



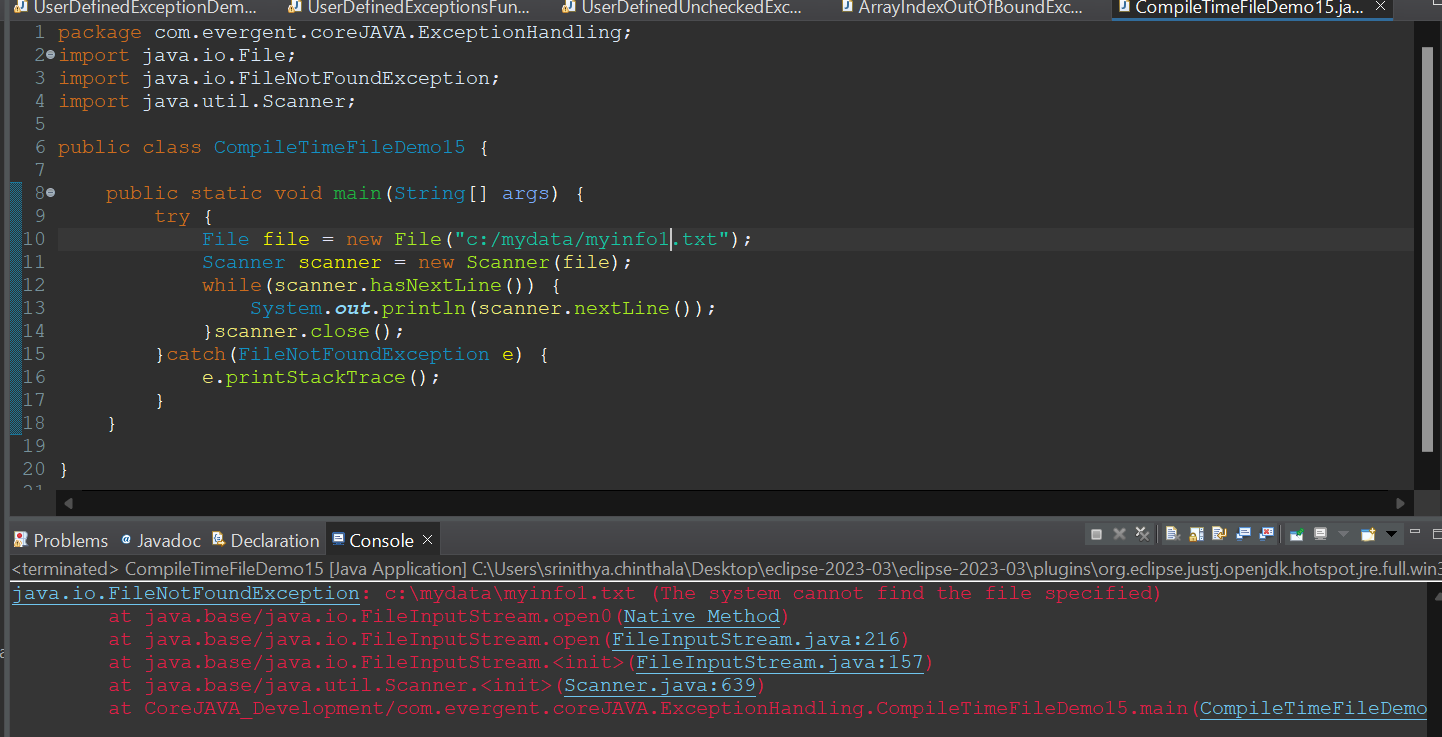
* UserDefinedUncheckedExceptionsDemo1
* ArrayIndexOutOfBoundException:



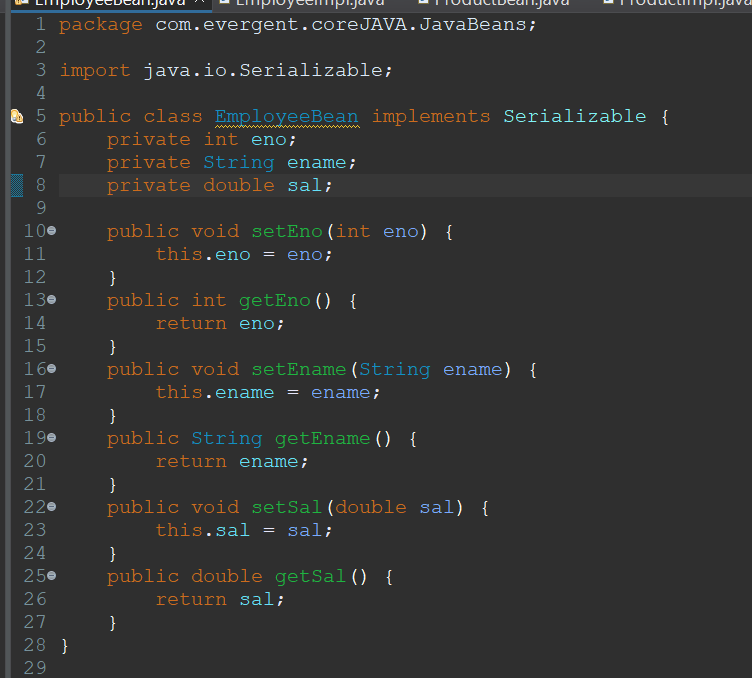
* CommandLineAurguments14:



* CompileTimeFileDemo15:



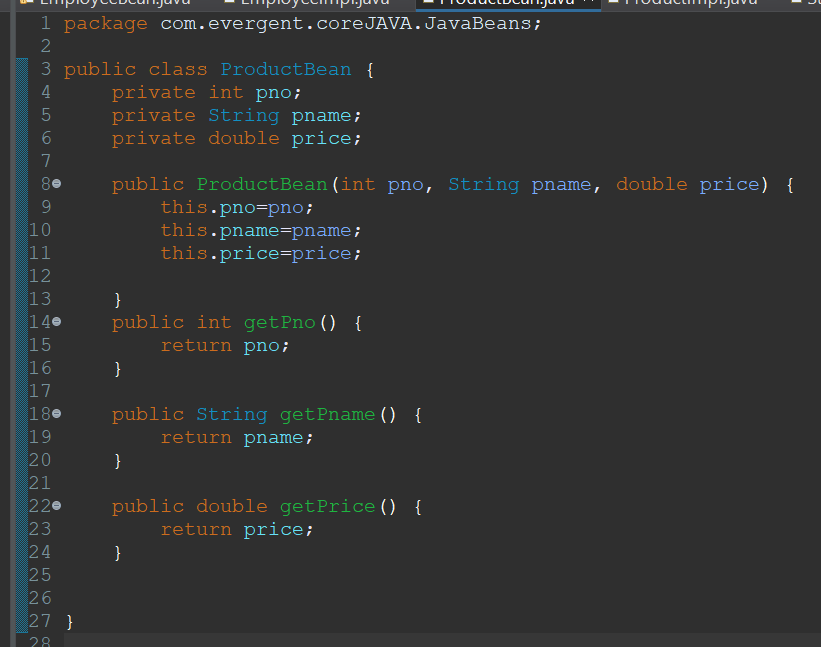
* EmployeeBean(set/get) :



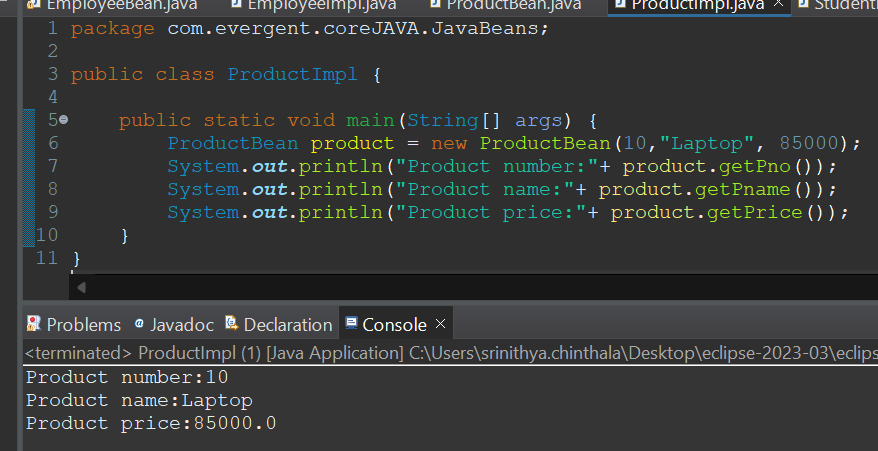
* EmployeeImpl:



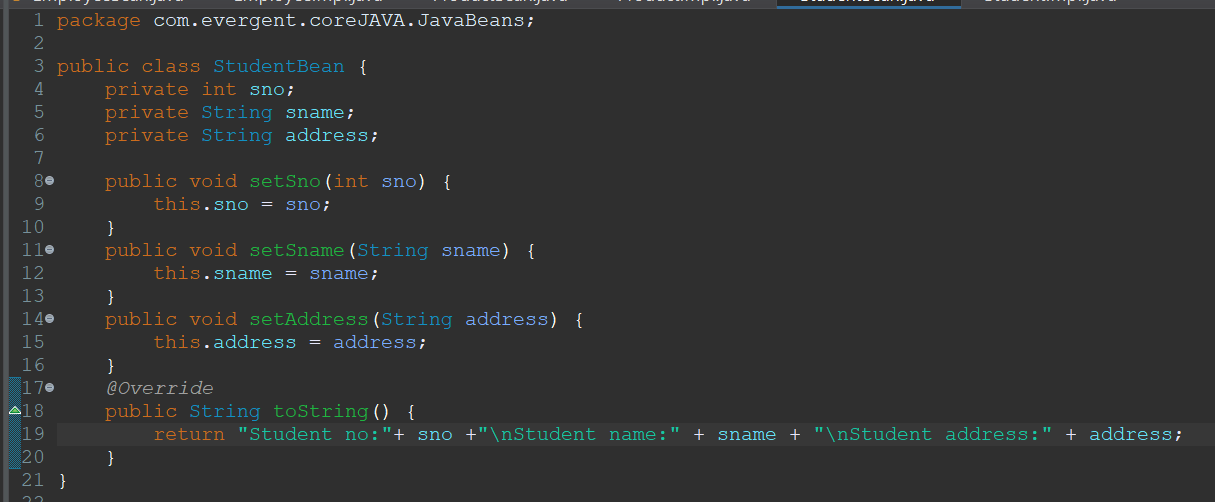
* ProductBean(Constructor and get):



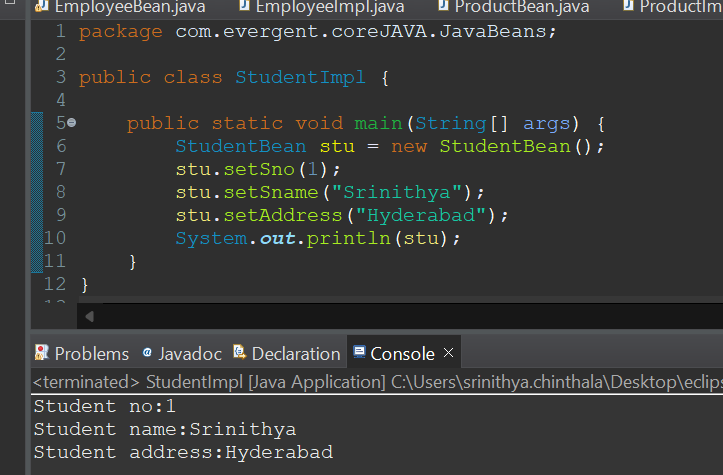
* ProductImpl:



* StudentBean(set and toString):



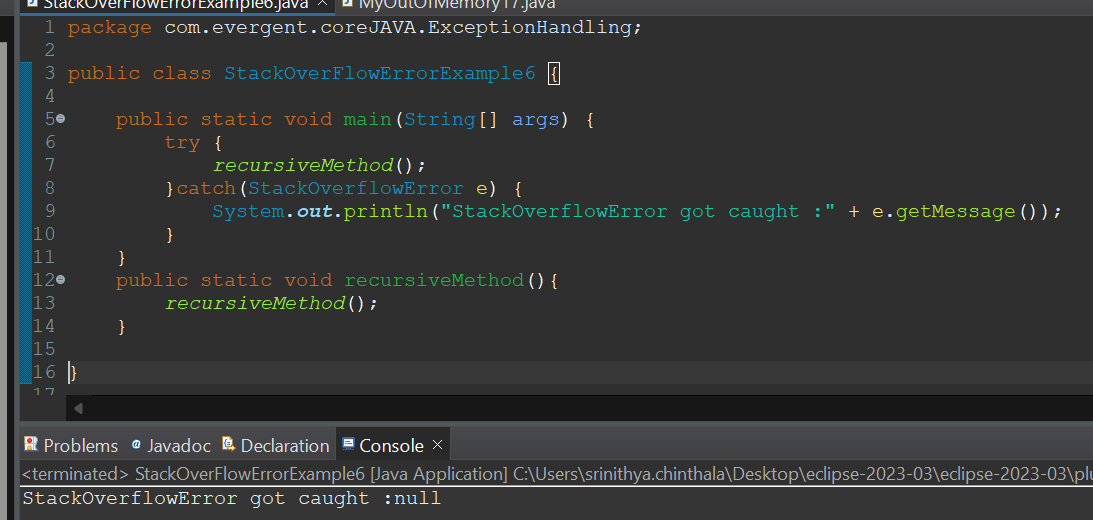
* StudentImpl:



**Day 13(22-08-2024)**

**Morning Session:**

* exception handling(Errors)
* Overview of Git and github
* StackOverFlowErrorExample6:



* MyOutOfMemory17:

