**INDEX\_V1**

Date:-05/08

1. **language and Applications**
2. **Java Features**

**Why java is platform independent**

**OOps**

**Exception handling**

**Multithreading**

**Web Applications**

**Open Source**

**Secure**

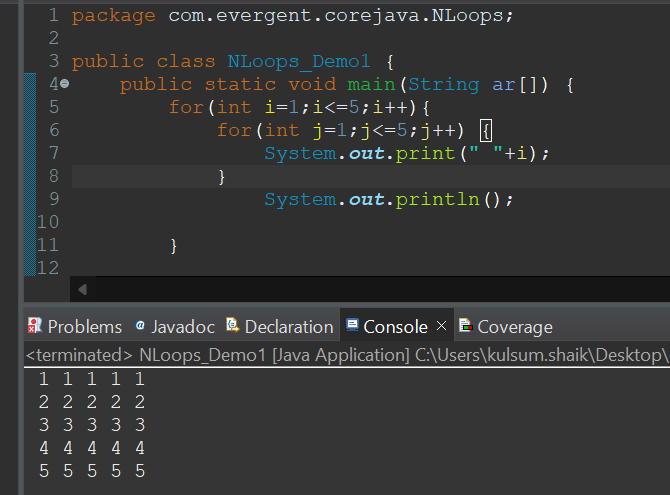
**Support Networking**

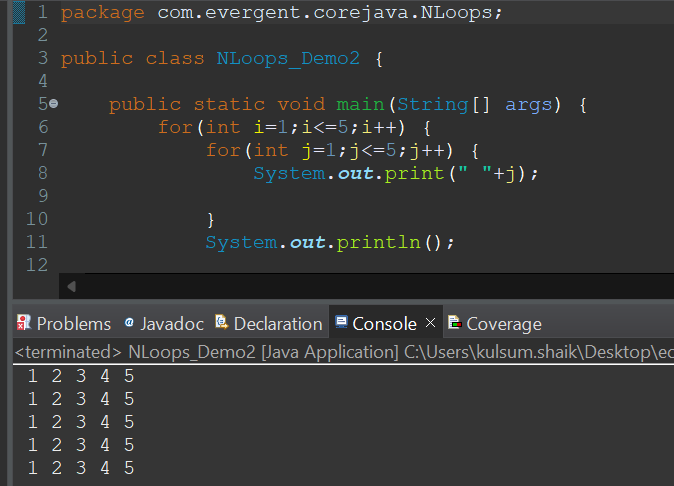
**Memory Management(Garbage Collector)**

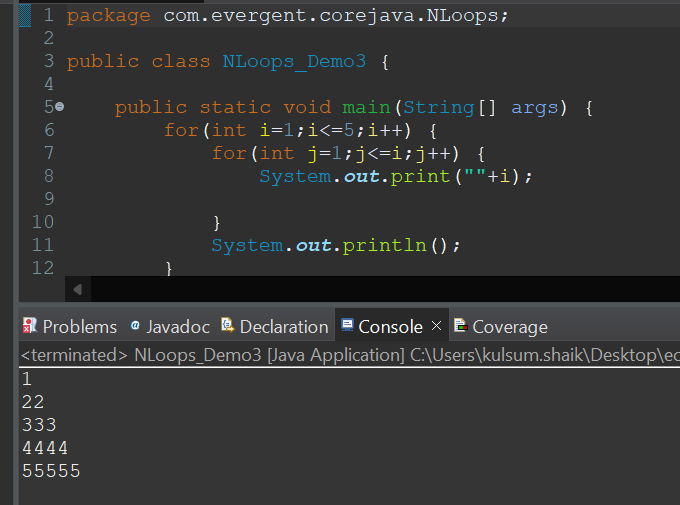
1. **JDK ,JRE ,JVM**
2. **Basic Java Programming**
3. **Packages**

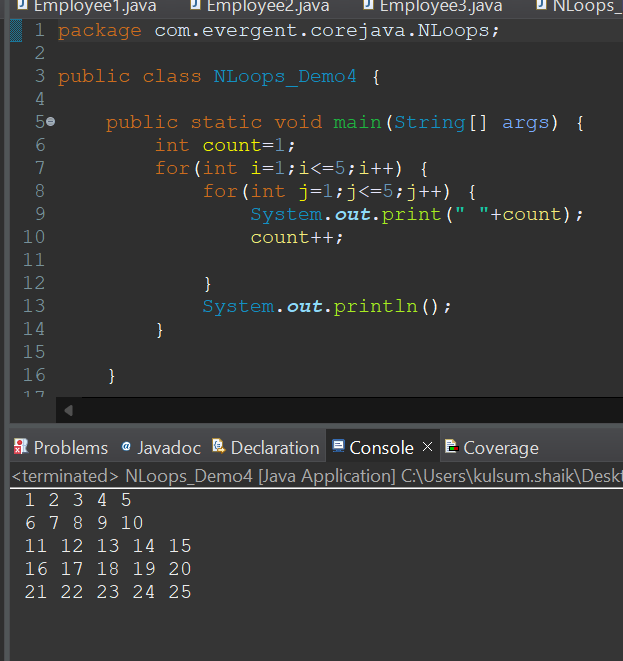
**Date:06/08/2024**

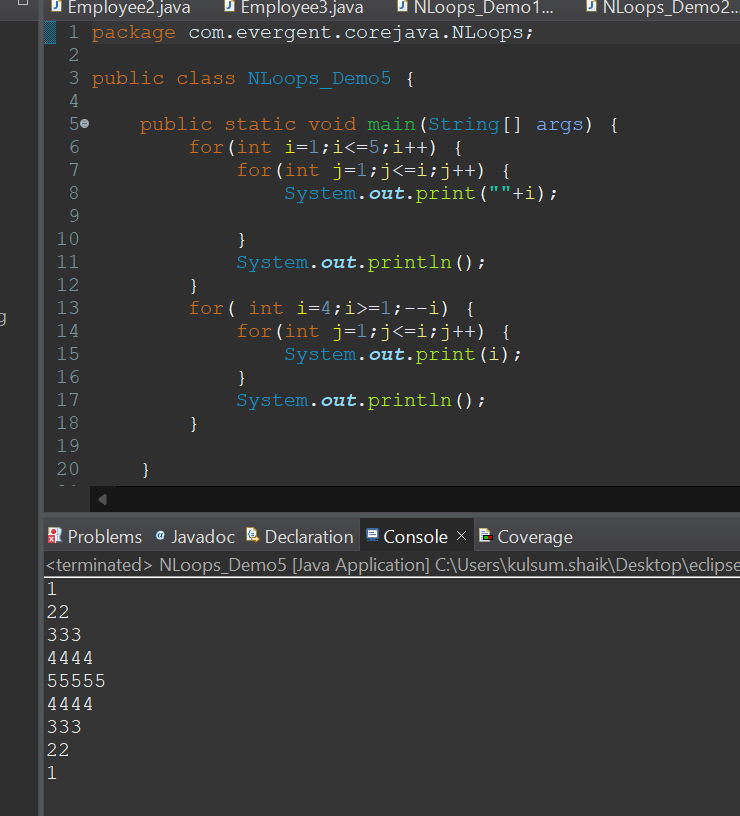
1. **Nested Loops**

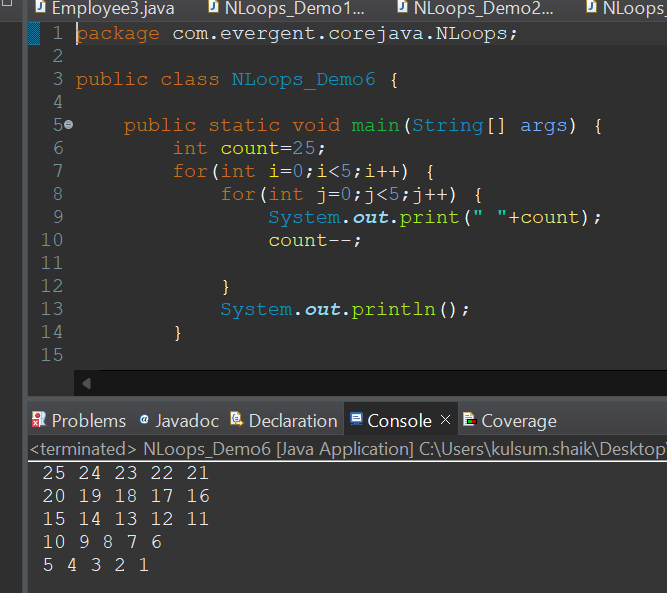




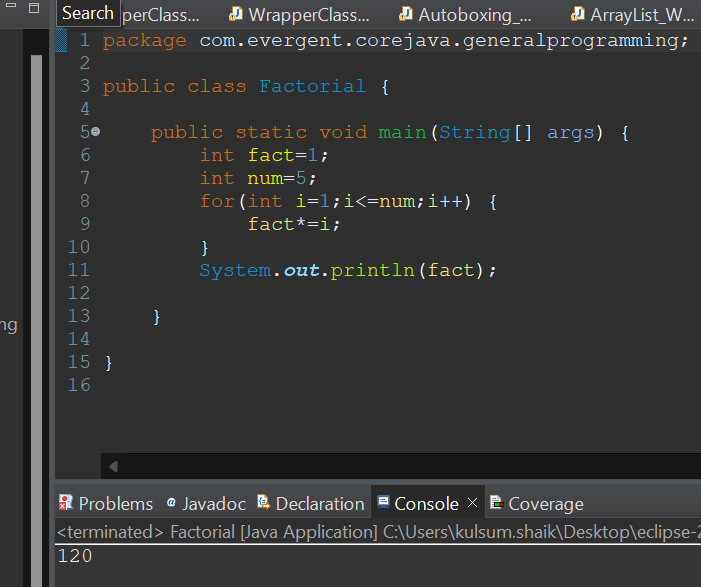


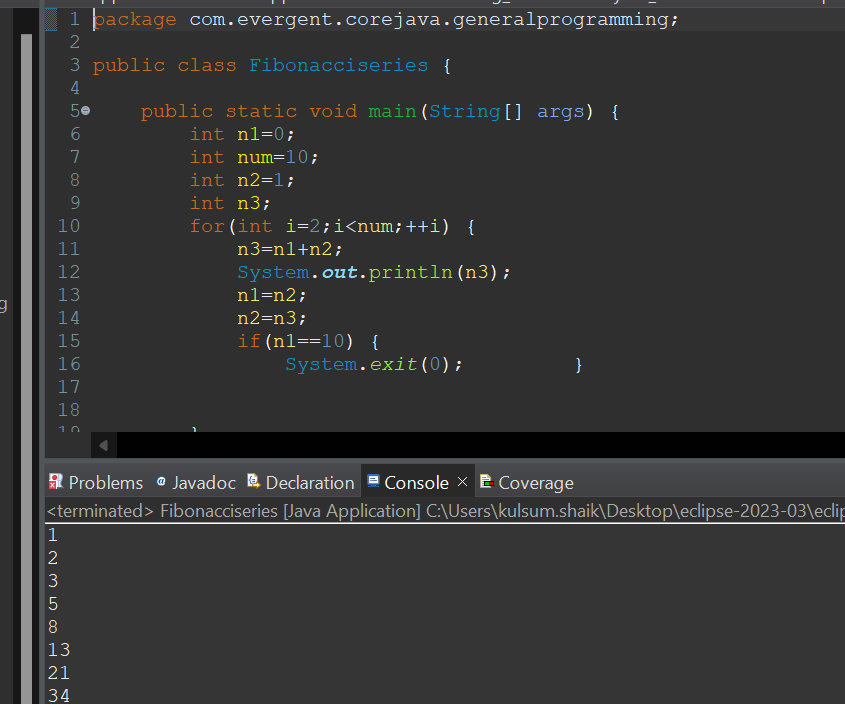






1. **Single Array**
2. **Two Dimensional Array**
3. **Logical Programming**





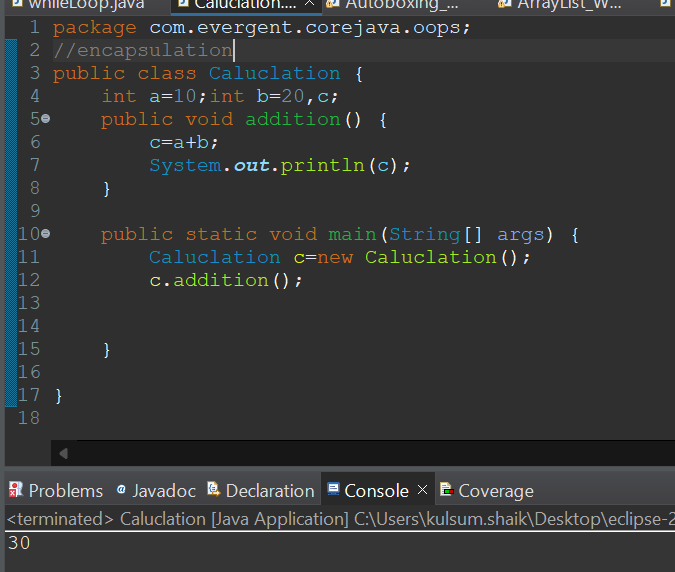
1. **Switch Cases**
2. **enum**
3. **Scanner**
4. **Object class**
5. **Event Management Application.**

**Date:07/08/24**

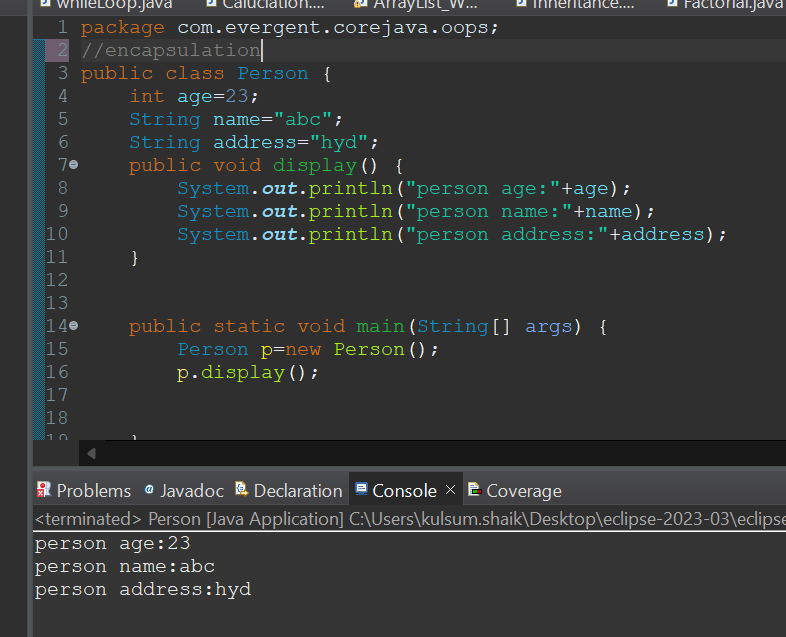
**OOps concept**

**1) Encapsulation**

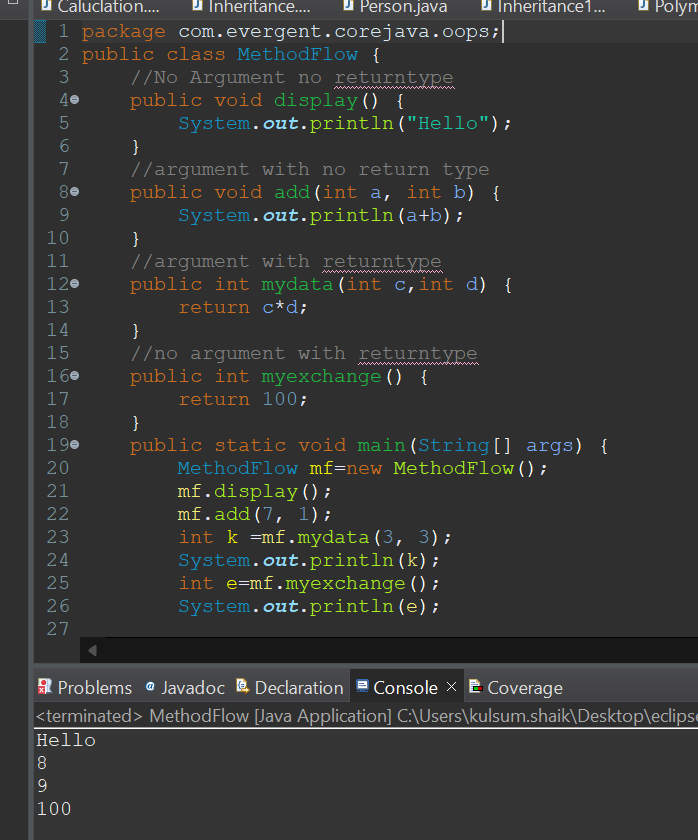
**Calculation Class**



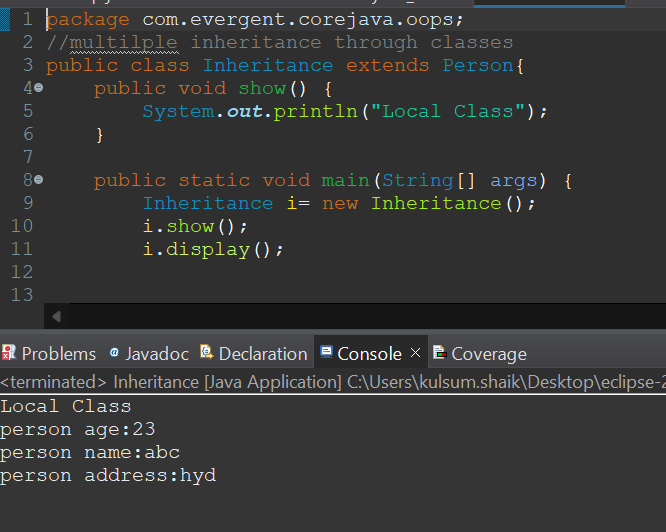
**Person Class**

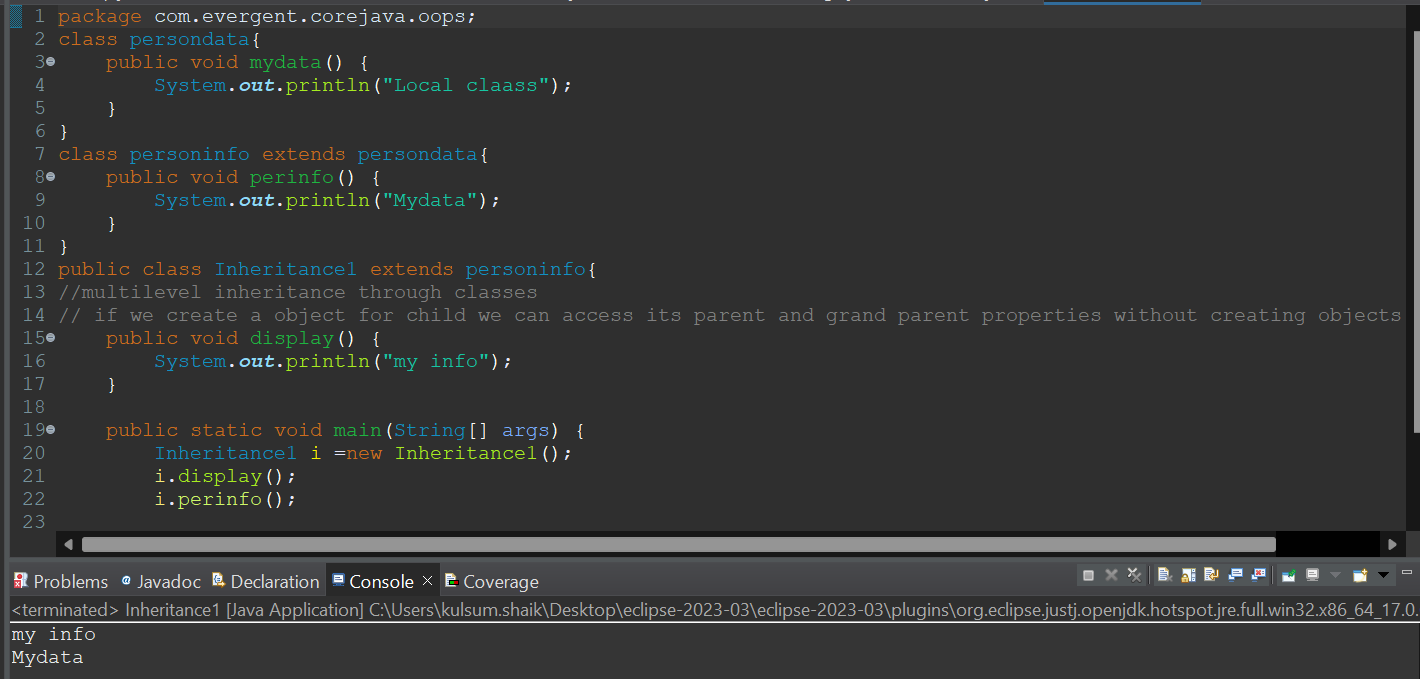


**Method flow**

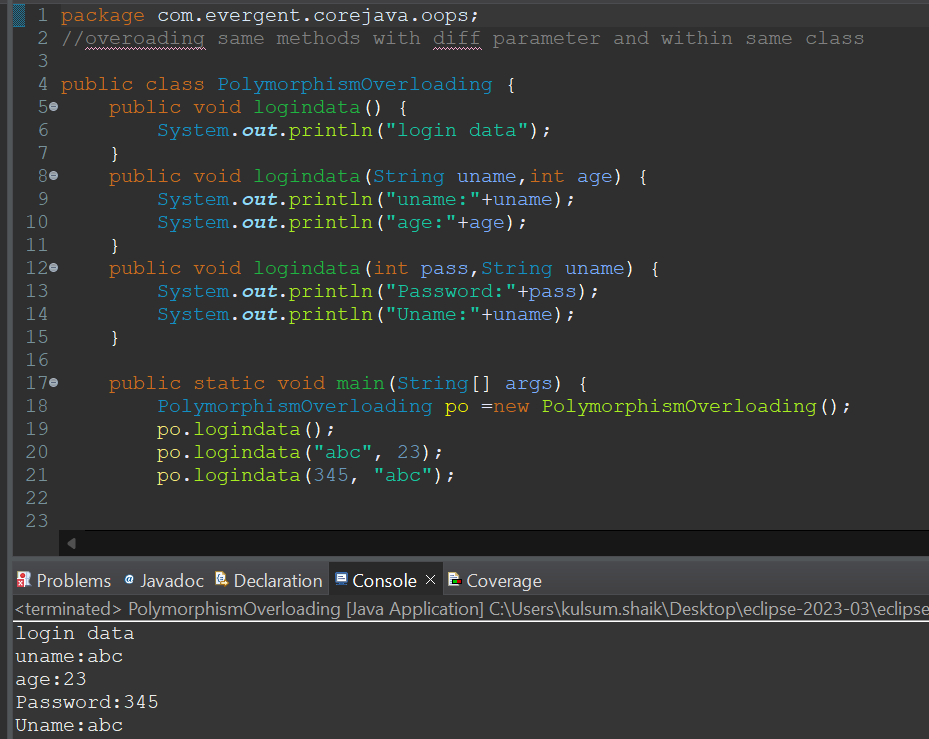


1. **About System.out.println**
2. **Inheritance**

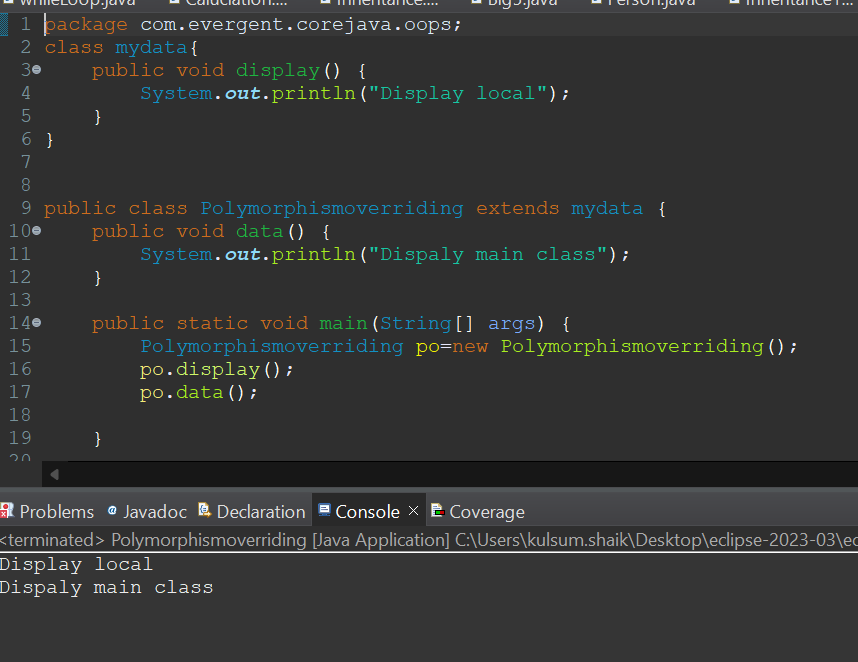




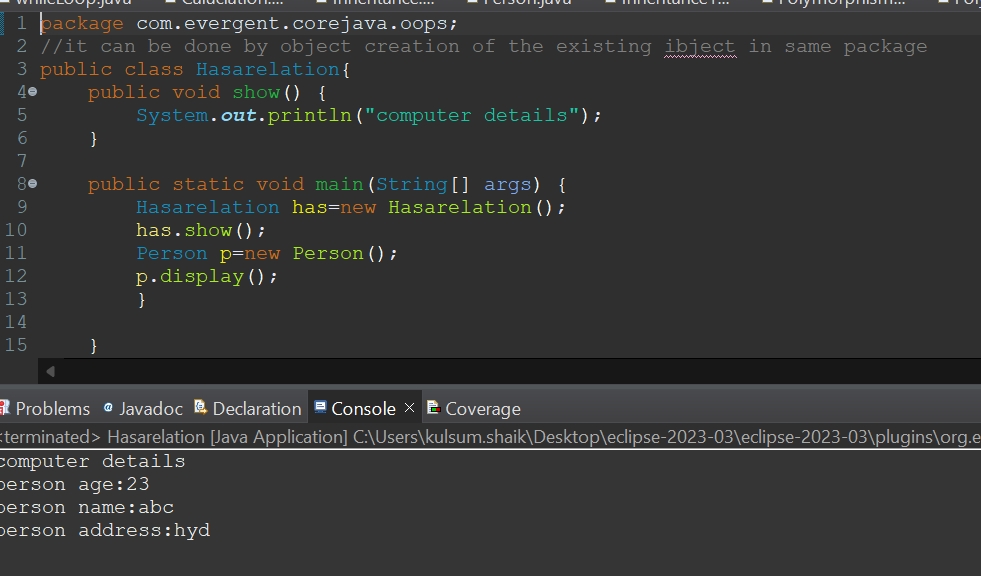
1. **Polymorphism**
2. **Overloading**



1. **Overriding**



1. **Abstract**
2. **Is\_A\_Relation**
3. **Has\_A\_Relation**

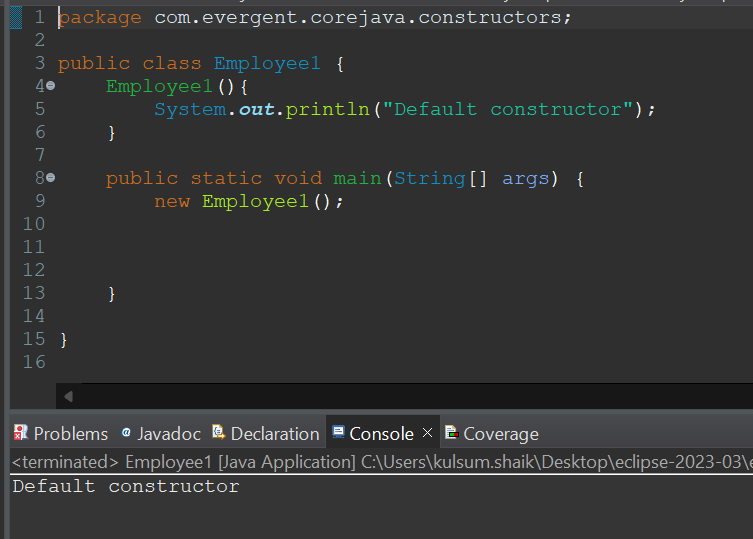


**Date:08/08/24**

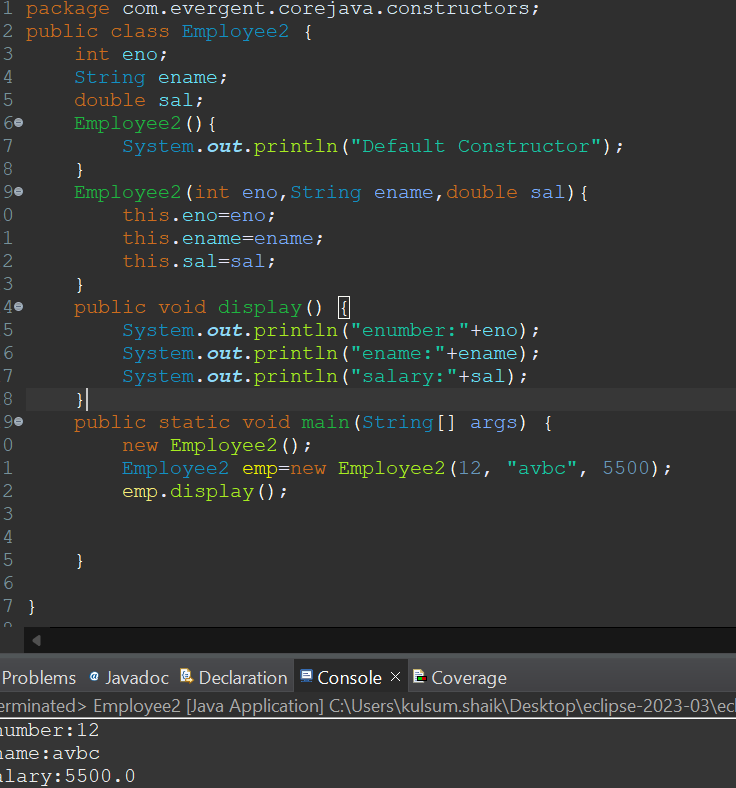
**Constructor**

**1) Types of constructors**

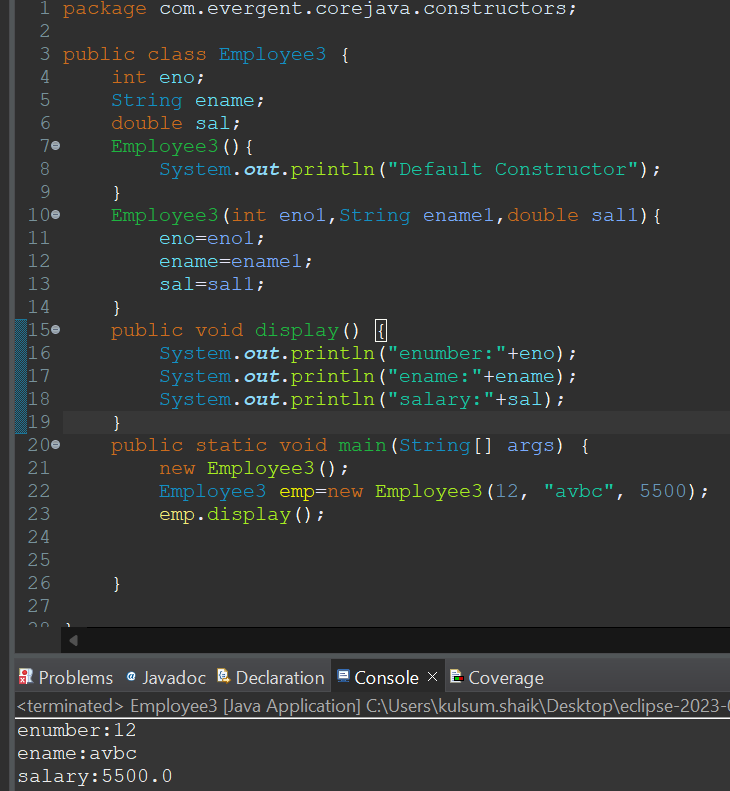
**a. Default constructor**



**b. Prameterized constructor**



1. **Accessing Constructor through object creation**



**3)Difference between constructor and method**

**4)Intializing variables using constructor**

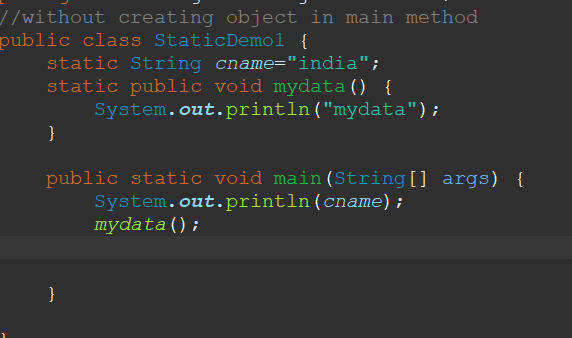
**5)this keyword super keyword**

**6)Constructor overloading**

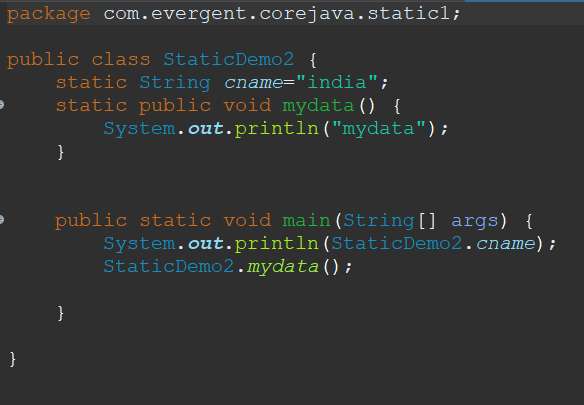
**7)Copy constructor**

**Date:09/08/2024**

1. **Static is a keyword**
2. **Static can declare variables and methods**

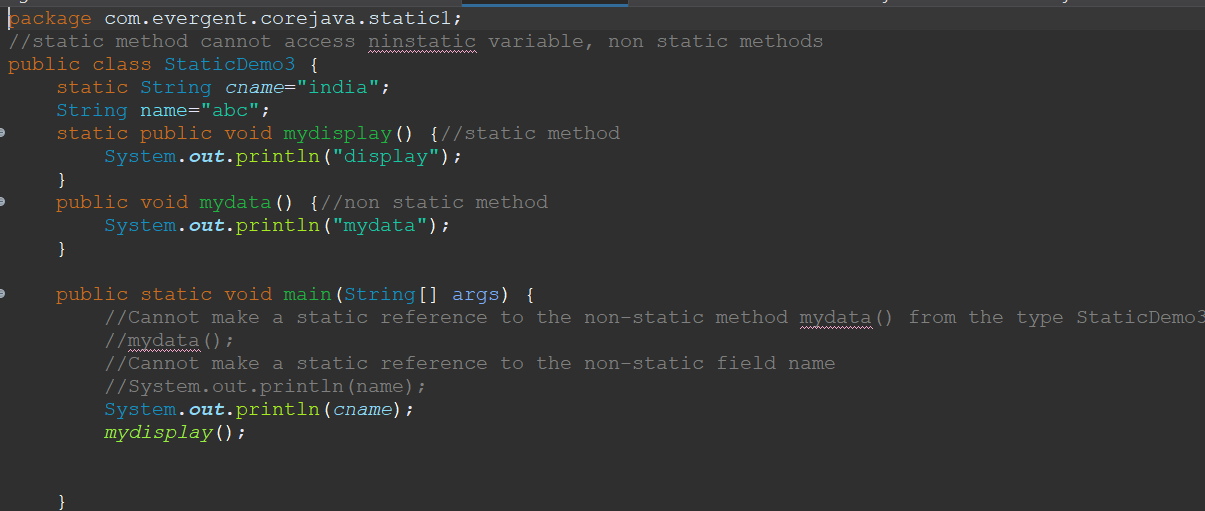


**3)Static can declare as variables and methods through classname.methodname and class name.variablename**

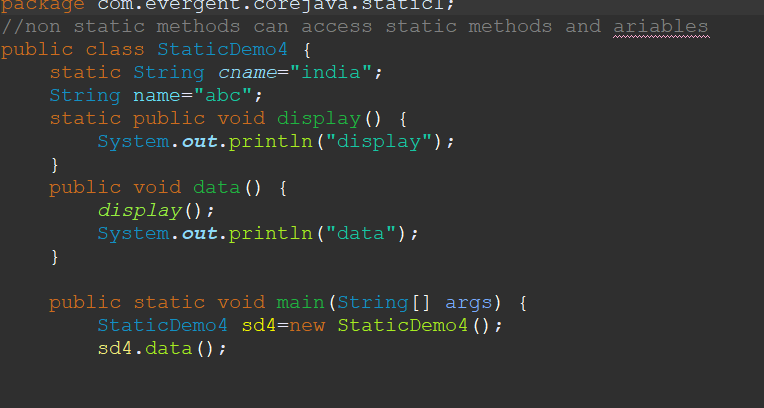


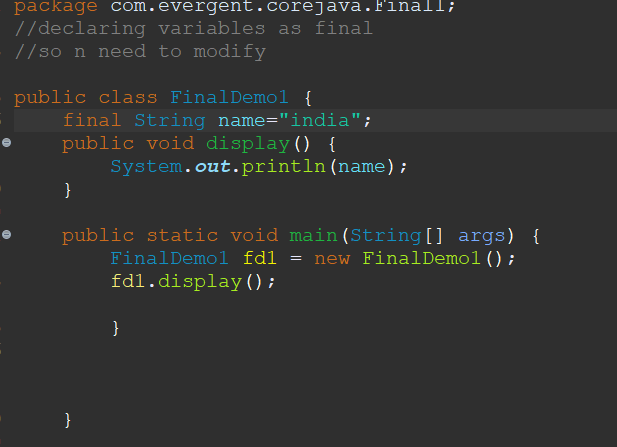
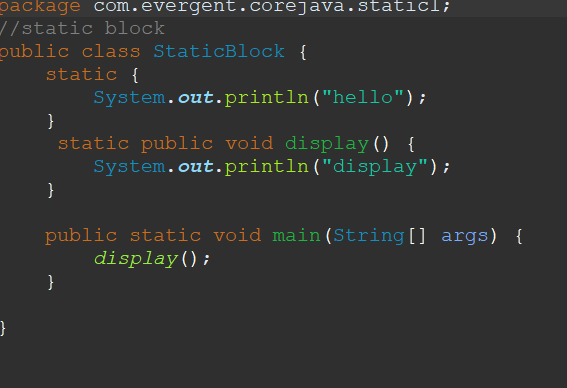
**4)Static methods can access static variables and methods**

**5)Static methods cannot access non static methods and variables**

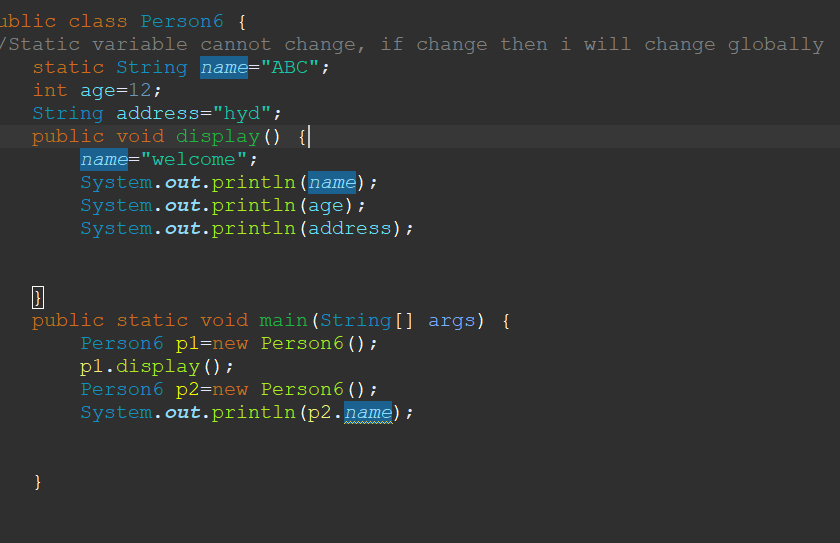


**6)Non static methods can acess static methods and variables**



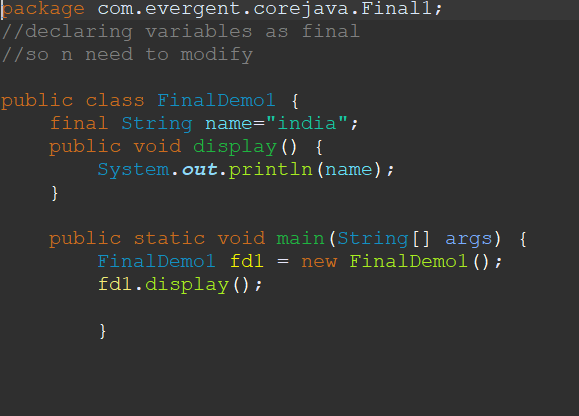
**Static block**

**If we change static varible locally,it effects globally**

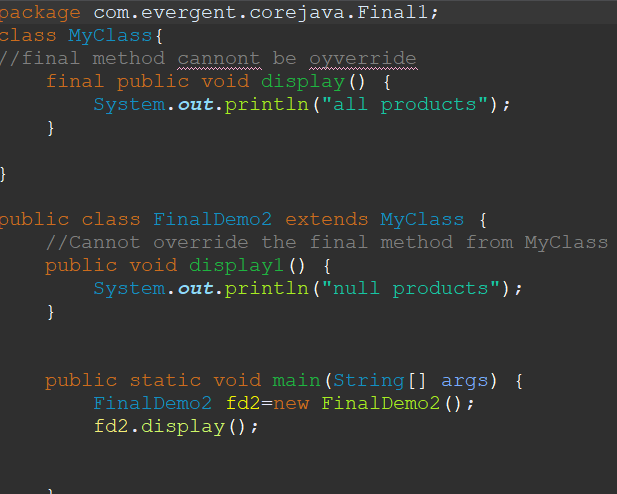


**FINAL keyword**

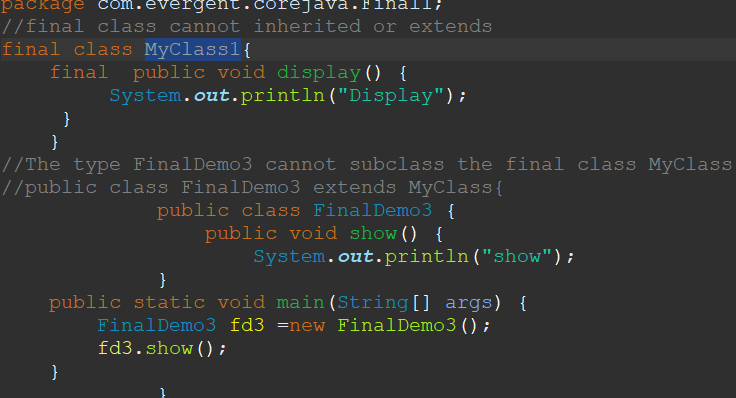
**1)Final as a variable -we cannot modify variable**



1. **Final methods-we cannot override method (we can inherit class but cannot override method)**



**Final class-we cannont even inherit the classs using extends keyword**



**STRINGS**

**String class String Buffer**

(Disadva) (advantage) (advan) (disadv)

Immutable non synchronization mutable synchron

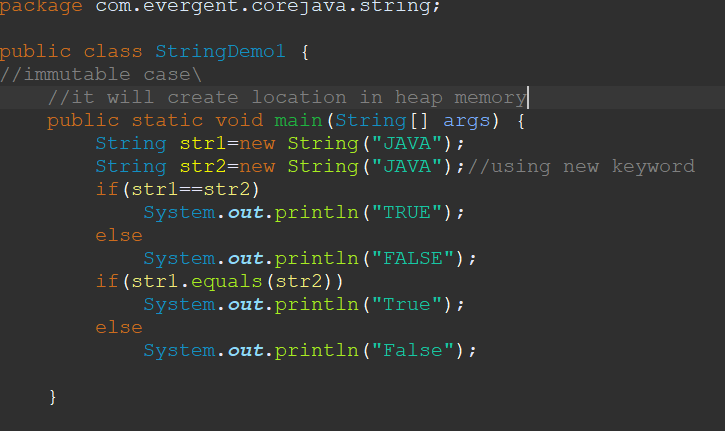
String BUILDER(JDK1.5)

**12/08**

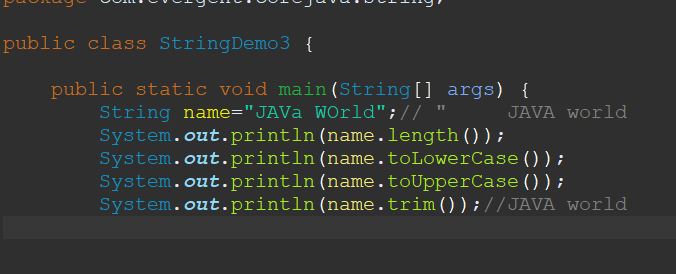
**1)String Class:**

**String is final class**

**String is immutable**



**String class having methods**



**String class methods are non-Synchronised**

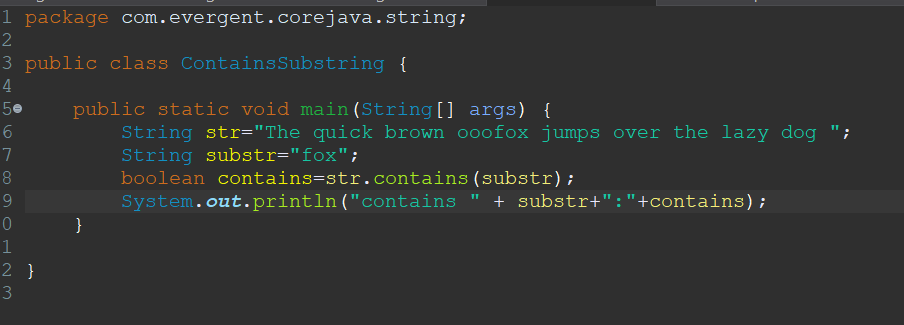
1. **String Buffer**

**String as final**

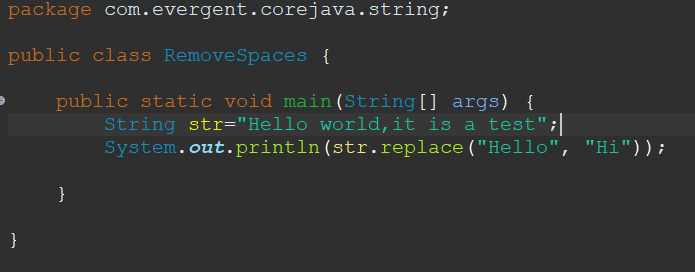
**String buffer is mutable**

**String Buffer having methods**

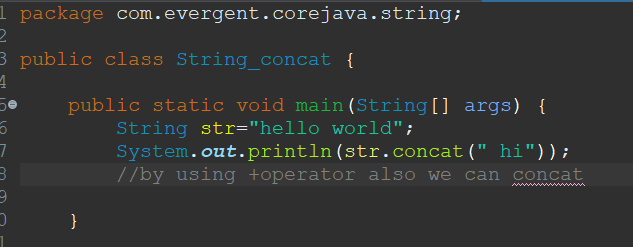
**Program1**



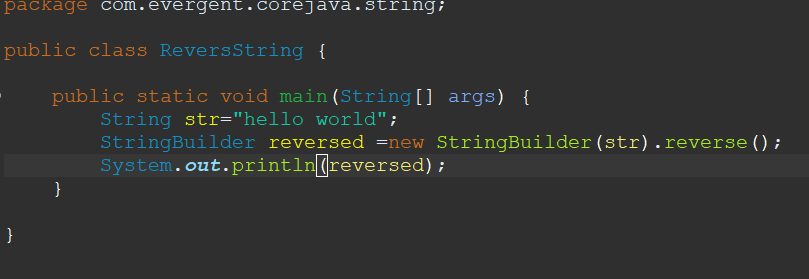
**Proram2**



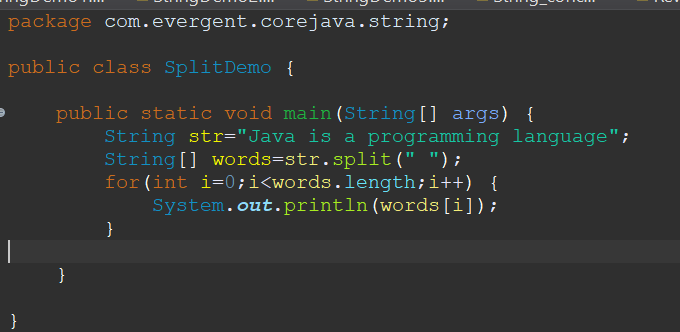
**Program3**



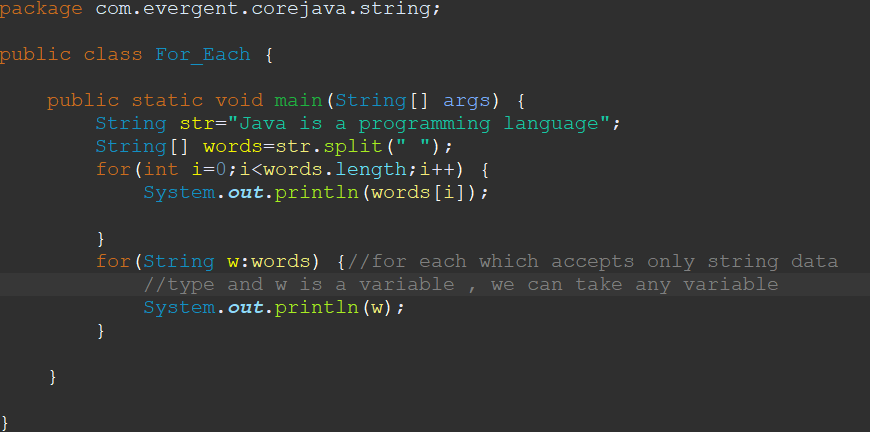
**Program 4**



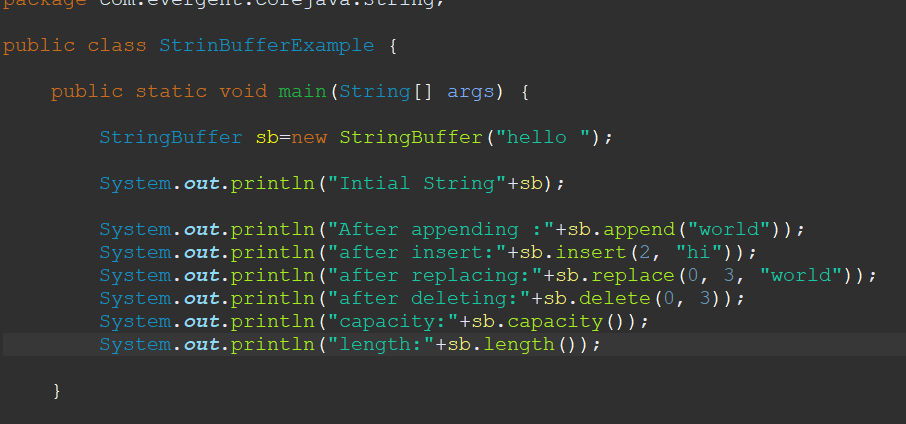
**Program5**



**Program 6**



**Program 7**



**String Buffer are Synchronization**

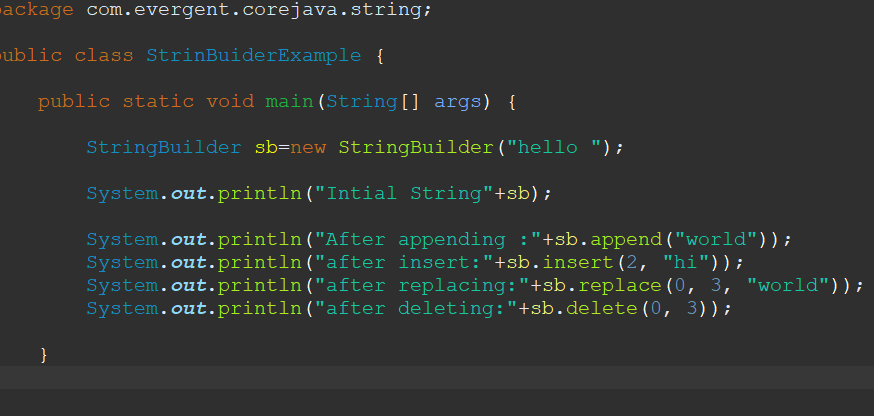
1. **String Builder**

**String as final**

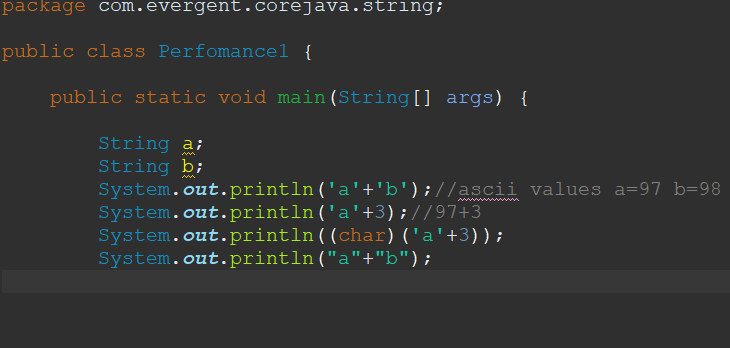
**String Builder is mutable**

**String Builder having methods**

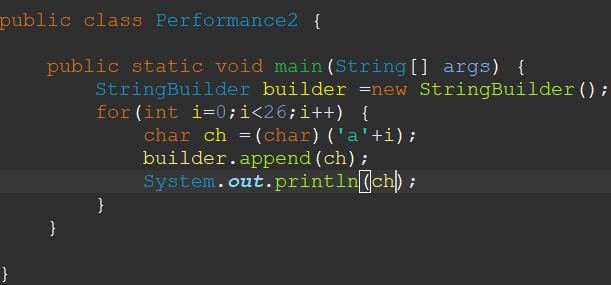
**Program 1**



**Program 2**



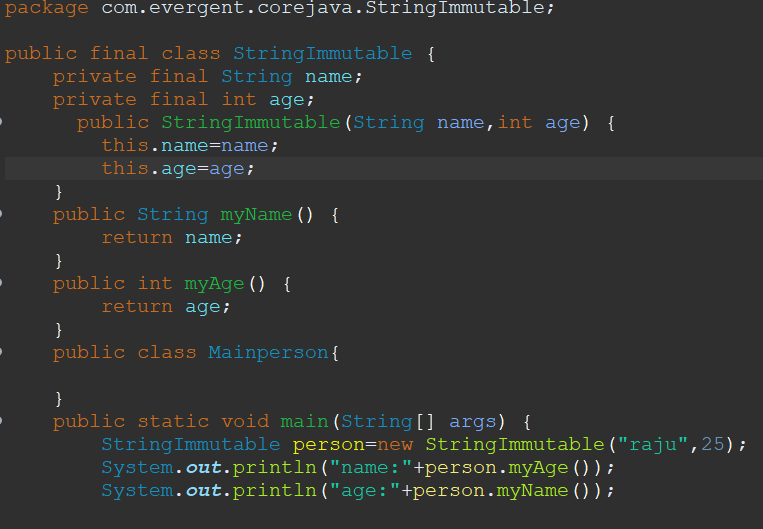
**Program 3**



**String Builder are non Synchronization**

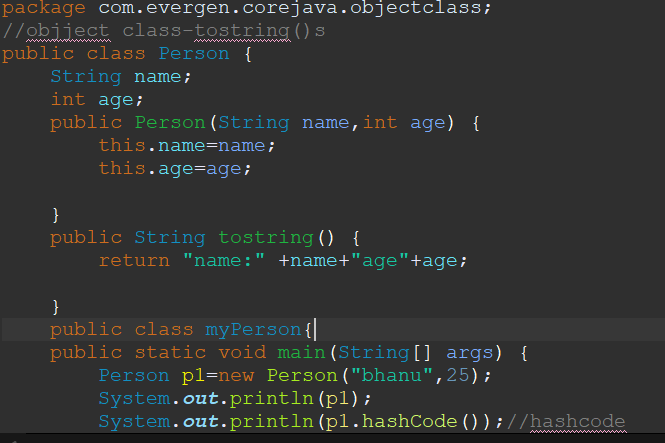
**Can we make class as Immutable?**

**Yes, when we declare class as final and declaring variables final and private**



**Object class-**

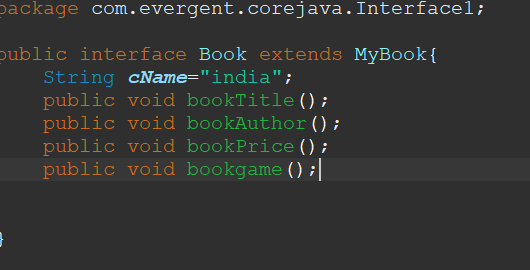
1. **tostring()**
2. **Hash code**

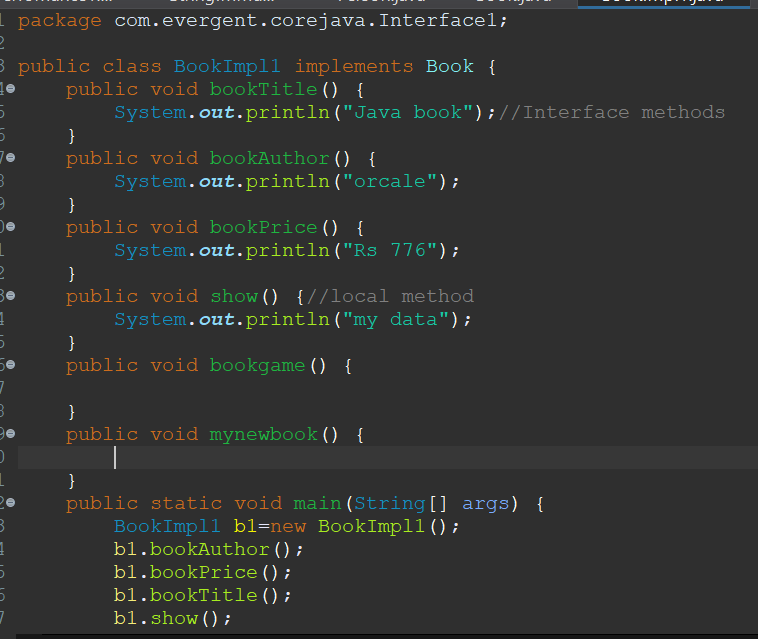


**INTERFACE**

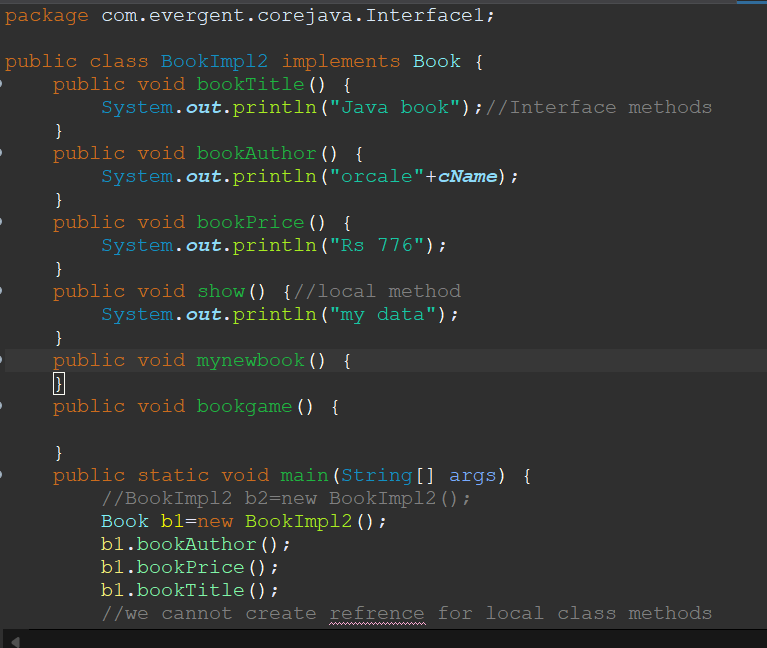
1. **interface is a keyword**
2. **We can declare only method signature but, not implementation in interface**

**THE BELOW PICTURE CAN COVER POINTS OF 1 2 3 4 6 8**





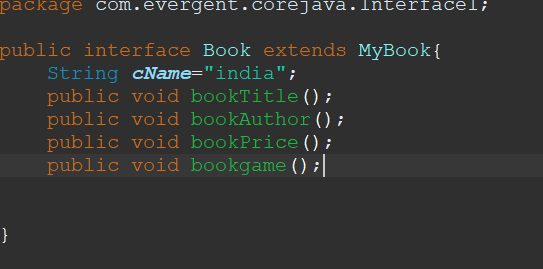
1. **By default all interface methods are abstract**
2. **If any class implements interface then the class should override all methods of interface or else its shows compile time error.**
3. **We cannot create object to interface , but we can create a reference to interface**



1. **Java support multiple inheritance through interfaces**



1. **One class can have more than one interfaces**
2. **One interface can extends other interface.**



**Use of interface in software industry:**

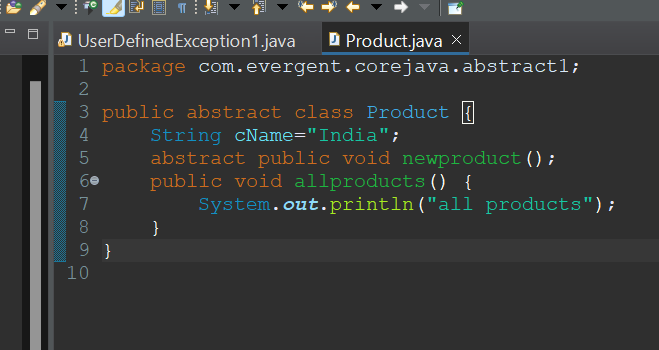
**Its acts as templates for the project**

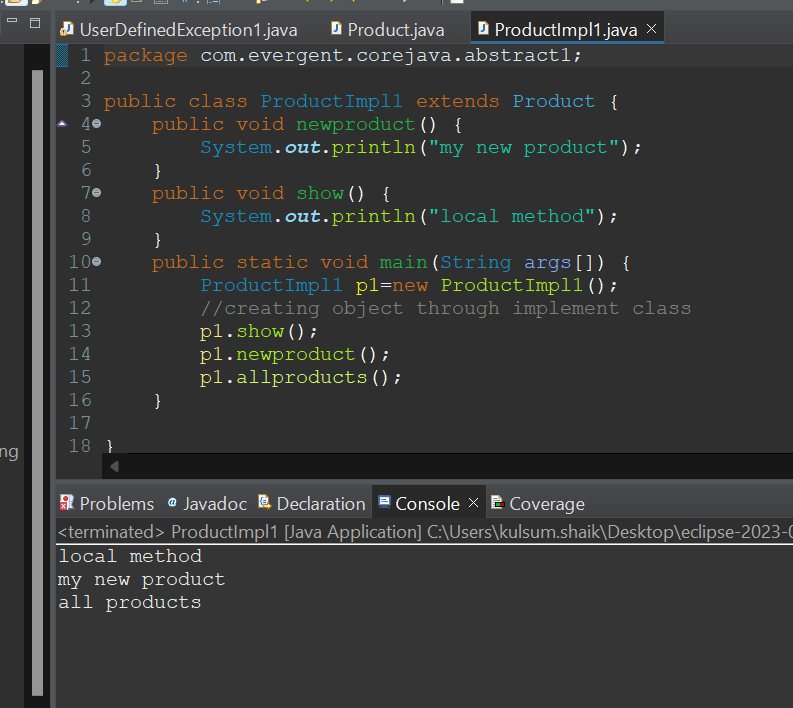
**And everyone should follow the same structure**

**ABSTRACT**

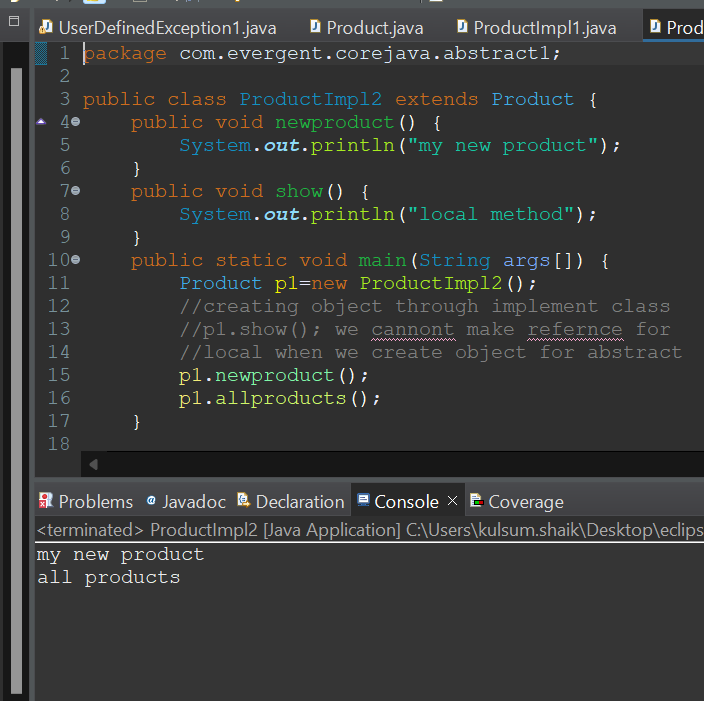
1. **Abstract is a keyword**
2. **Abstract class having abstract methods nd concrete methods**
3. **If any class having one abstarct method that class should be declare as a abstract keyword otherwise its shows compile time error**
4. **If any class extends abstract class then that class should be override all abstract methods or else its shows comile time error**
5. **We cannot create object to abstract class**
6. **But we can create interface to abstract class**

**Program1**

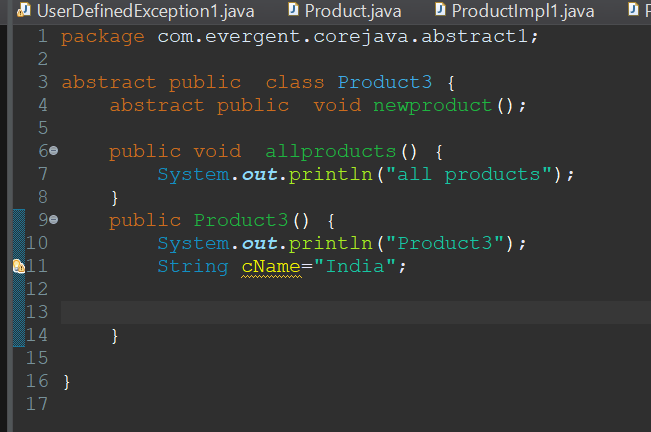




Program2



**Program3**





**19/08/24**

**EXCEPTION HANDLING**

1. **exception handling is mechanism**
2. **Exceptions are inbuilt mechanism of java**
3. **All exception are executed while abnormal conditions only**
4. **Normal flow it wont execute any exceptions**
5. **Once any exceptions are occurring in java then remains lines of code is unreadable.**
6. **Java.lang.Throwble is super class for exception and error**
7. **There are two types of exception in java**
8. **checked exception**
9. **Unchecked exception**
10. **all checked exceptions are compile time exceptions**
11. **All unchecked exceptions are runtime exceptions**
12. **there are 5 keywords**

**try()**

**Catch()**

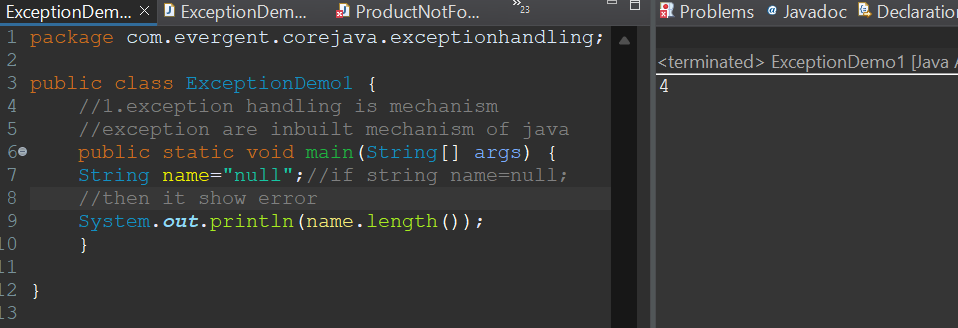
**Finally()**

**Throws()**

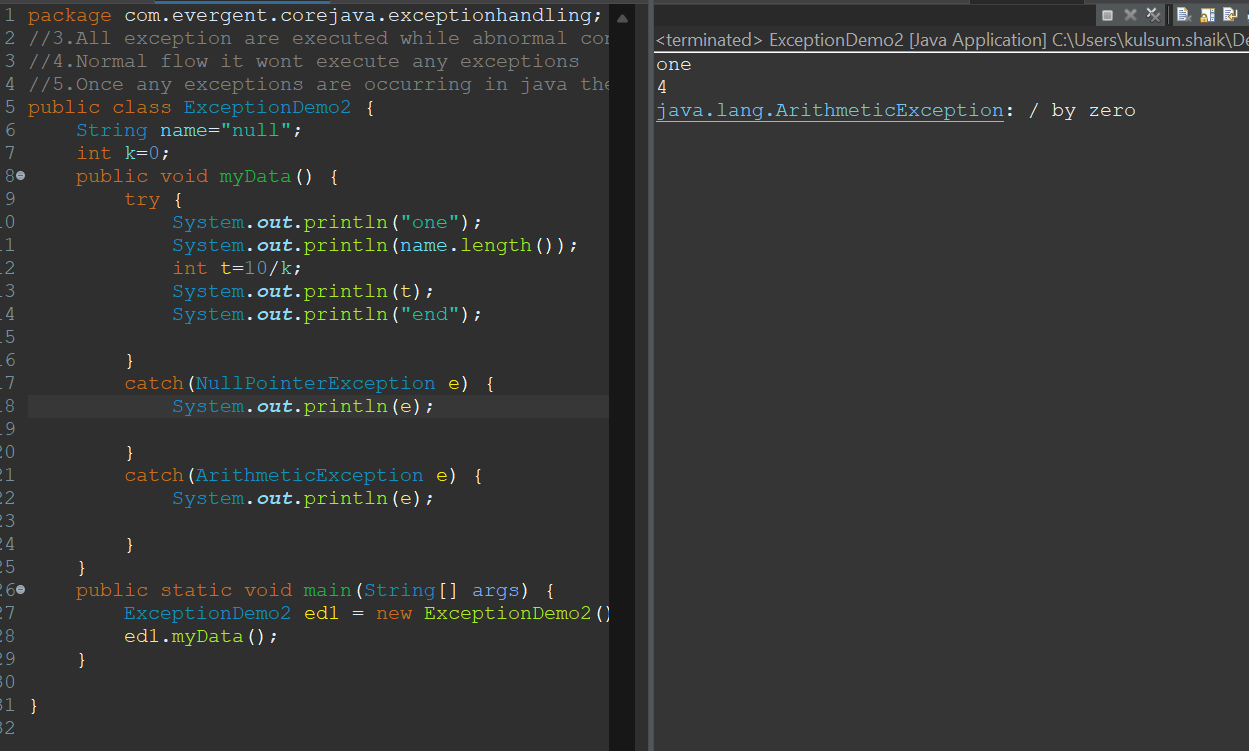
**Throw()**

1. **try is for business logic**
2. **Catch is for handling exceptions**
3. **Finally is block,if exceptions is occur or not finally block will executed**
4. **Throws an exception will be executed method by method**
5. **Throw is for runtime exceptions and will call predefined exceptions or not user defined exceptions**
6. **Try followed by either catch or finally block**
7. **We should follow exceptions hierarchy**
8. **We can create our own exceptions(user defined)**
9. **Our own exceptions extends exception or runtime.**
10. **All exceptions classes are in to java.lang package**
11. **There is two exception in class developer should be handle one after one**
12. **Developer cant handle errors**

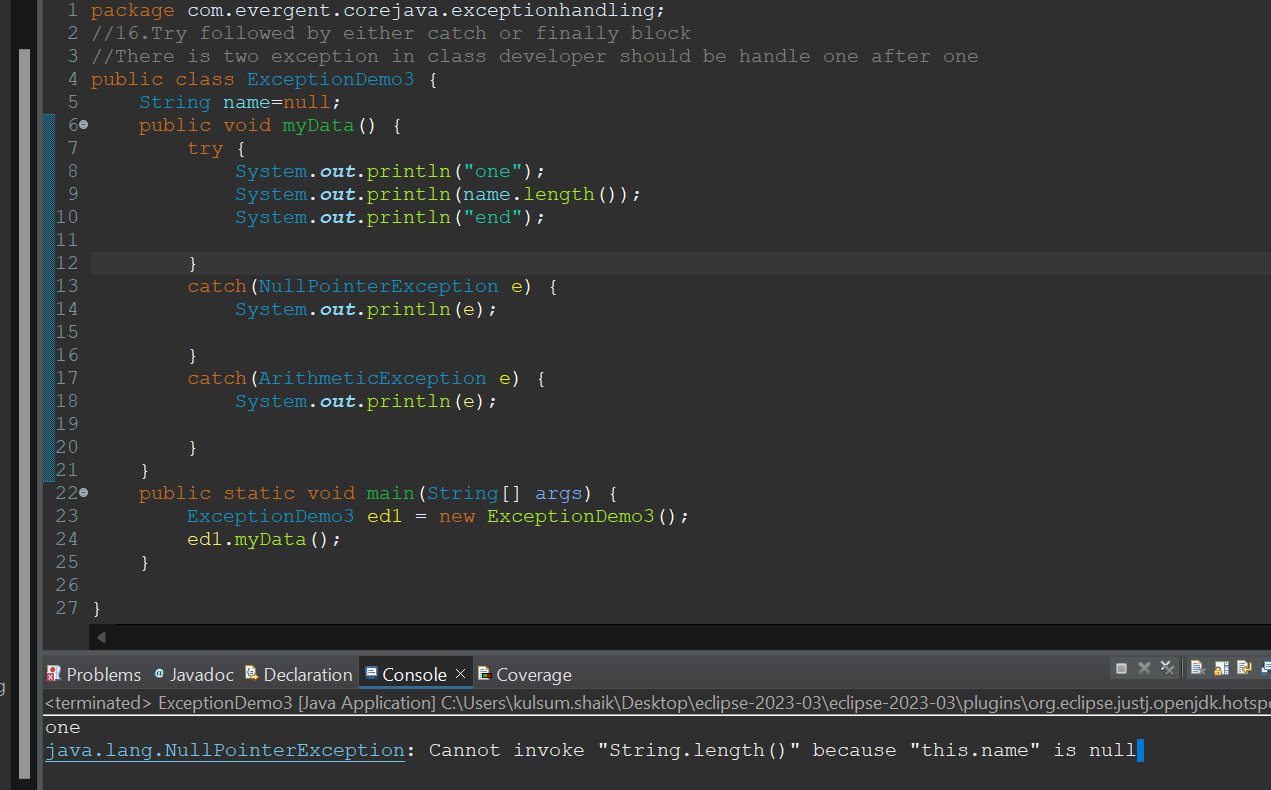
**Program1**



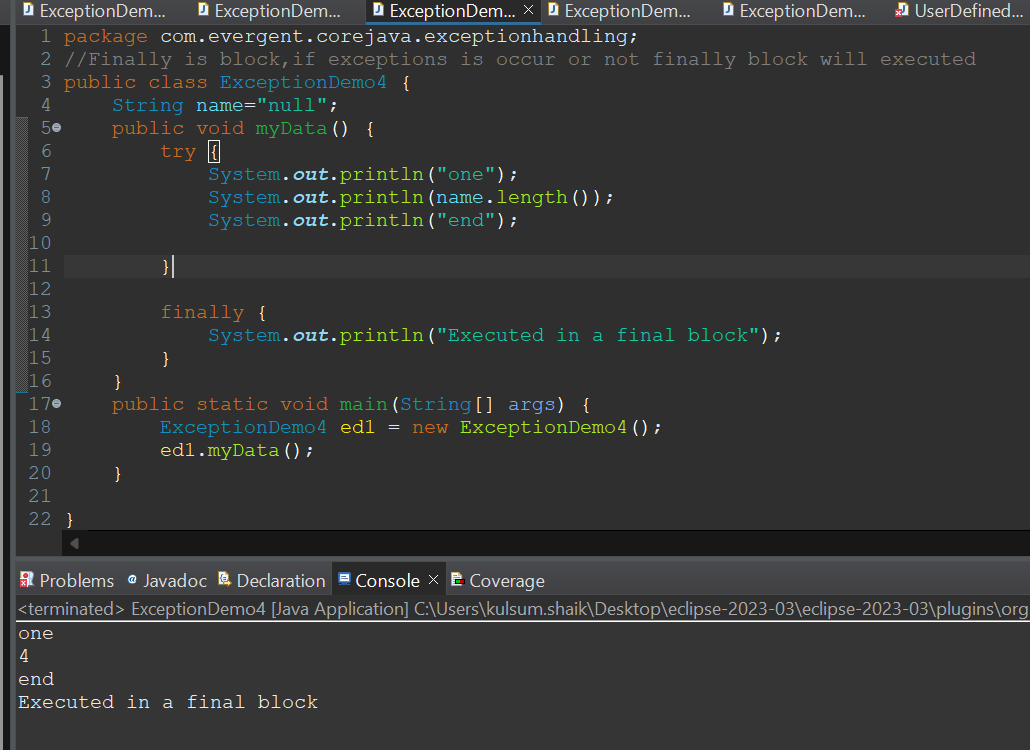
**Program2**



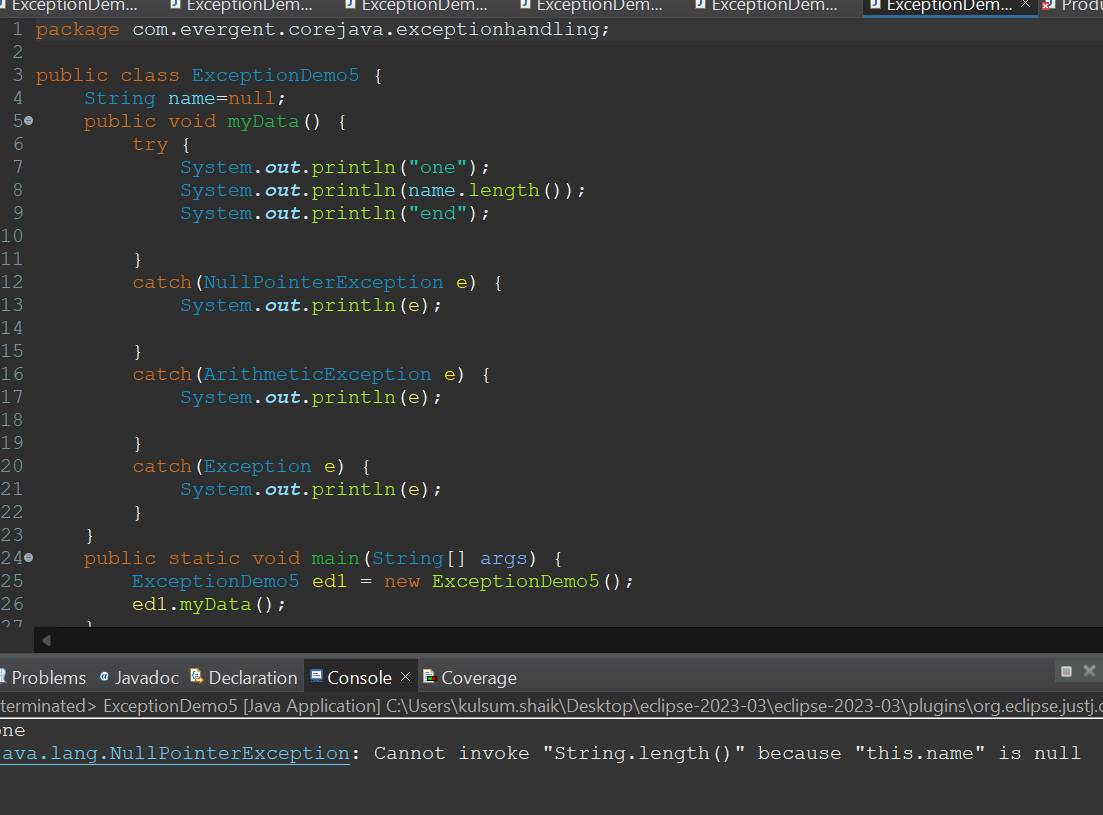
**Program3**



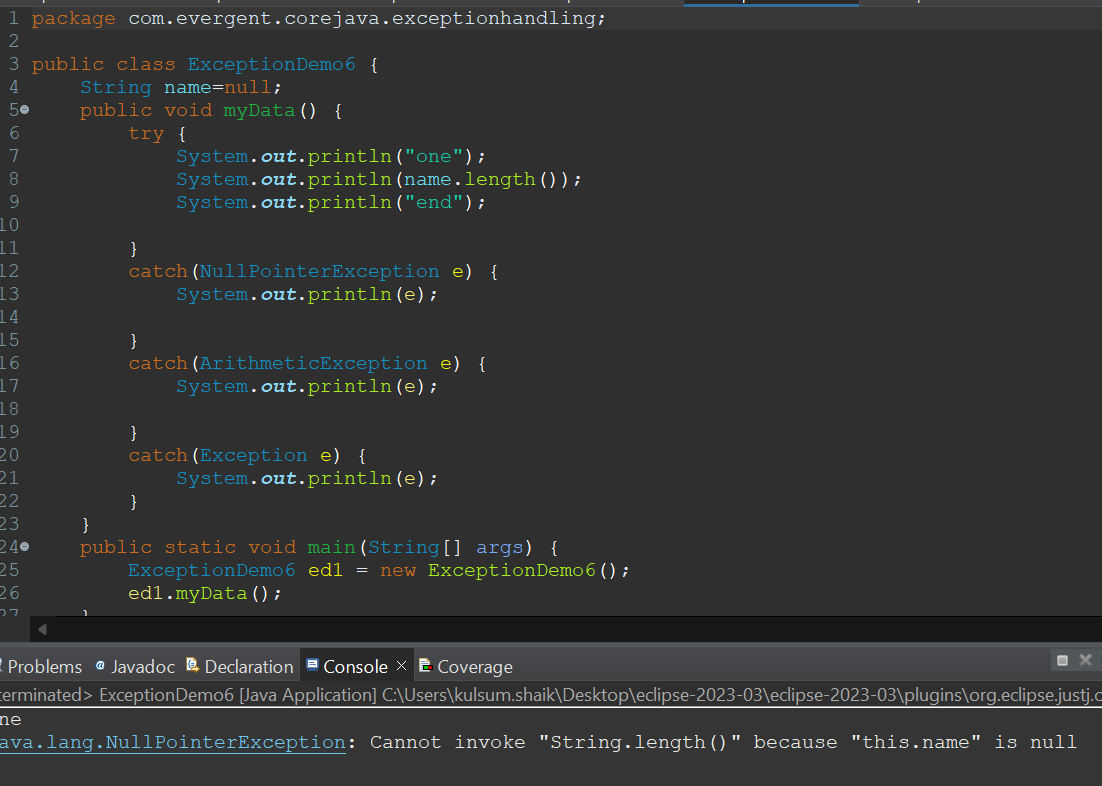
**Program4**



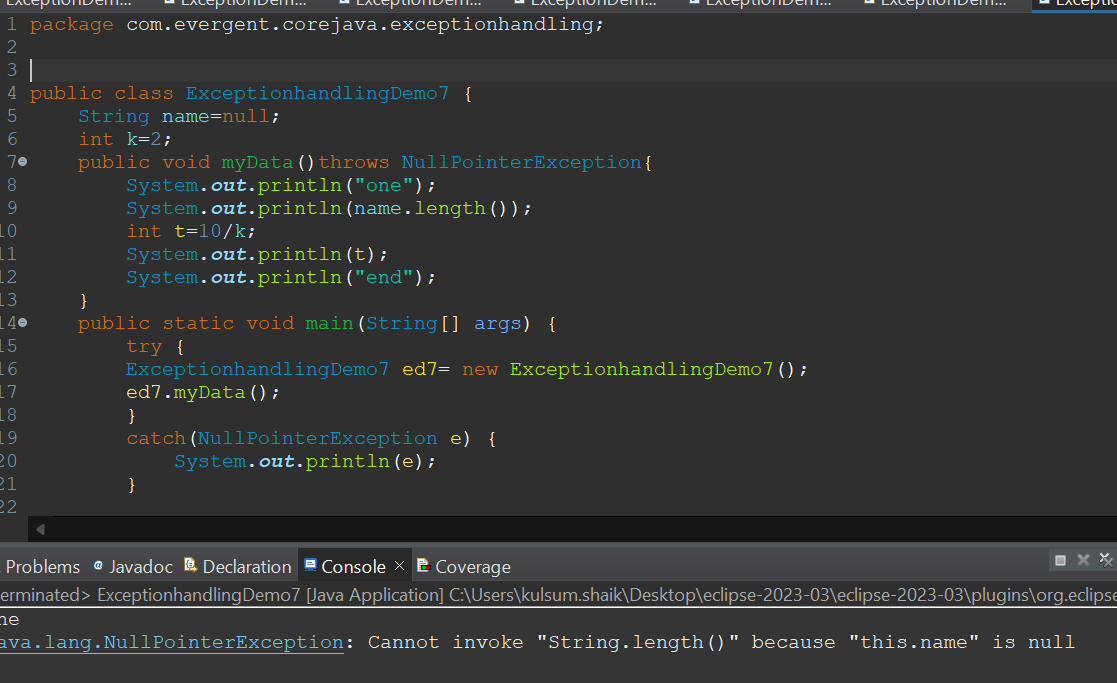
**Program5**



**Program6**

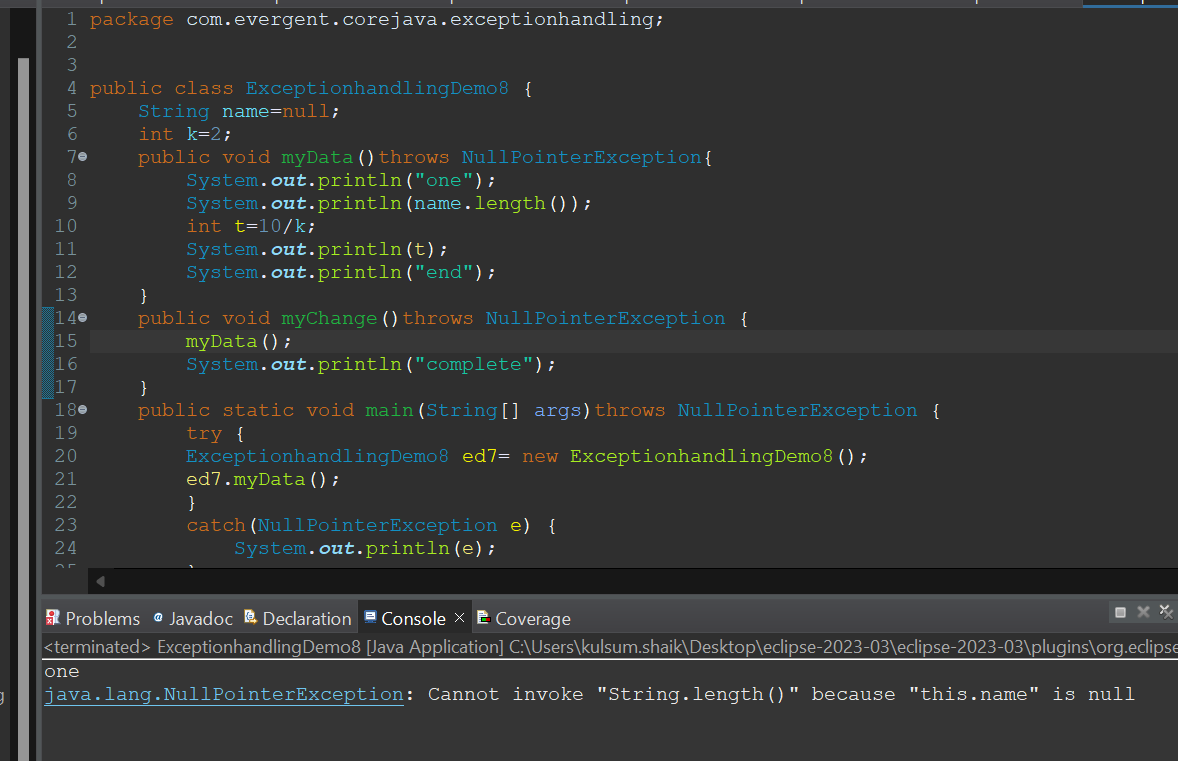


**Program7**

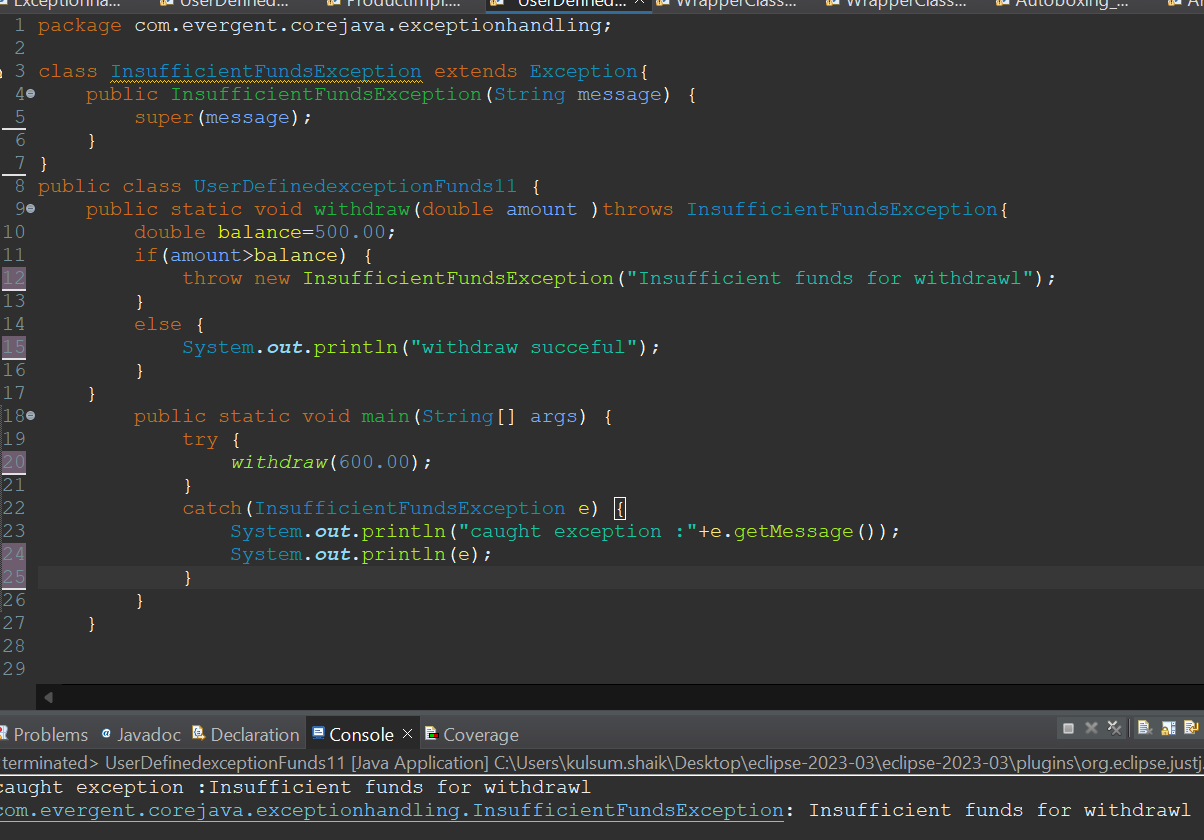


**Program8**

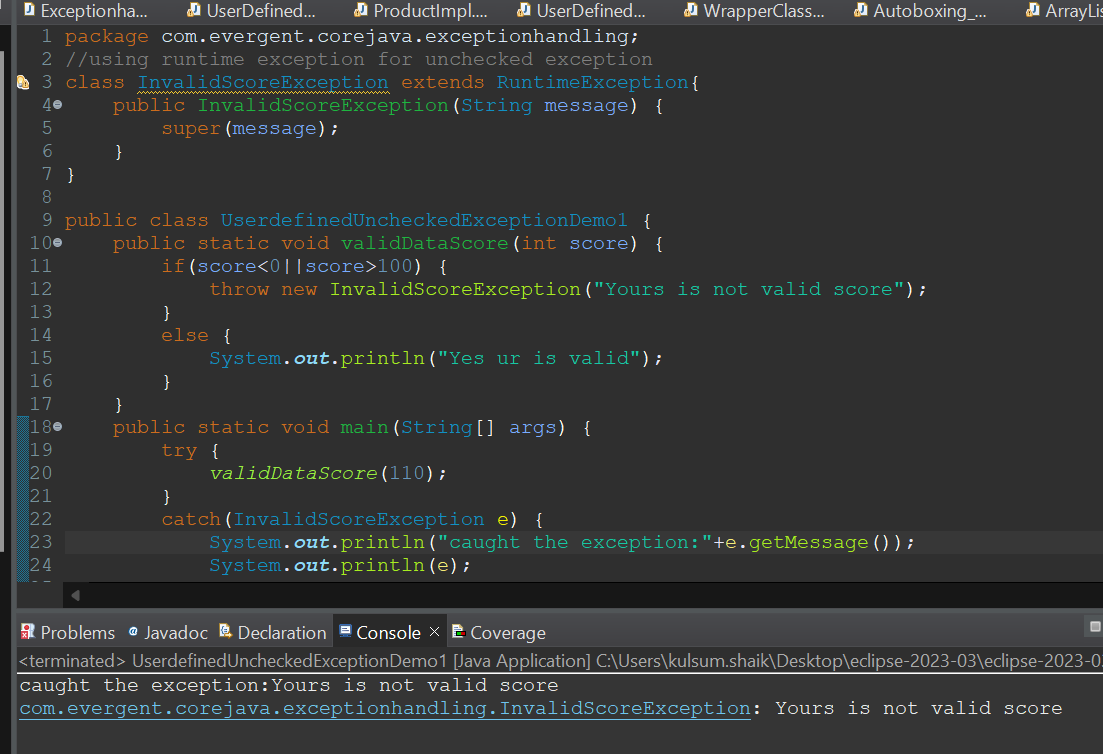
**(using throws keyword)**



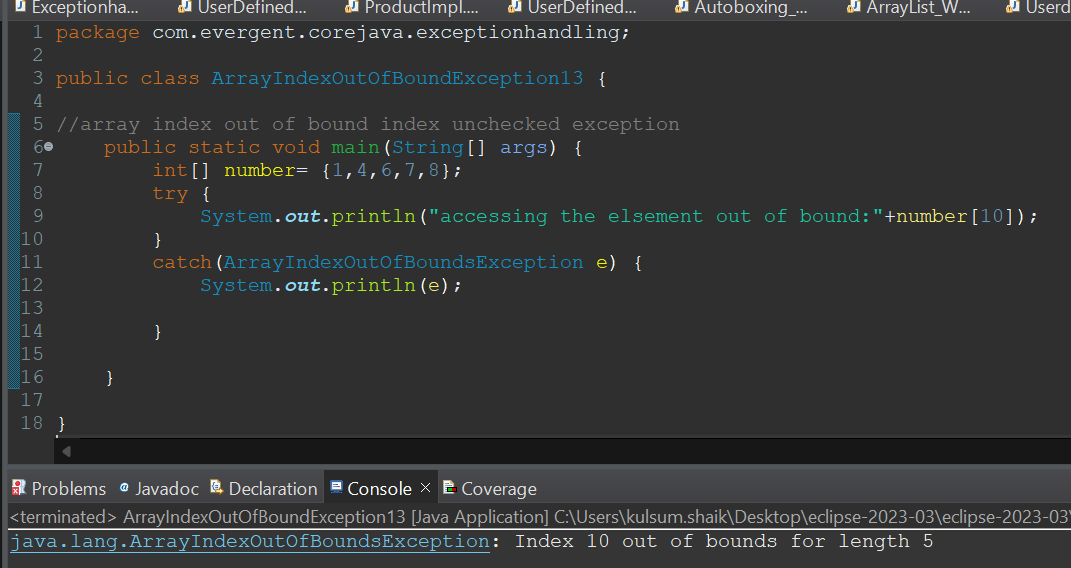
Program9



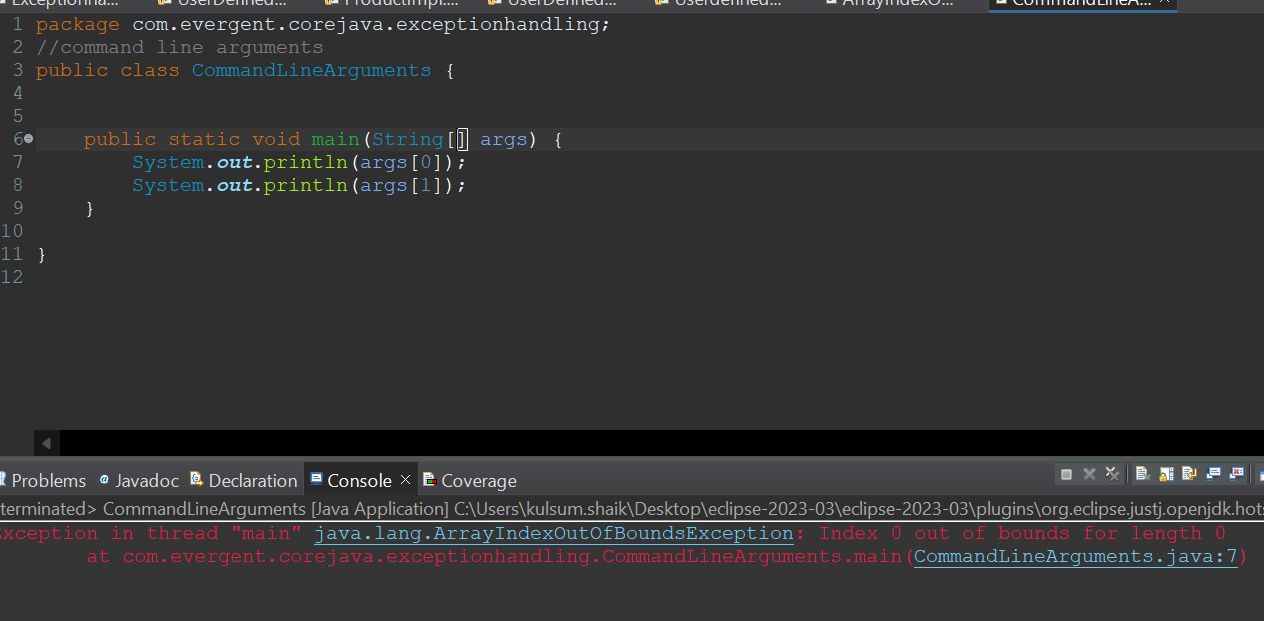
Program 10



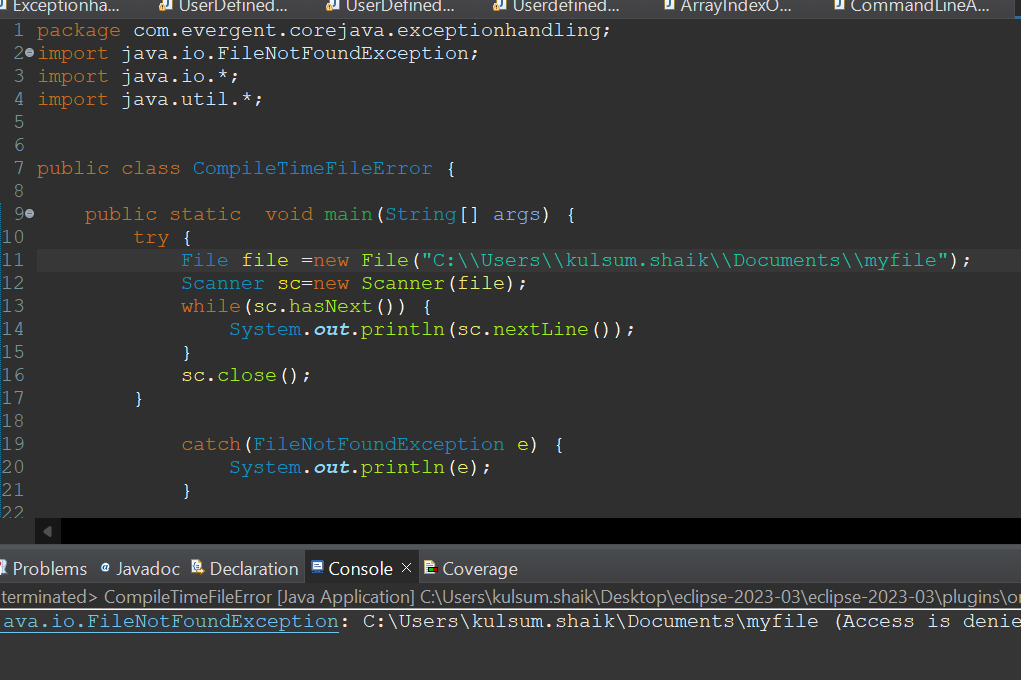
Program 11



Program12



Program13

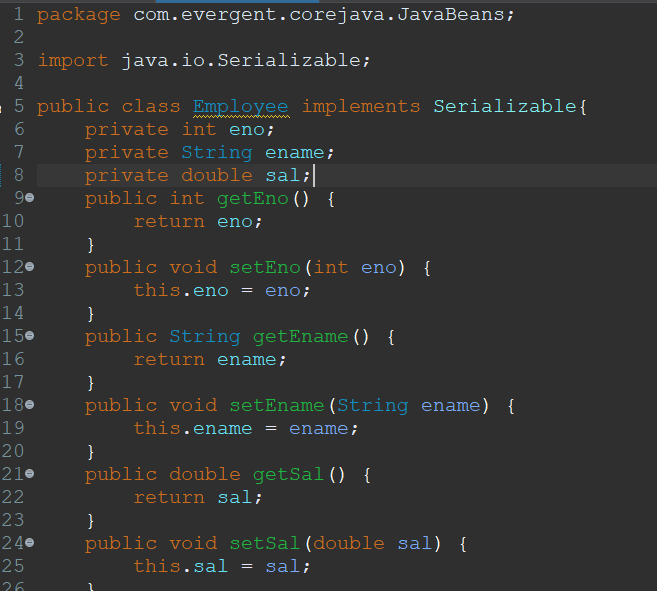


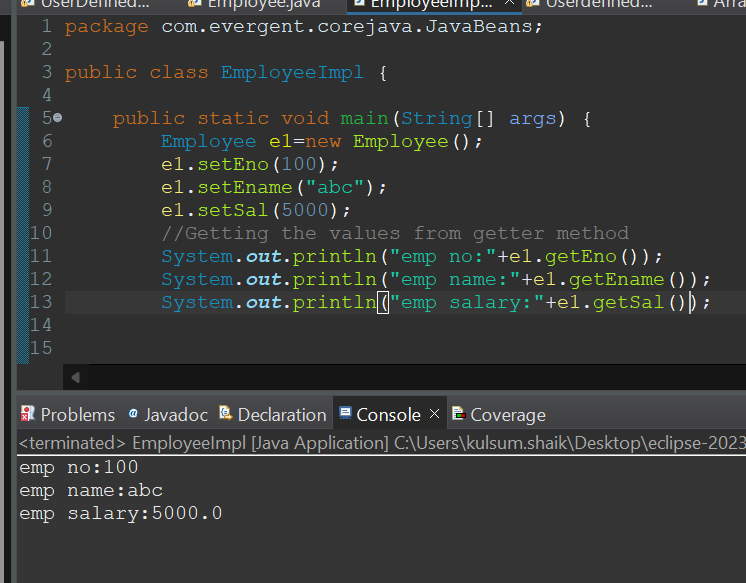
**JAVA BEANS**  
1)JAVA bean is lightweight

1. java bean is a mechanism
2. All attributes are private and get/set methods are public implements java.io.serializable interface
3. We can achieve tightly encapsulation through java beans

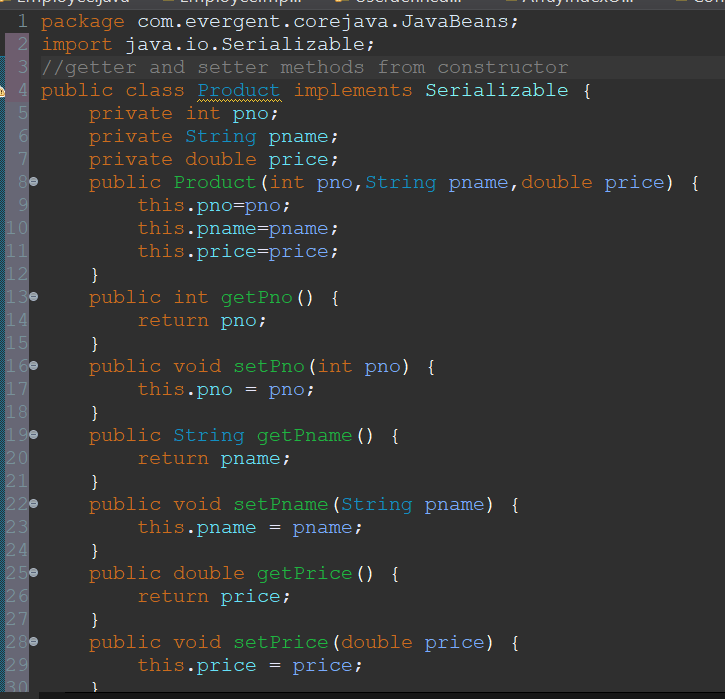
Program 1

Intializing getter and setter methods





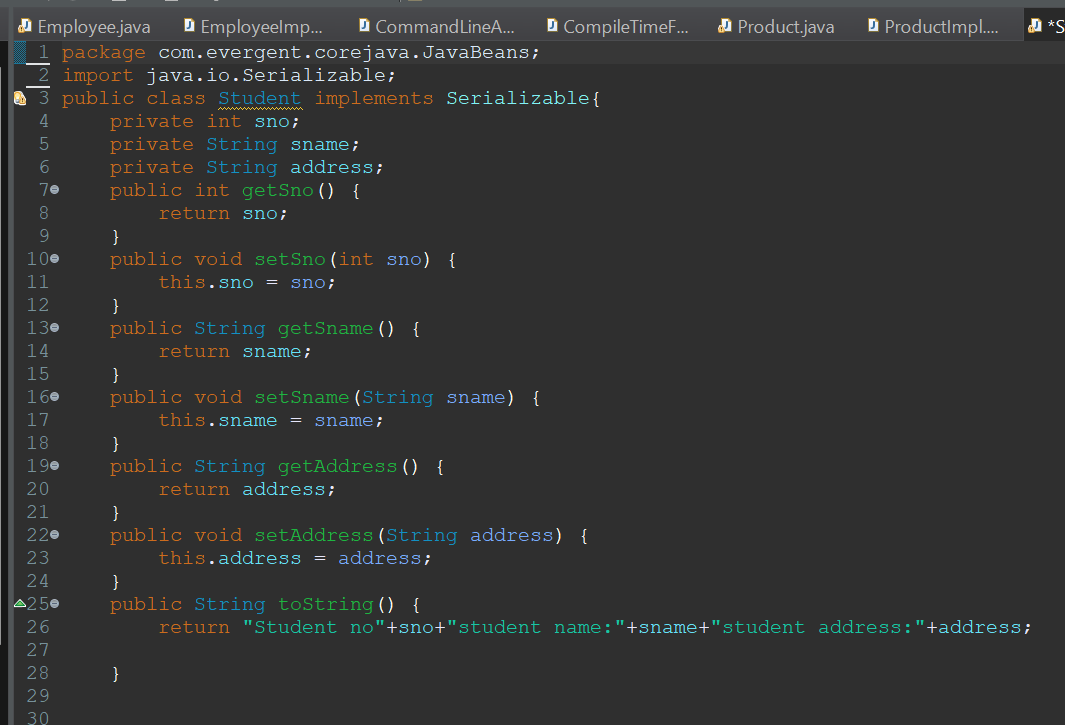
Program2

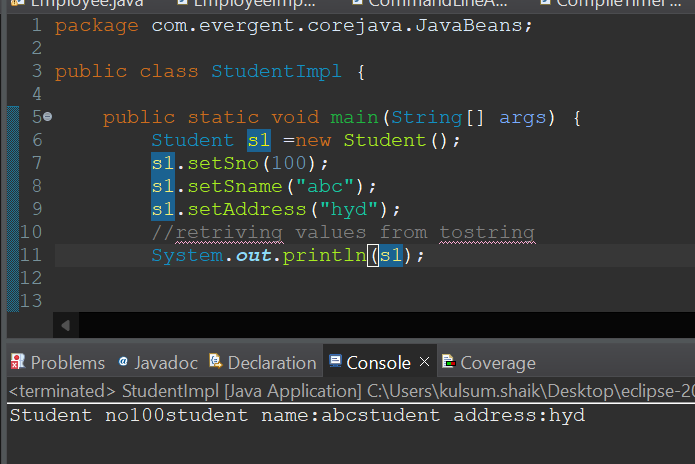




Program 3

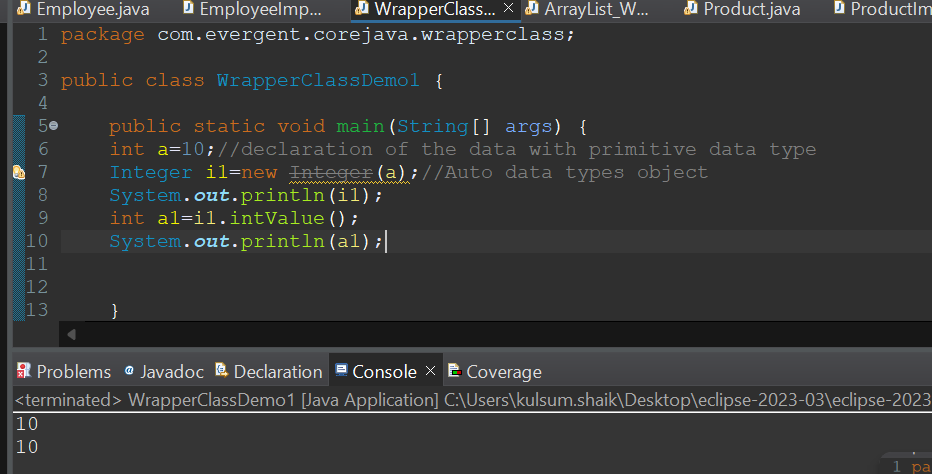
Using to string



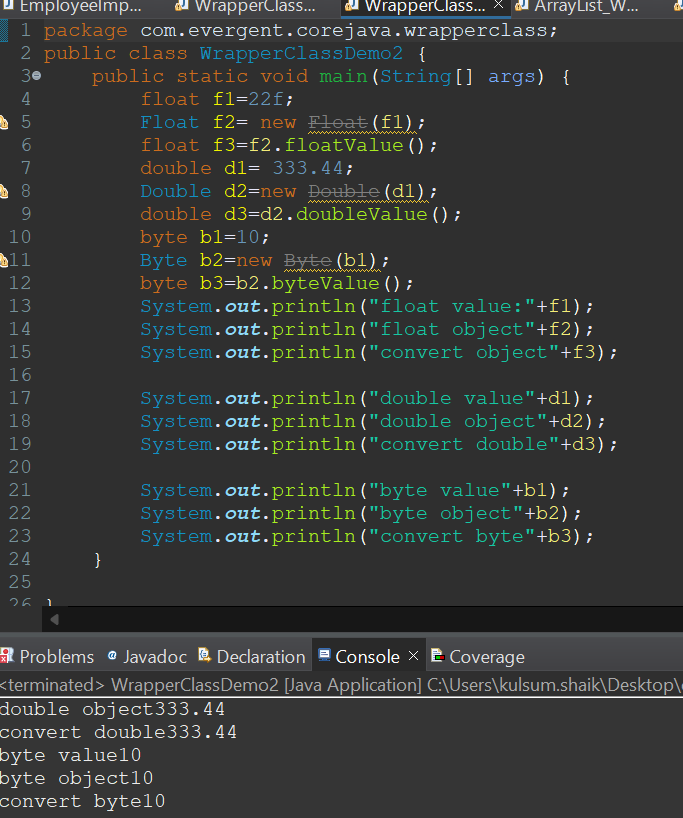


WRAPPER CLASSES

Program 1



Program 2



Program 3

