**VERSION 1.0**

**INDEX**

**Date:5/08/2024**

**Day-1**

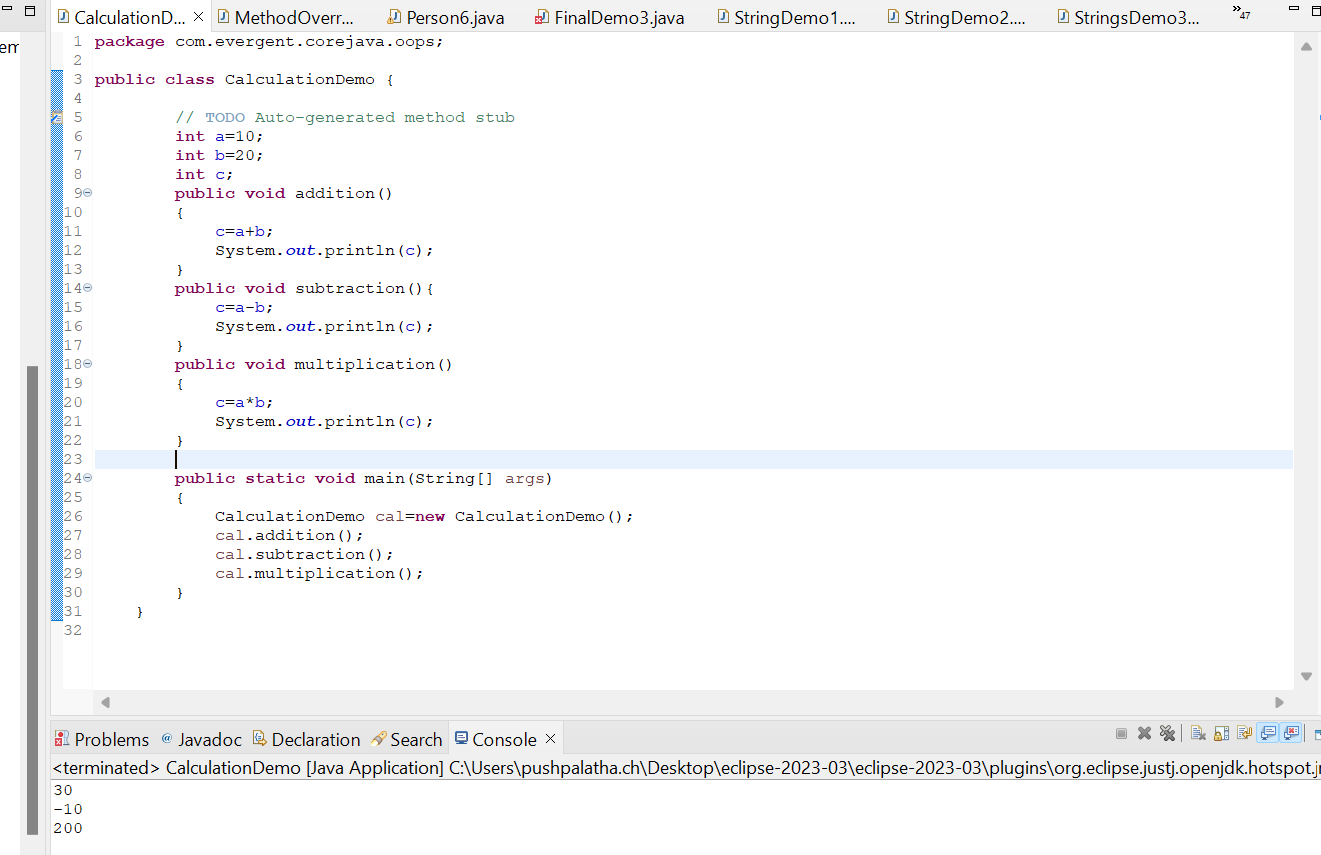
1.languages and applications

2.Java Features

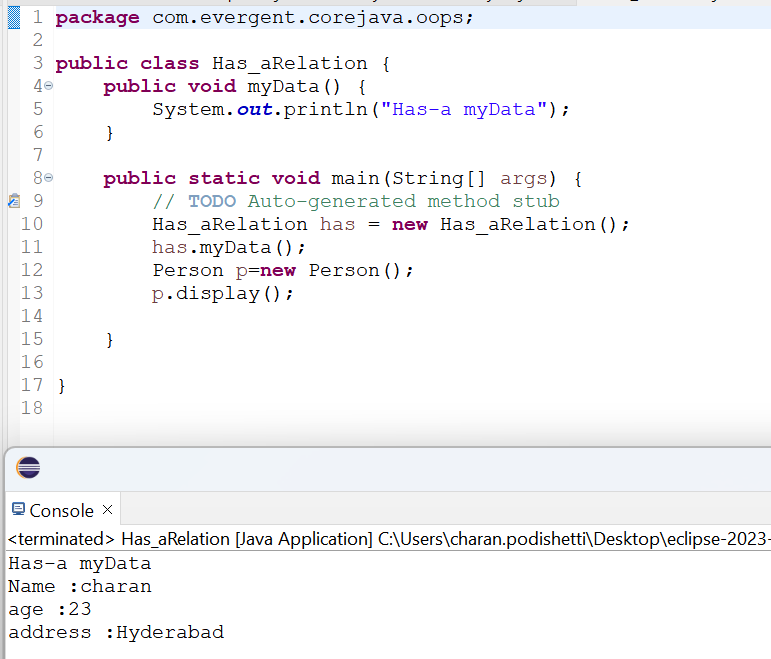
a.Why Java is platform Independent

b.oops

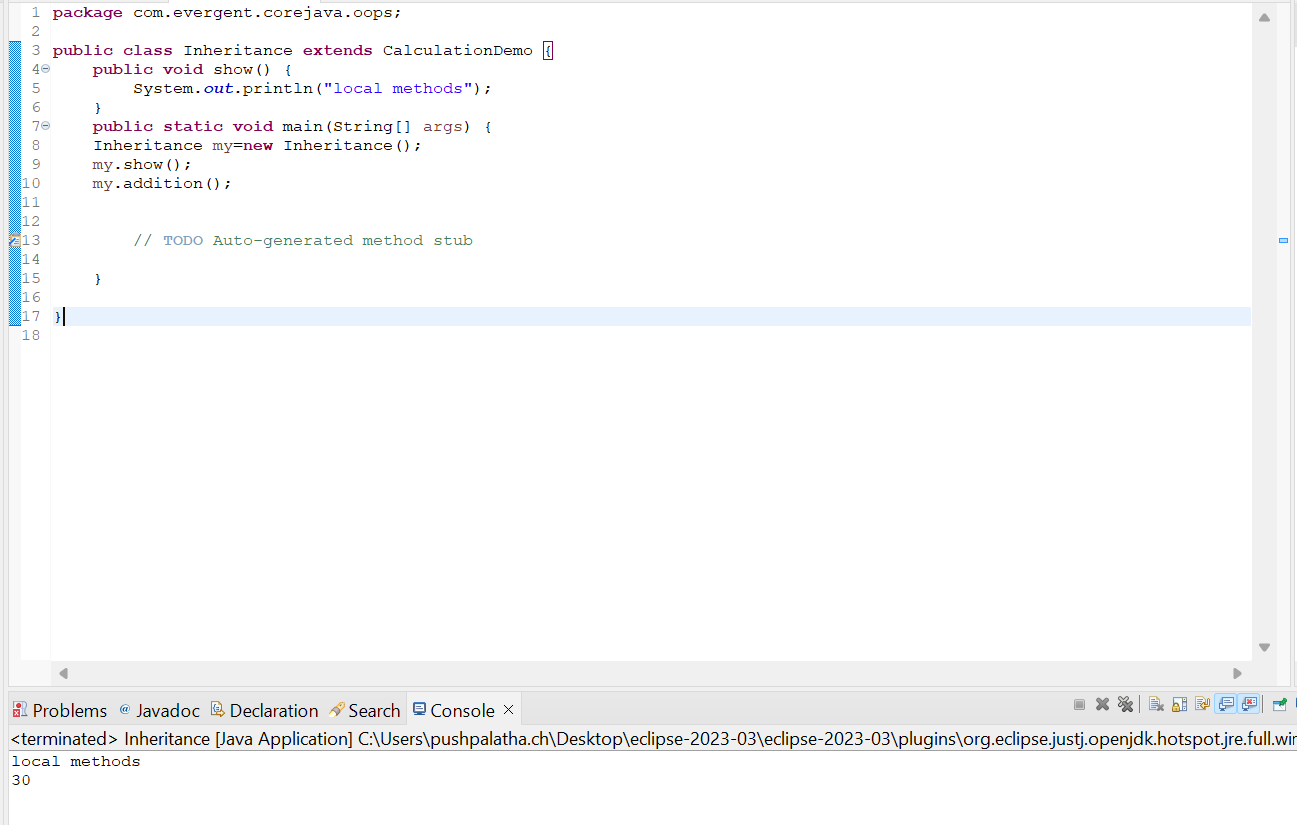
1)Performing arithmetic calculations.



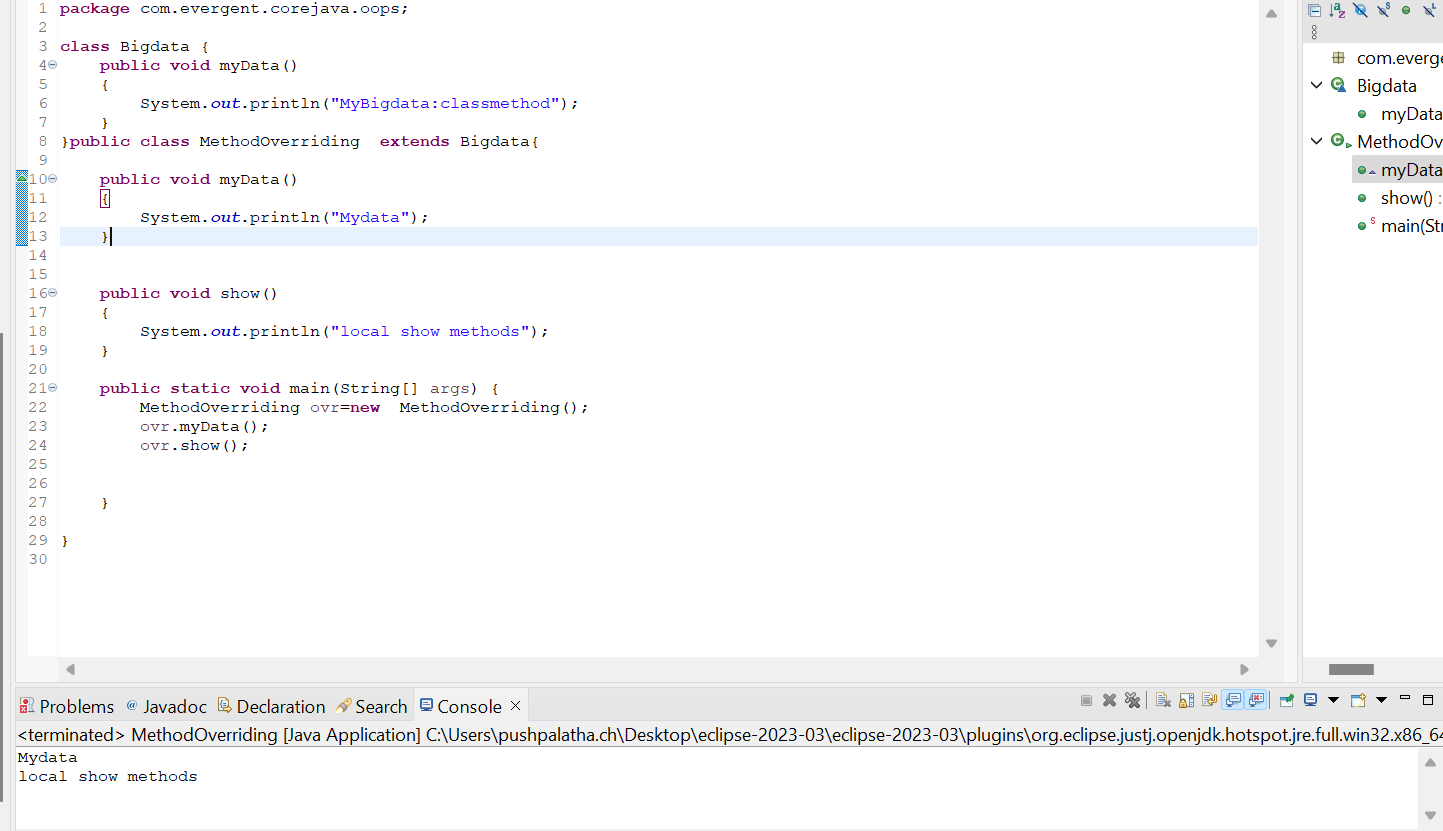
1. HAS\_A(based on object creation we can call the method from the class)



1. Inheritance(Re-usability of existing functionalities from super class to subclass)



1. MethodOverriding(run time polymorphism-with same name,same number of parameters with same return type)



1. Methods Flow

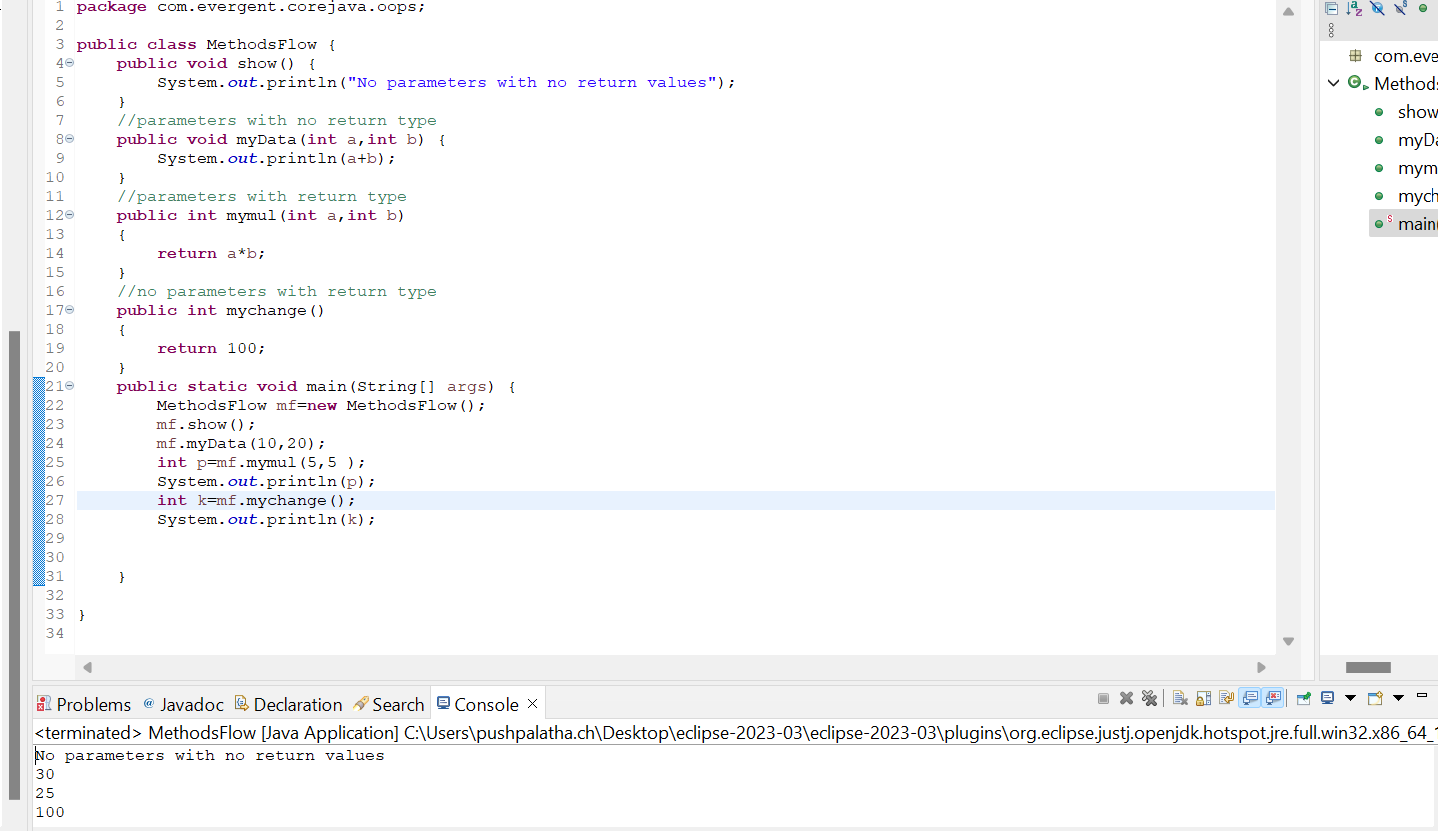
Basically we can call methods in four ways:

1)Methods with parameters without return type.

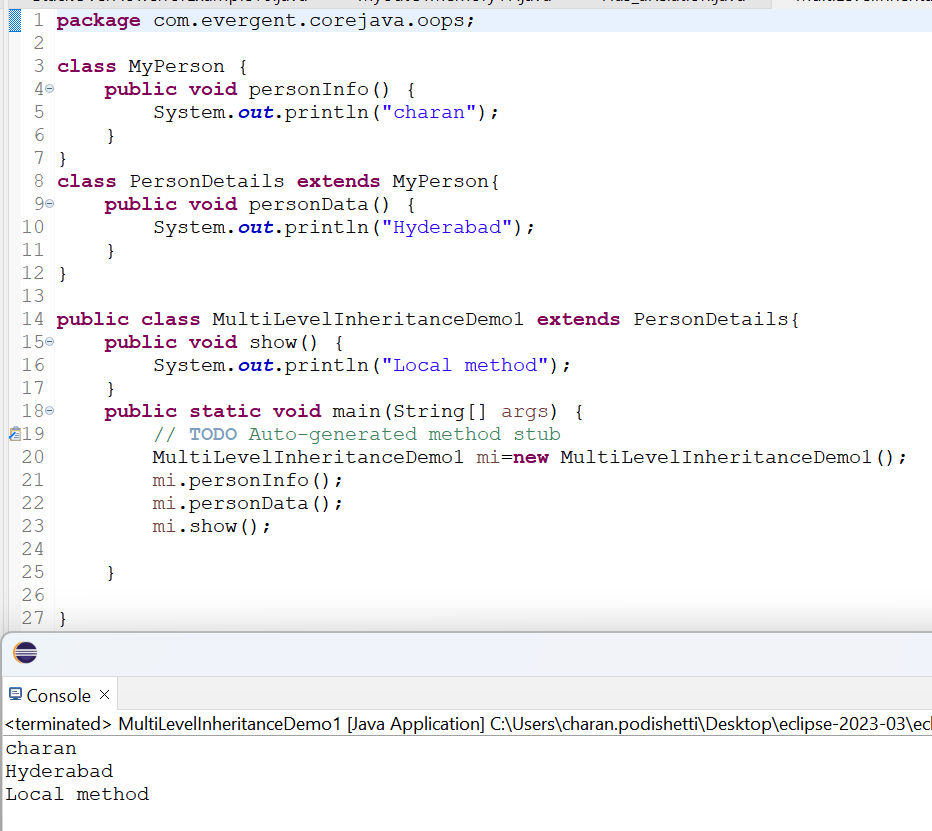
2)Methods with parameters with return type.

3)Methods without parameters without return type.

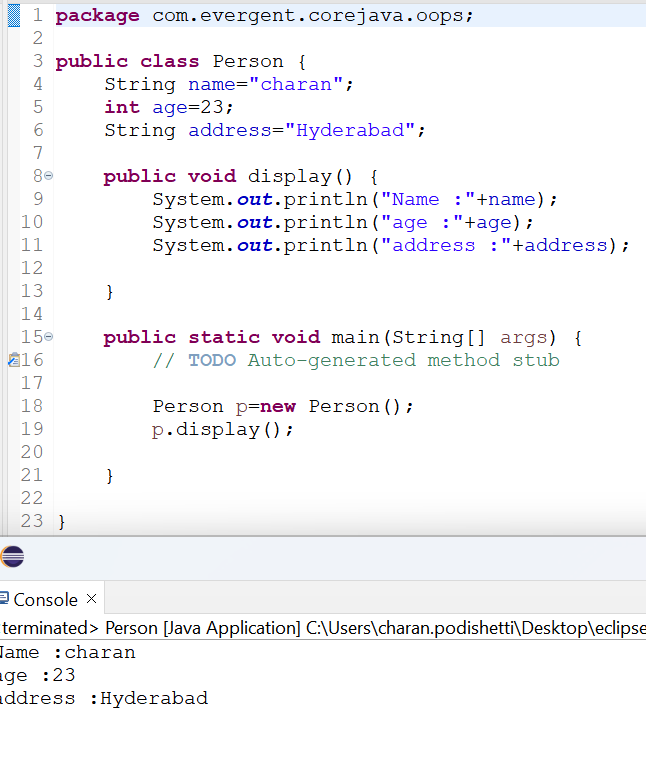
4)Methods without parameters with return type.



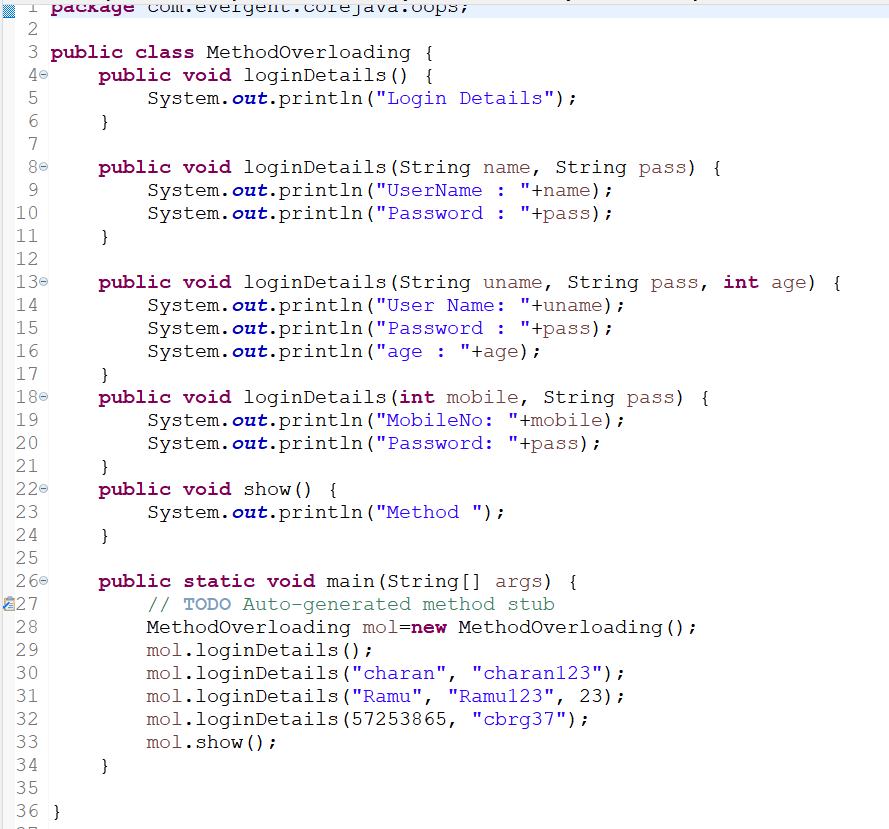
6.Multi Level Inheritance



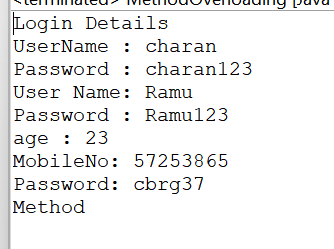
1. OOPS concept Example



1. Method Overloading(compile time polymorphism-with same name,different number of parameters with may or may not be the same return type)



**Output**:-



c.Exception handling

d.Multithreading

e.web application

f.open source

g.security

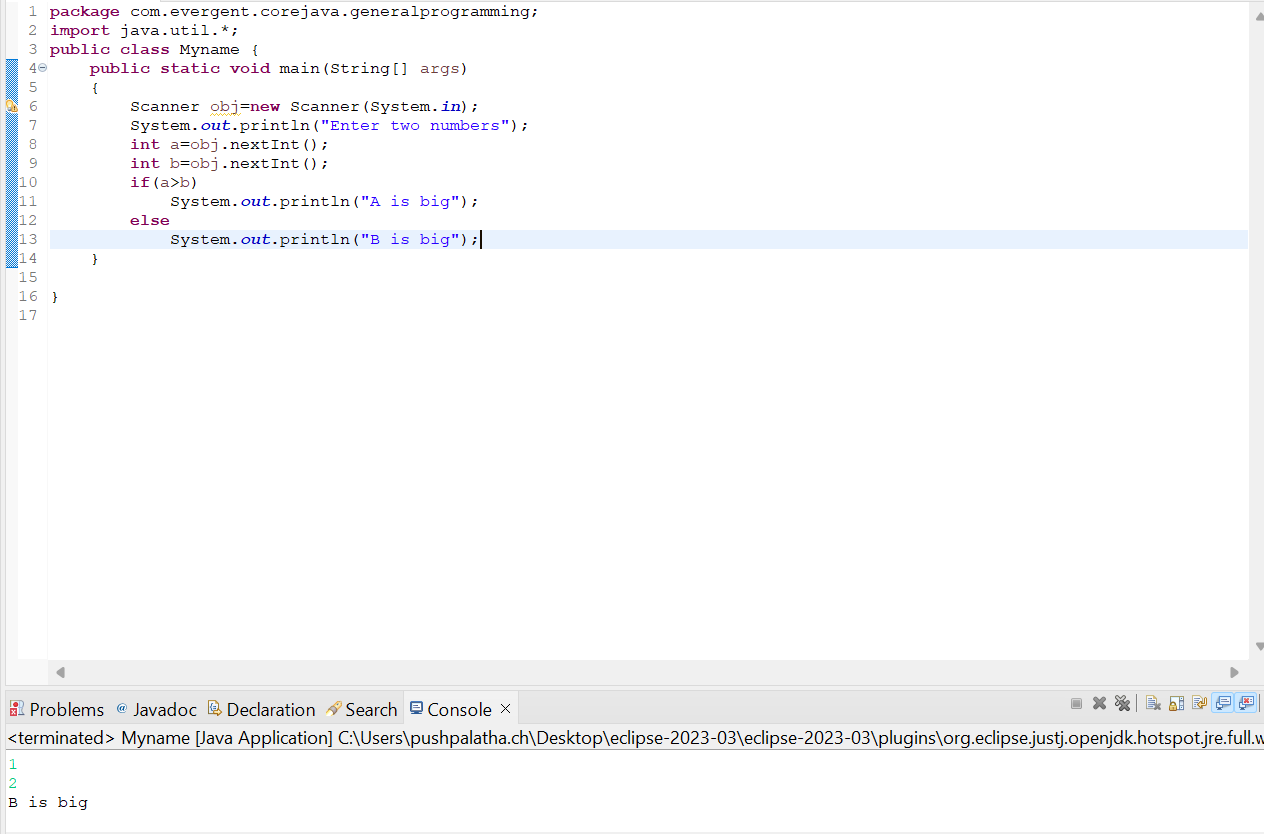
h.supports networking

i.memory management

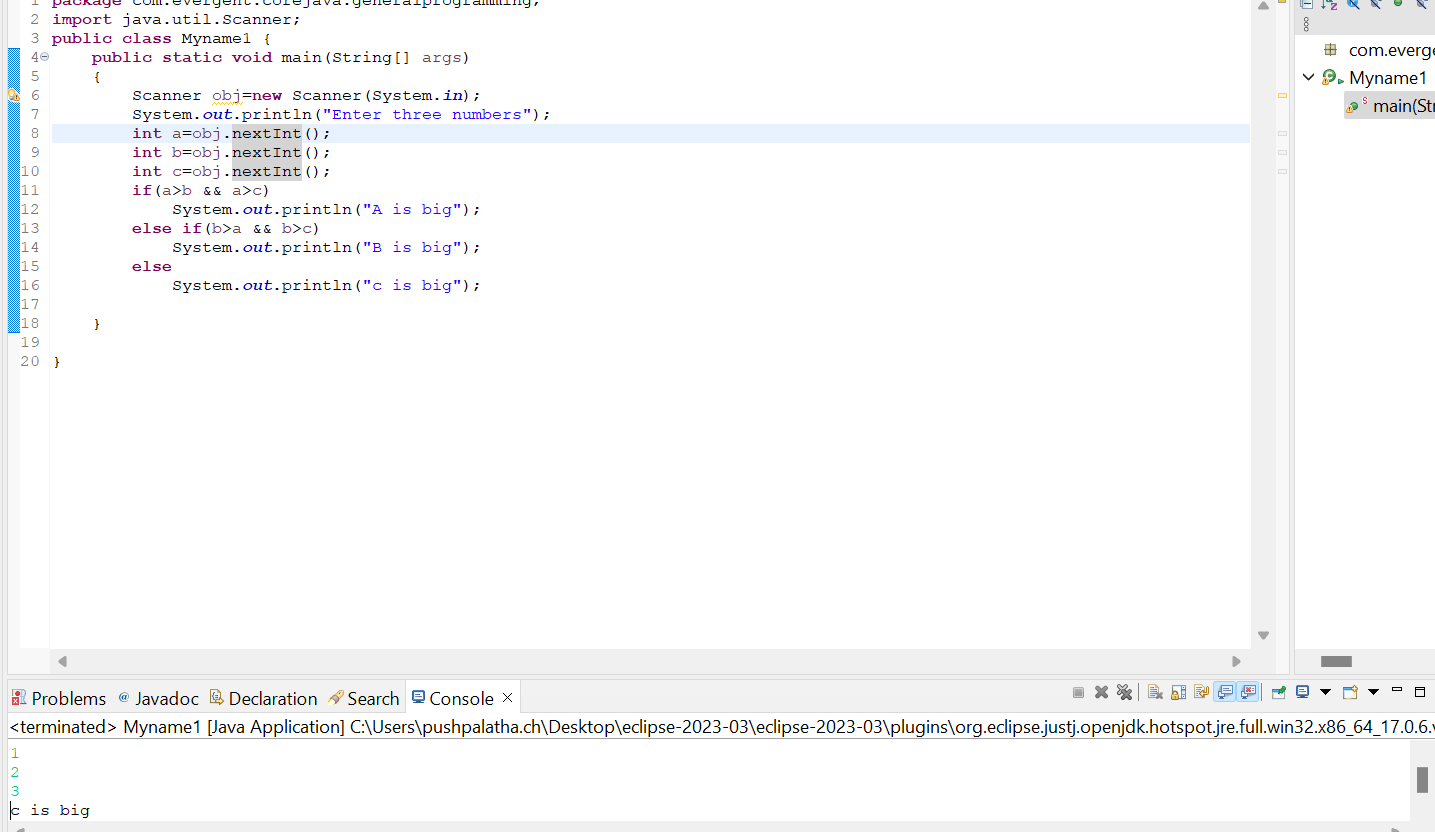
3. JDK,JRE,JVM

4.Basic programming

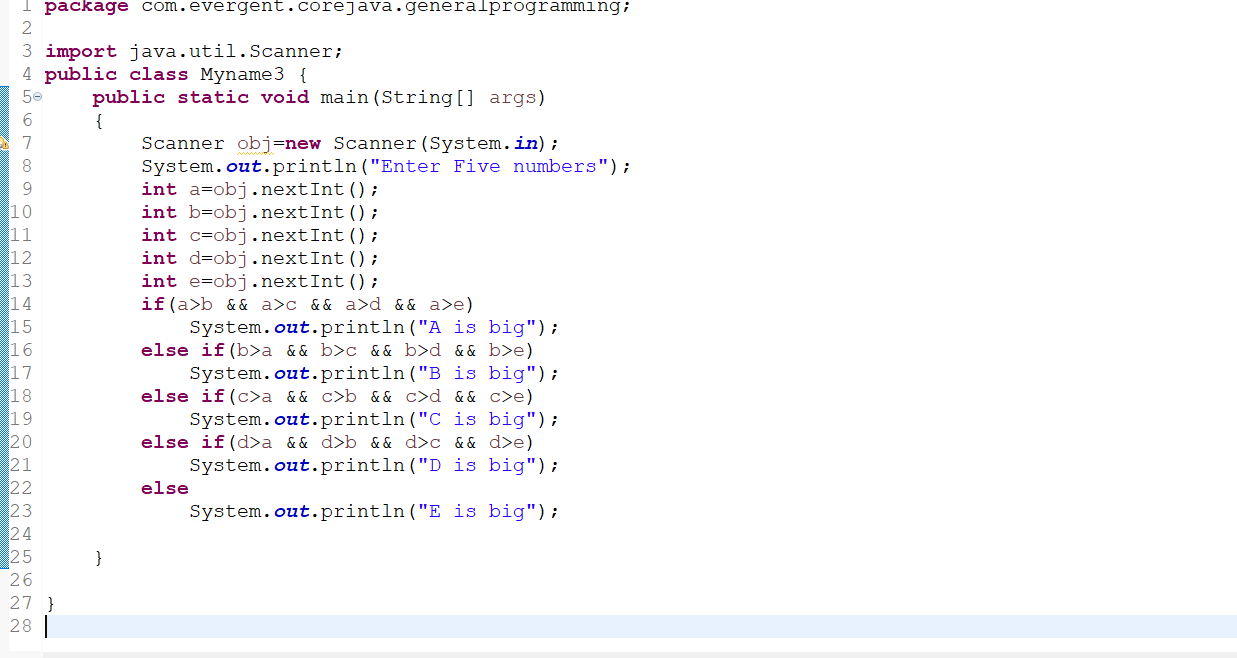
Biggest of 2 numbers

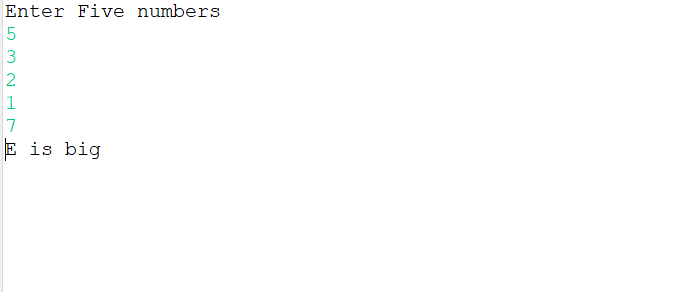


Biggest of 3 numbers



Biggest of 5 numbers





Swapping of 2 numbers without and with temp



5.Packages.

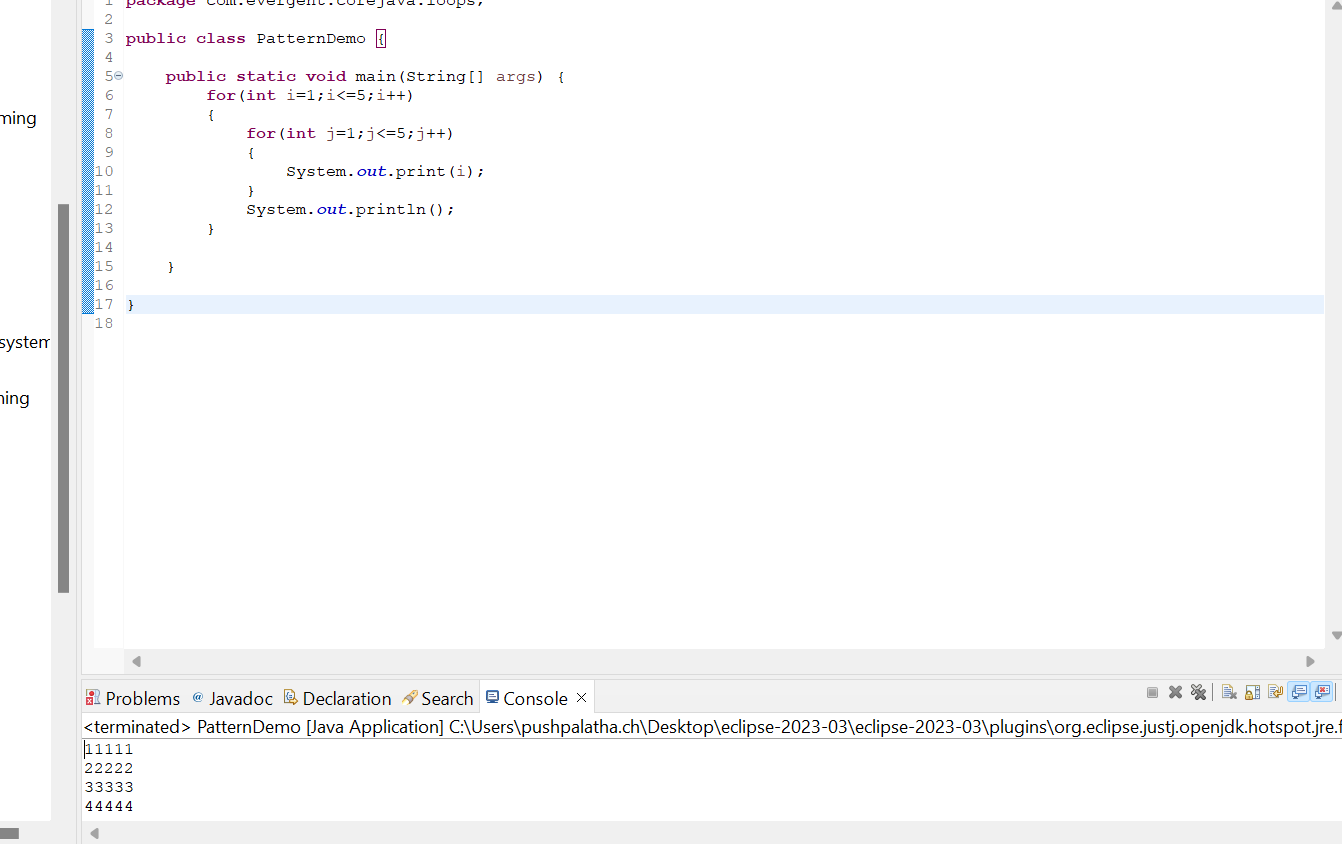
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Date:06/08/2024**

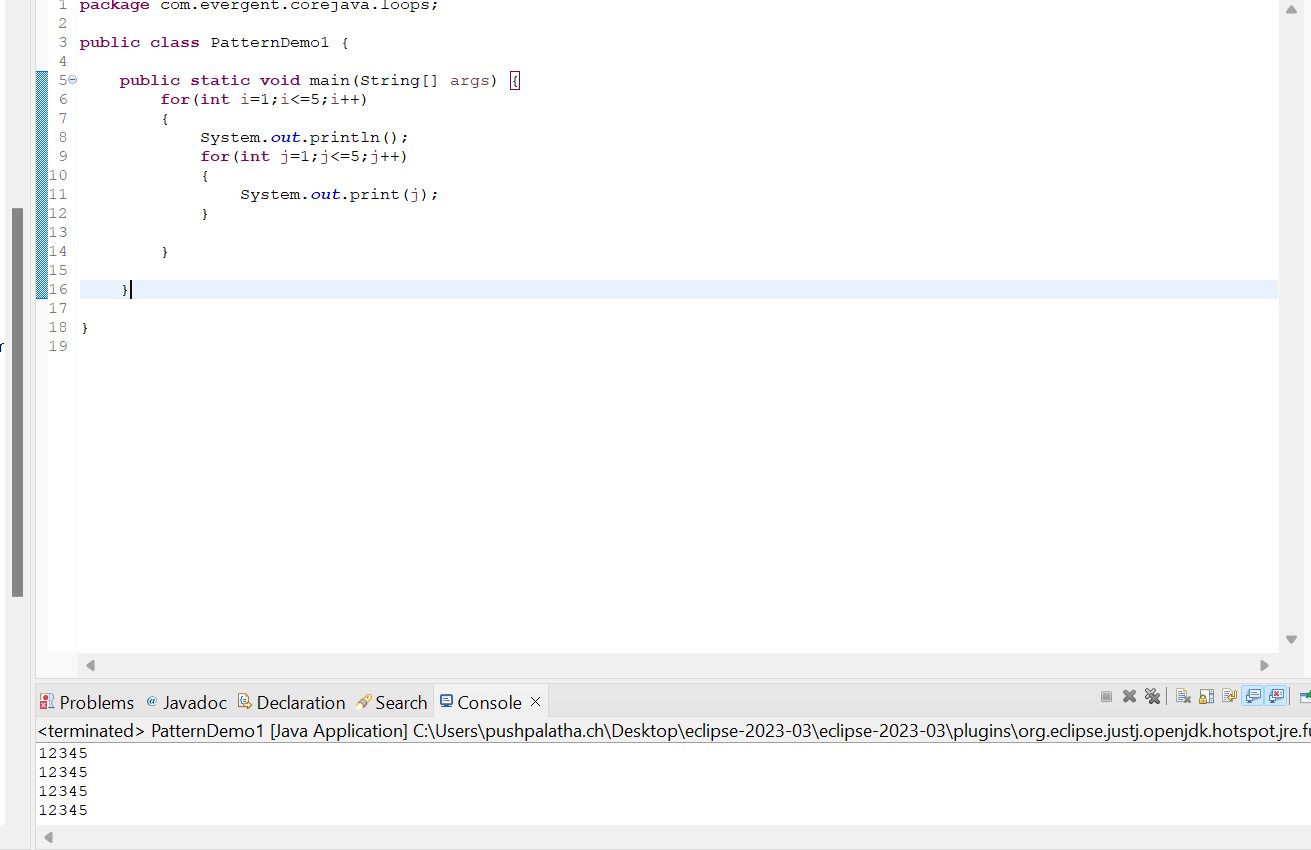
**Day-2**

1. Nested loops

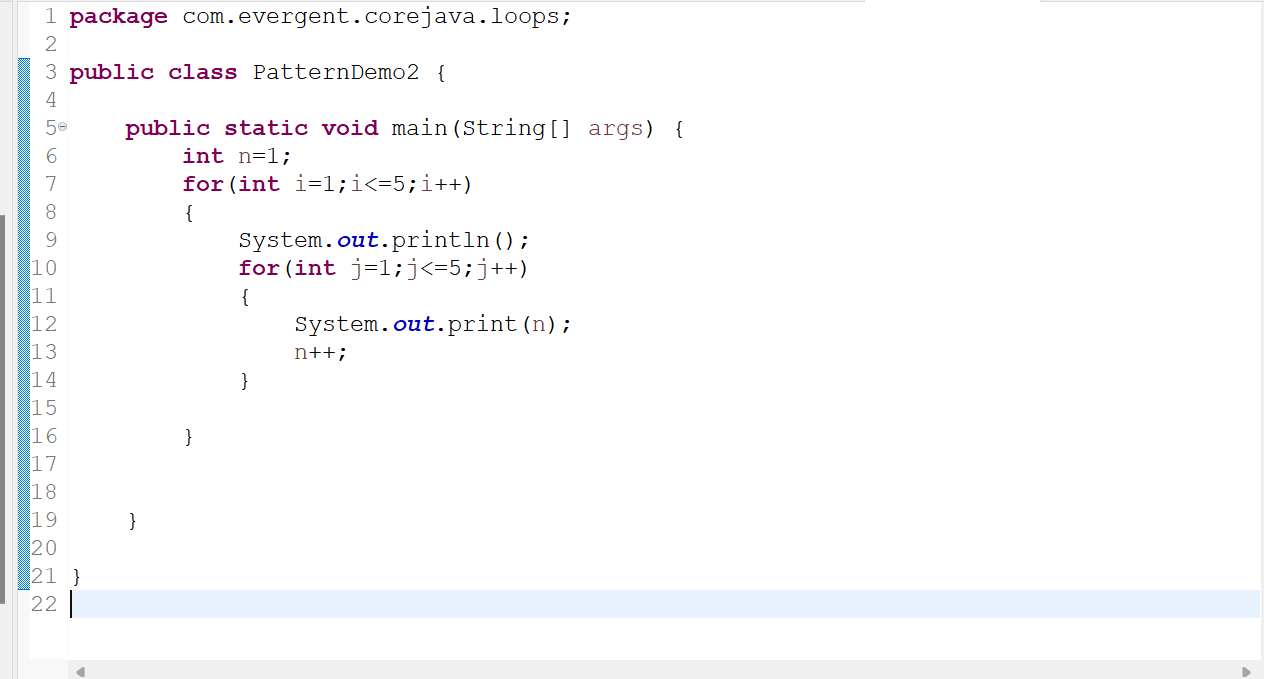
1.Pattern1

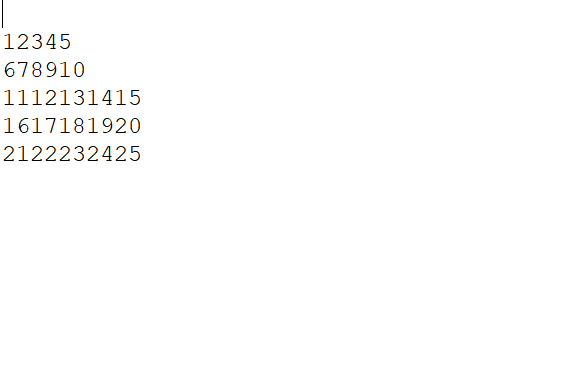


Pattern 2



Pattern 3

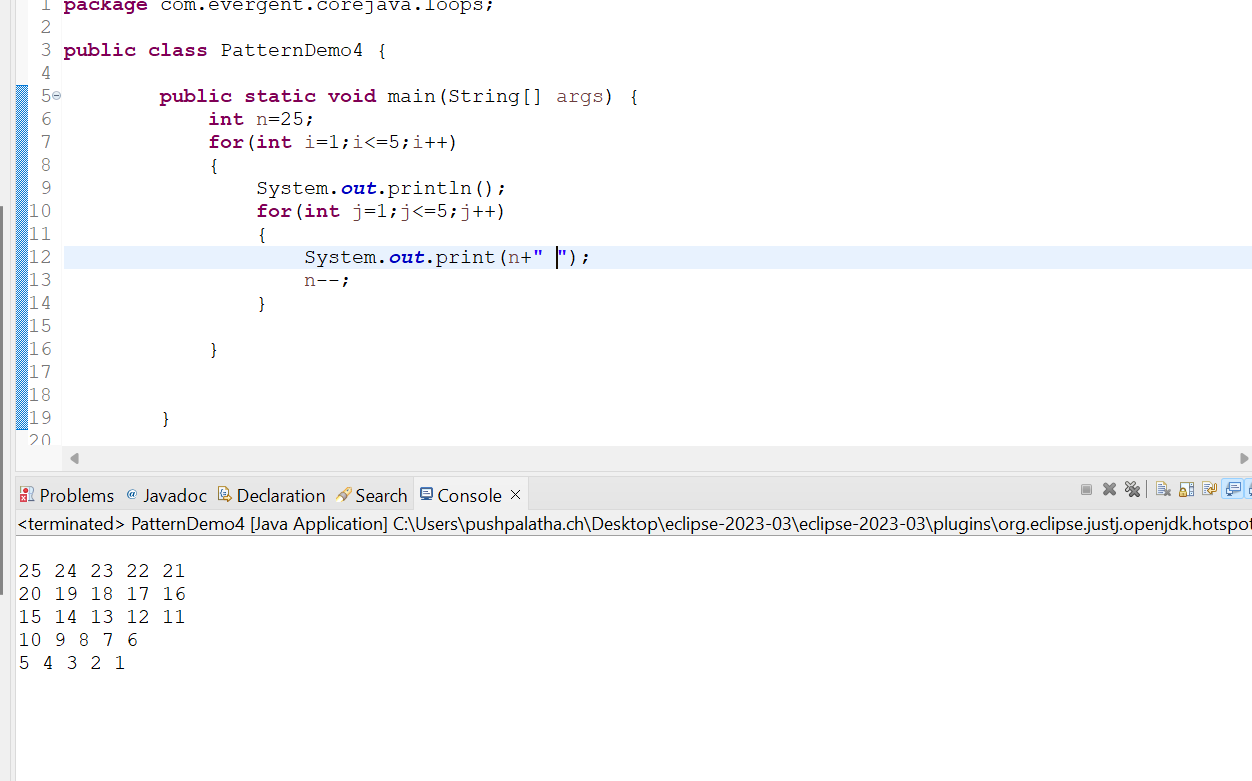




Pattern 4



Pattern 5

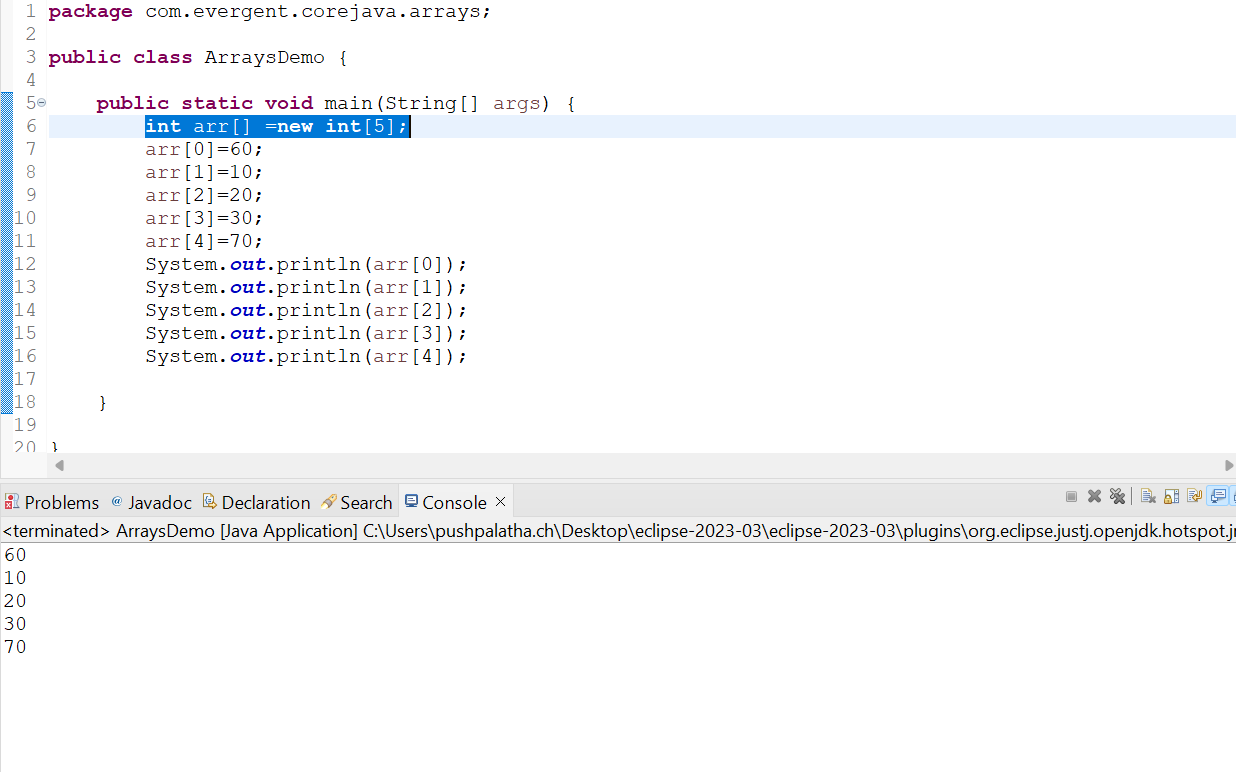


2.Arrays(collection of Homogenous data types stored in contiguous memory allocation with fixed size.)

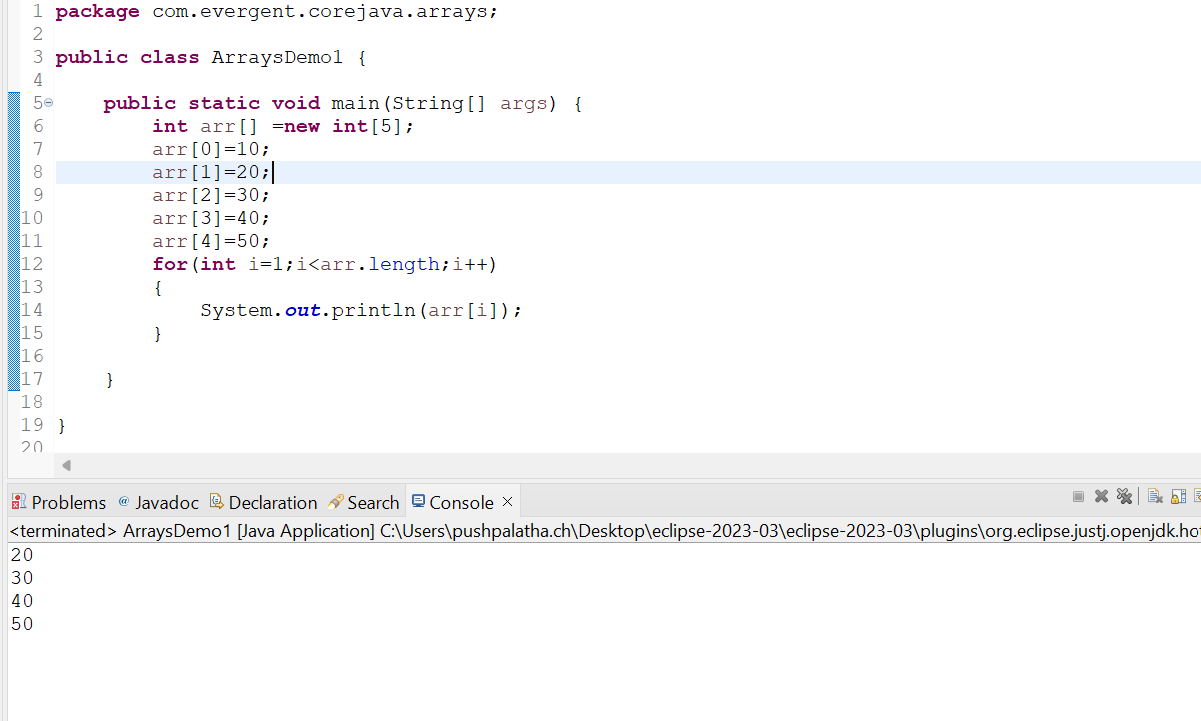
a.single dimensional array

b.double dimensional array

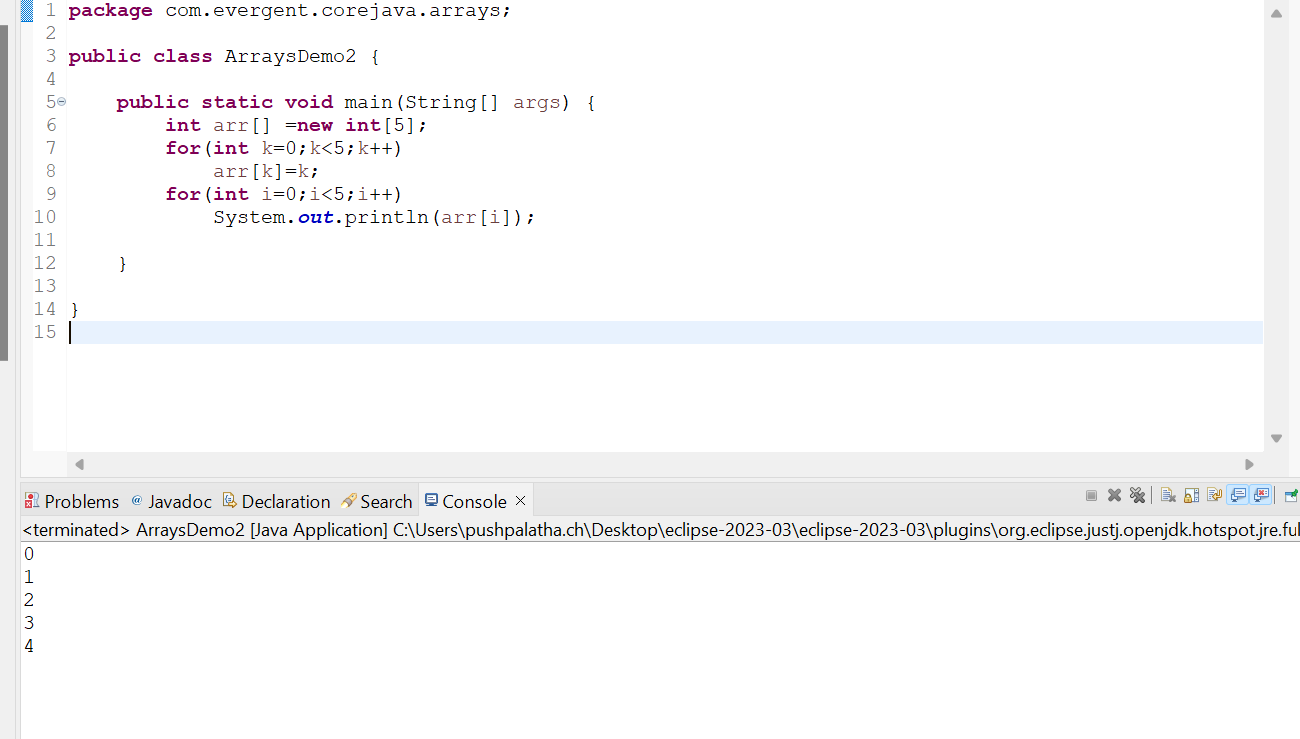
Array1(initializing,retriving of data in single dimensional array)



Array2(using for loop traversing the data in single dimensional array)



Array3



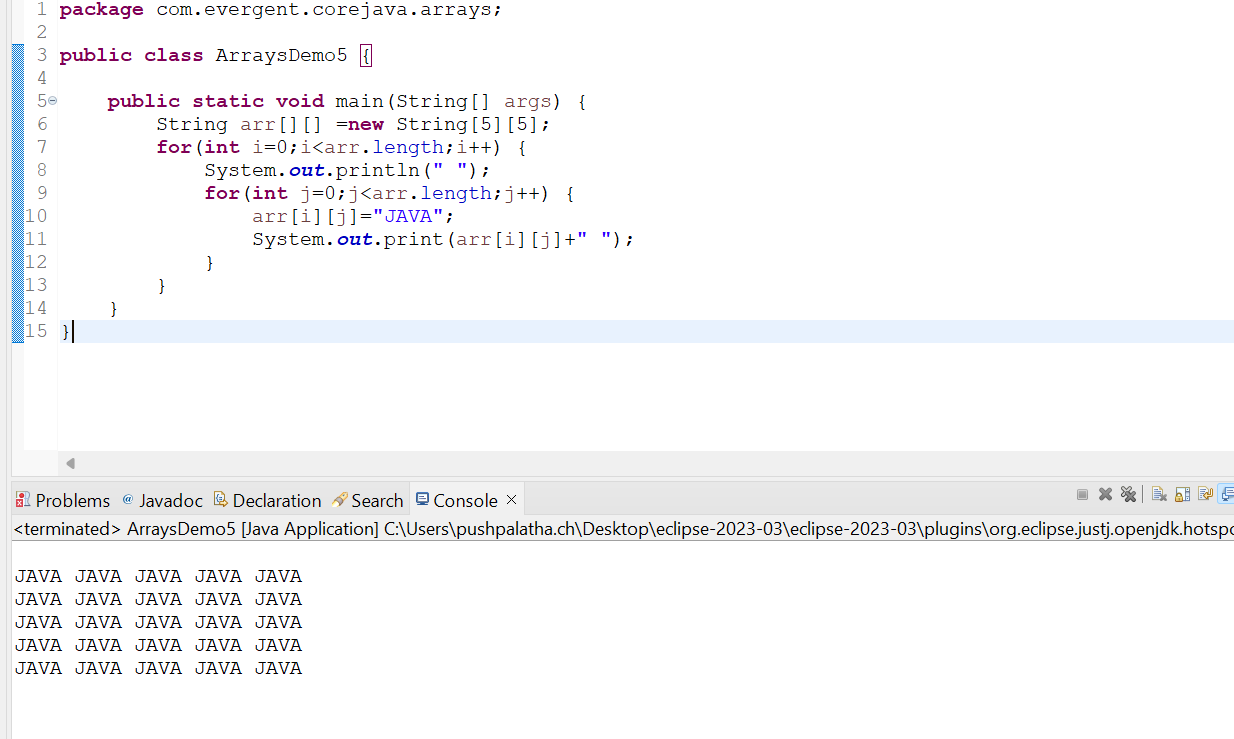
Array4



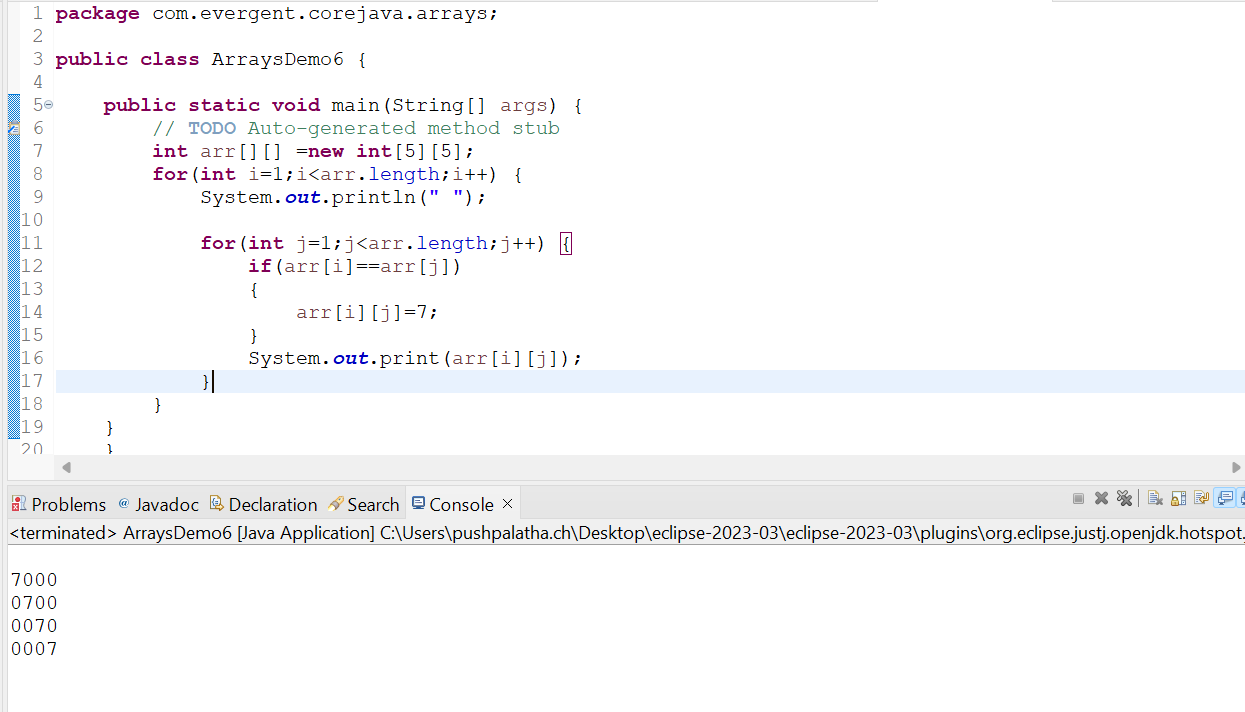
Array5



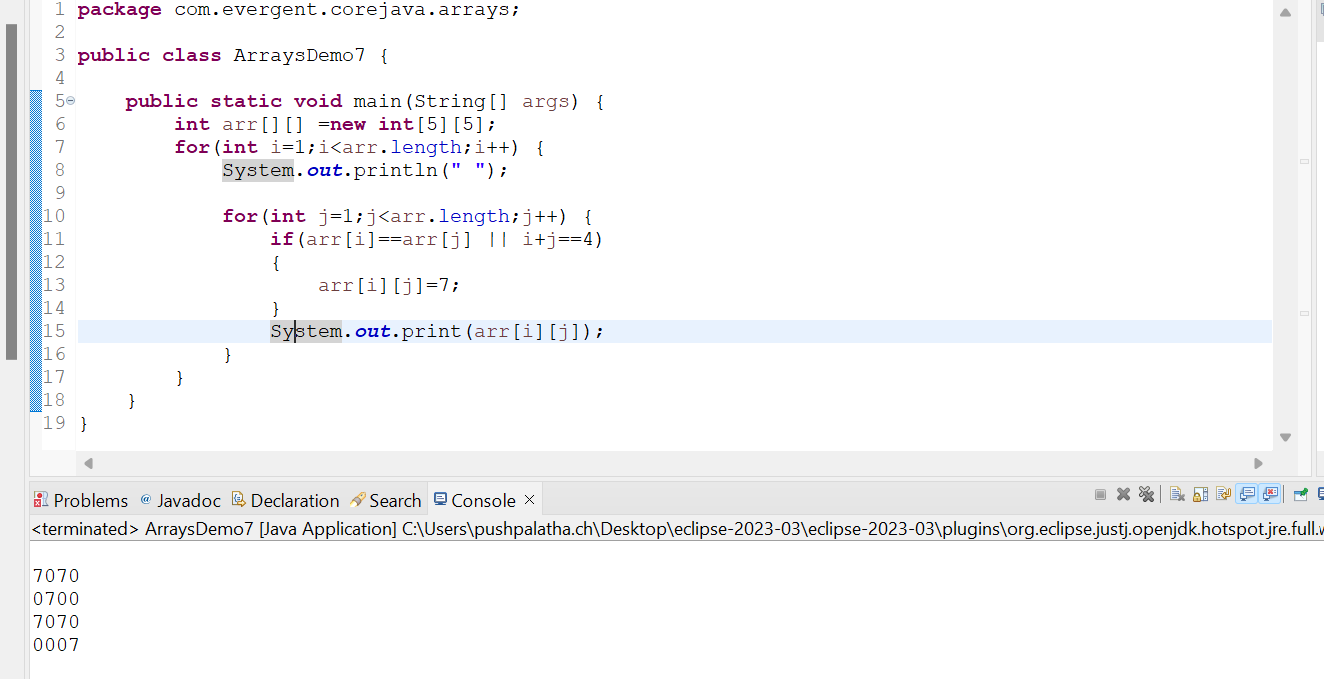
Array 6(Multidimensional Array)



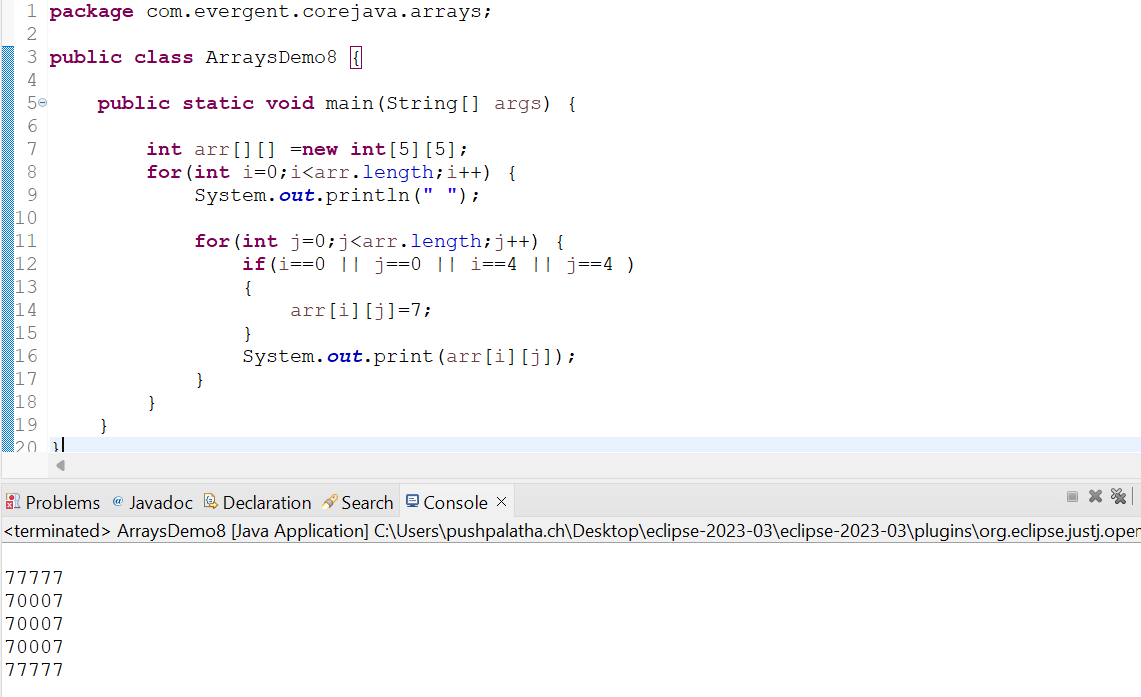
Array 7(Multi dimensional array)



Array 8(Multi dimensional Array)



Array 9(Multi dimensional Array)

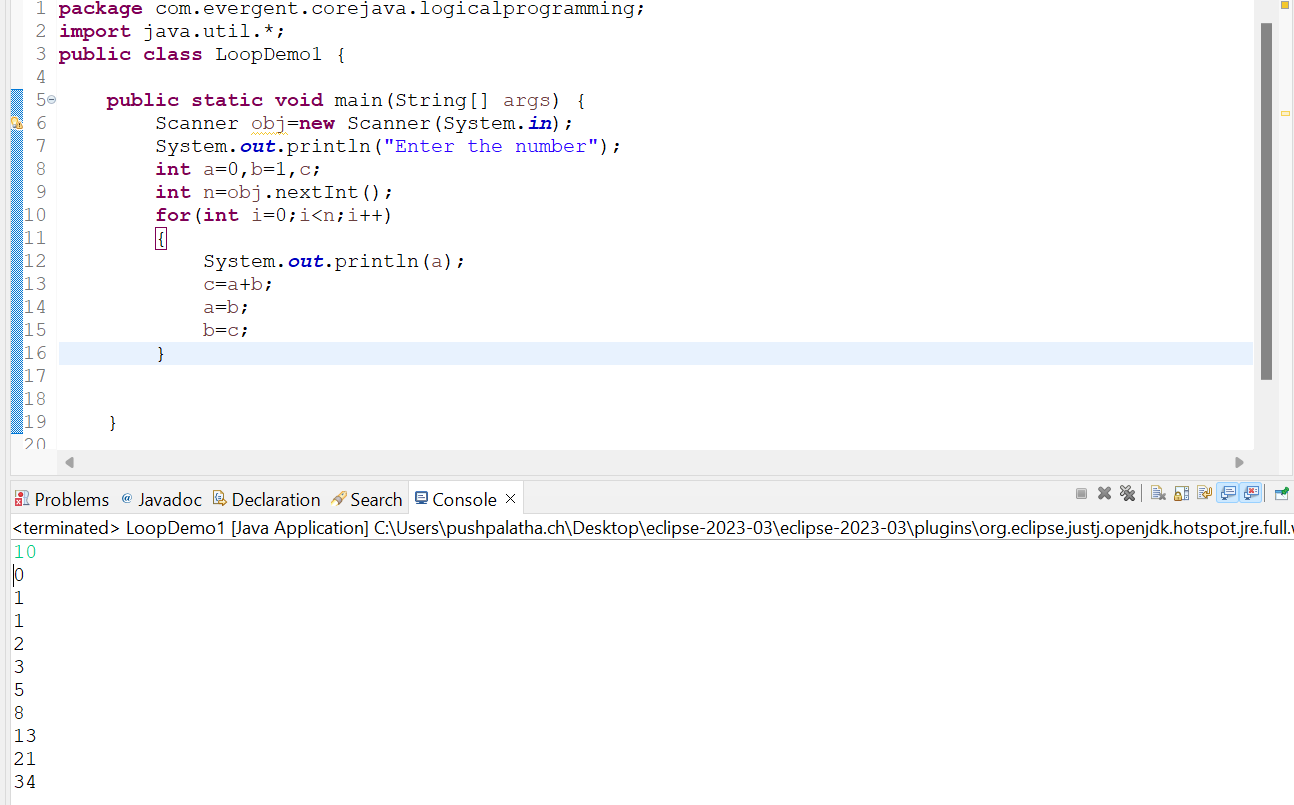


1. Logical programming

Factorial program



Fibonacci series program

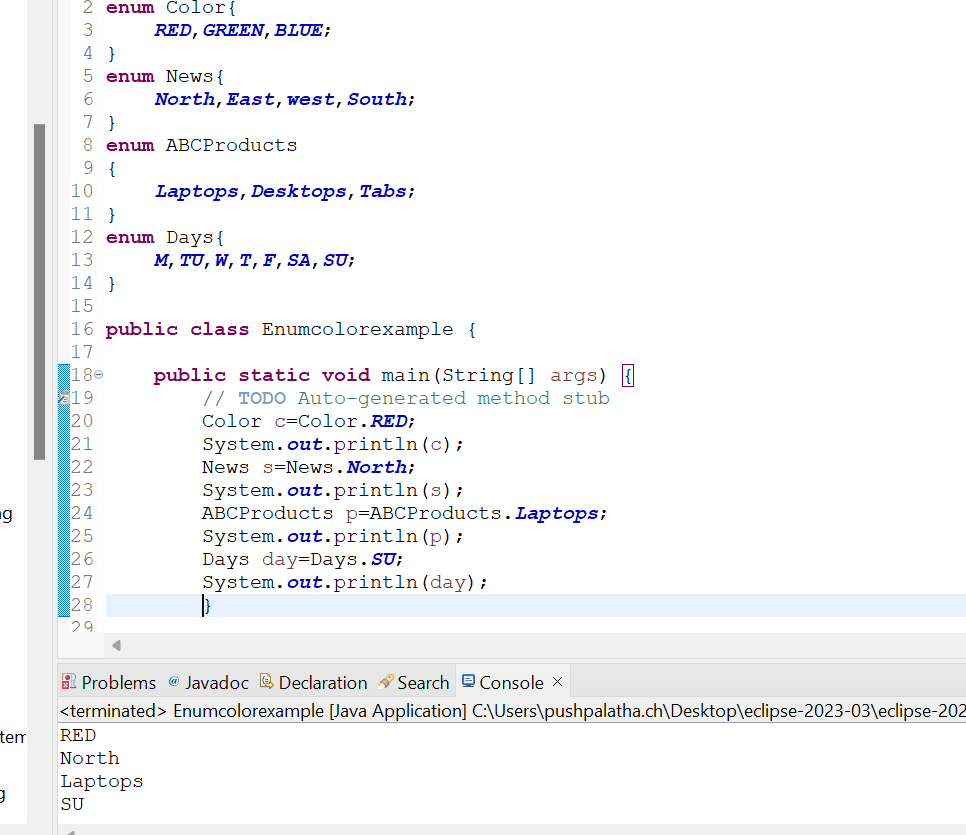


1. Switch case

5.Scanner class

6.Enum

Enum Example



7.event management system application

8.Object class methods

* equals()
* toString()
* getClass()
* finalize()
* clone()
* wait()
* notify()
* notifyAll()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Date 07/08/2024**

**Day-3**

1.Started OOPS concepts,

a.Encapsulation,

b.Inheritance,

c.Polymorphism (method overloading and method overriding) with example programs

d.method flows with suitable example program,

e.Is-A & Has-A relationships.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

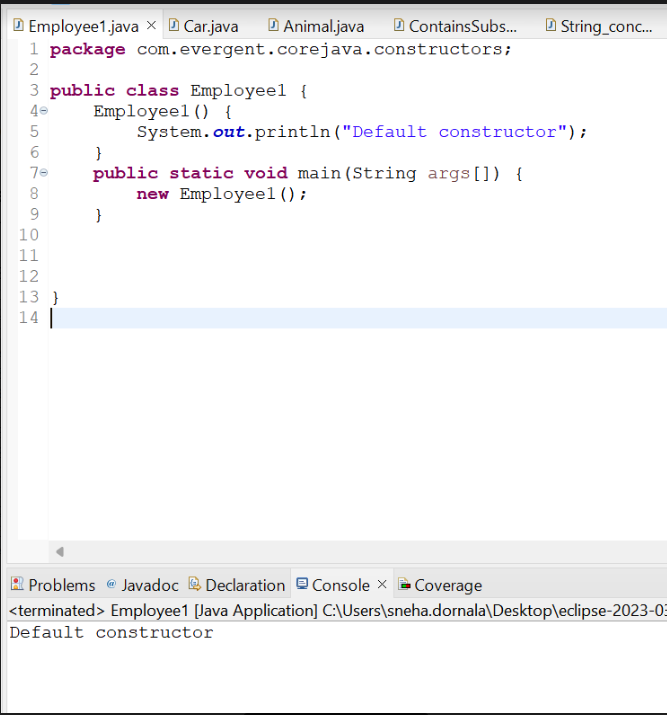
**Date 08/08/2024**

**Day-4**

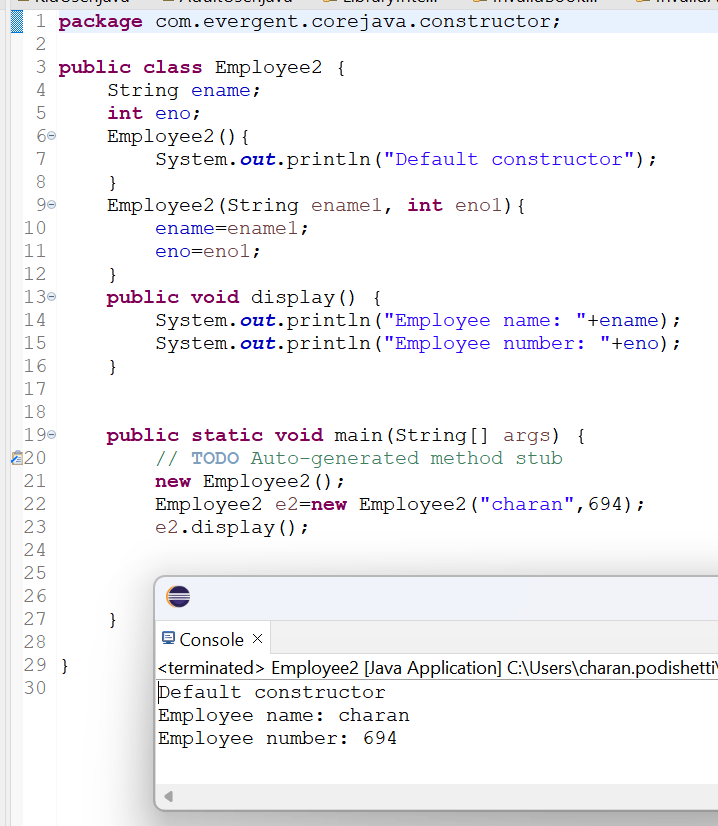
**Constructors**

1. Class name and constructor name should be same.
2. There are 2 types of constructors

Default constructor



Parametrized constructor

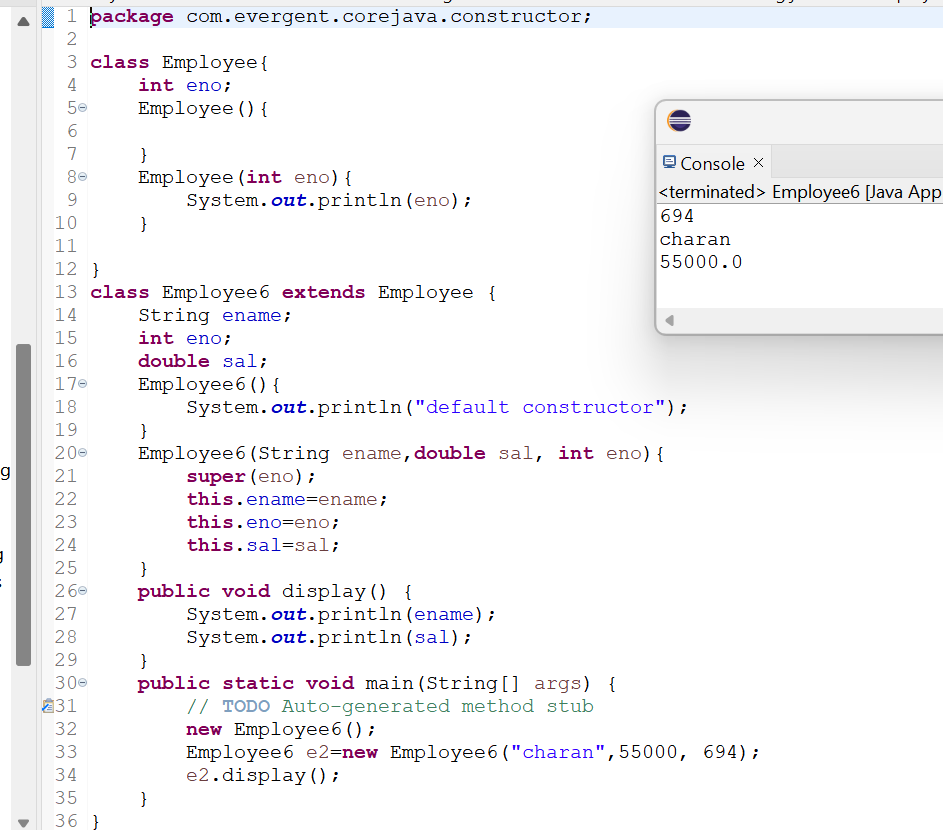


1. We can access constructors while creation of objects.
2. Constructors are mainly for intializing the objects.
3. Constructors does not any return type not even void also .If we declare as a void it will consider as a method not as constructor.
4. Every Class Contains atleast one default constructor Either we will create or JVM will generate at the time of compilation.
5. this and super Keywords.

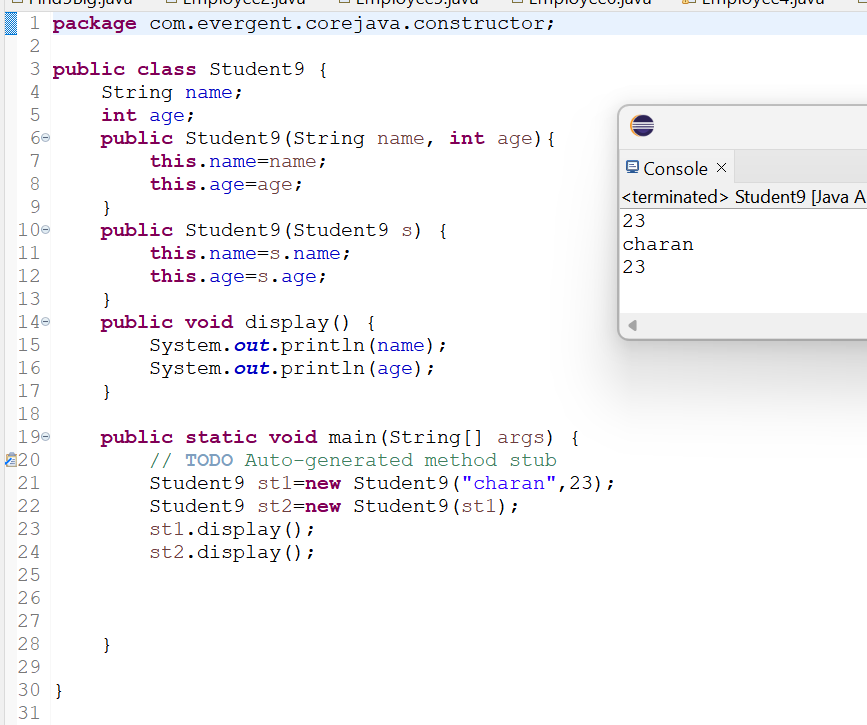
this(this is generally always pointing to instance variables and used to call current class constructor within the class)



Super(Super keyword is used to call super class constructor in constructors)

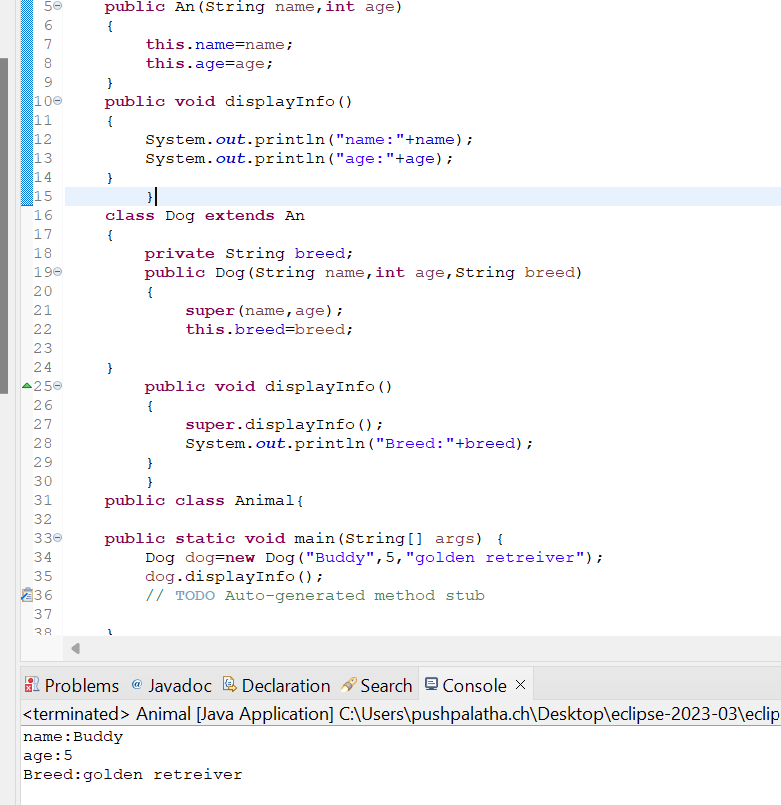


1. Object copy constructor.



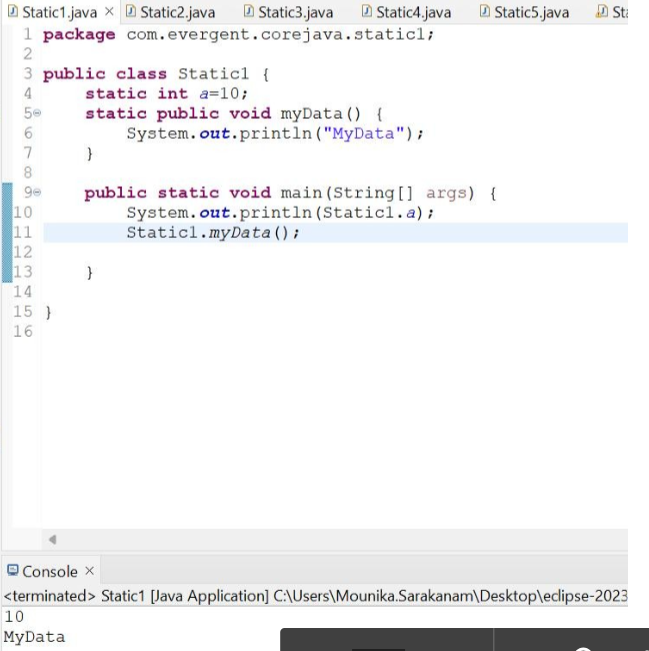
1. Always Constructors are overloaded.
2. We can call one Constructor to another constructor through this keyword.
3. We can intialize values to default constructor.



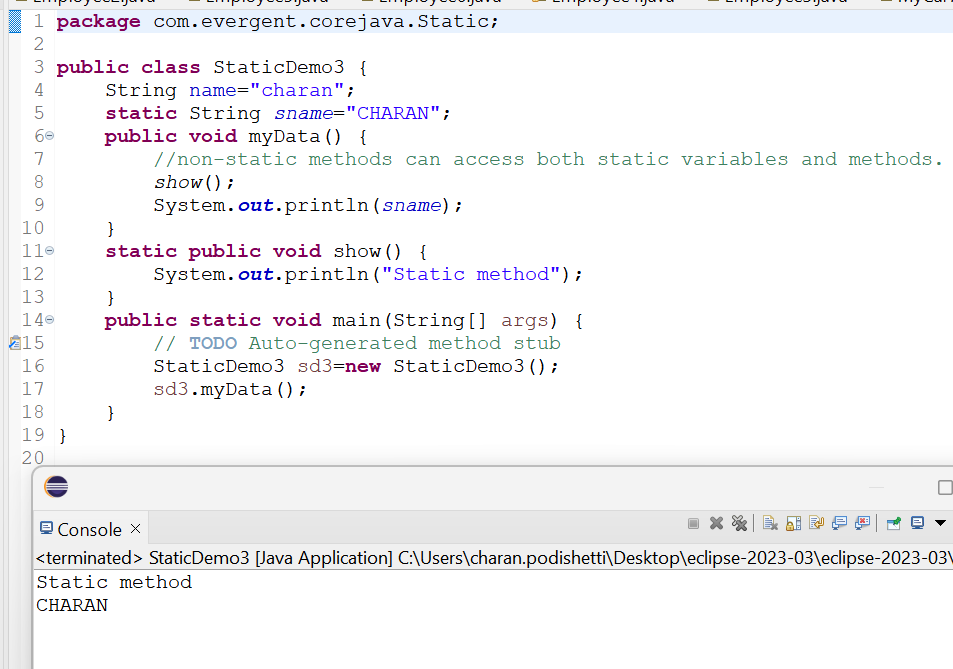
ConstructorOverloaded

**Static keyword**

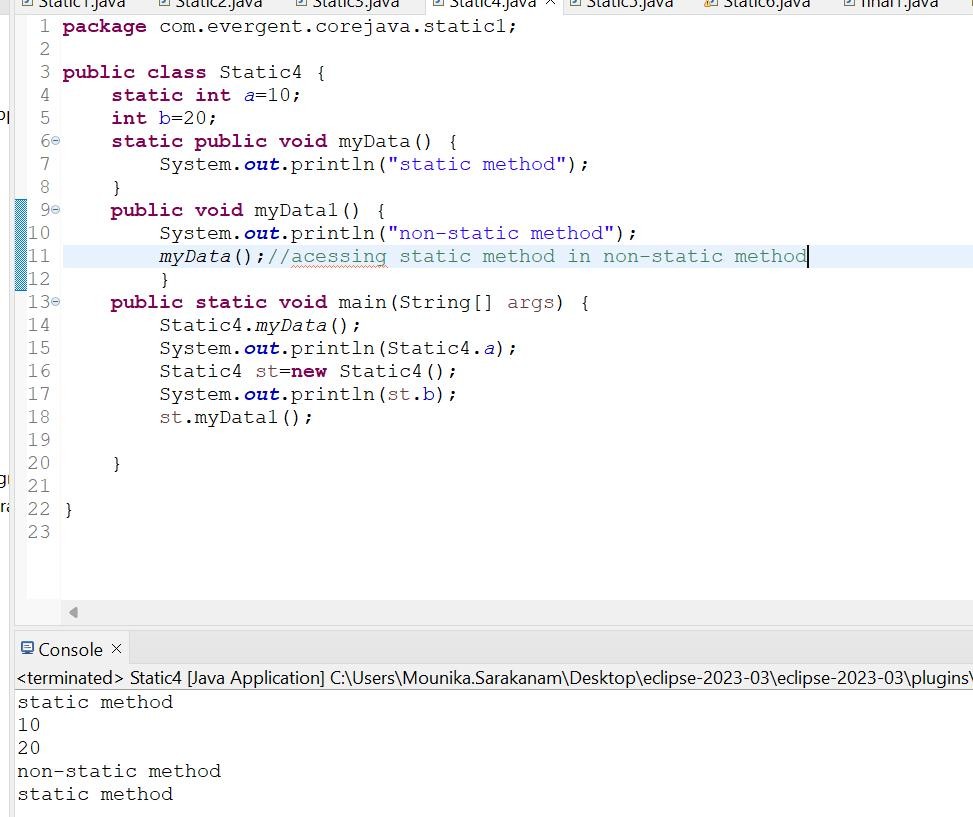
We can declare static as variables,methods



Inside static method we cannot access non static variables or methods.



Inside non static method we can use static methods or variables.

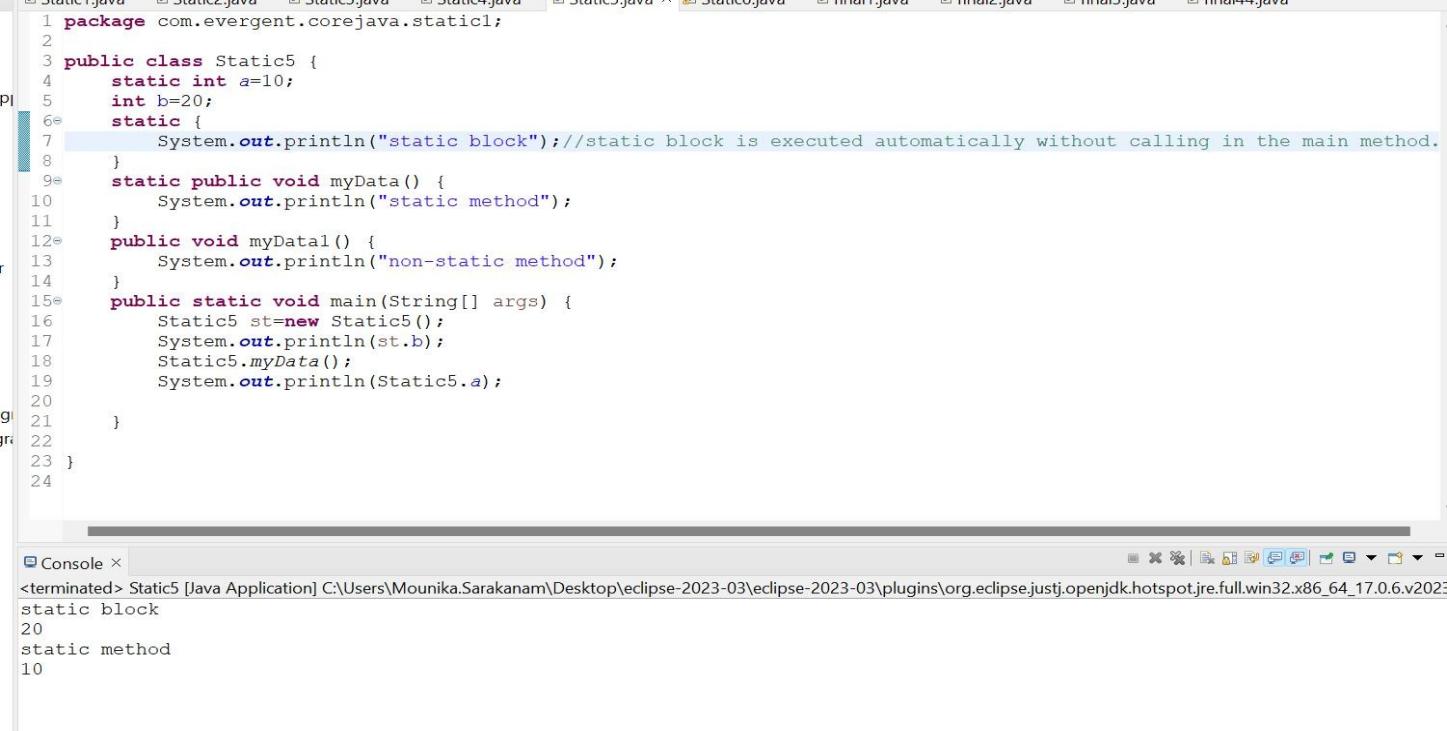


Static block:

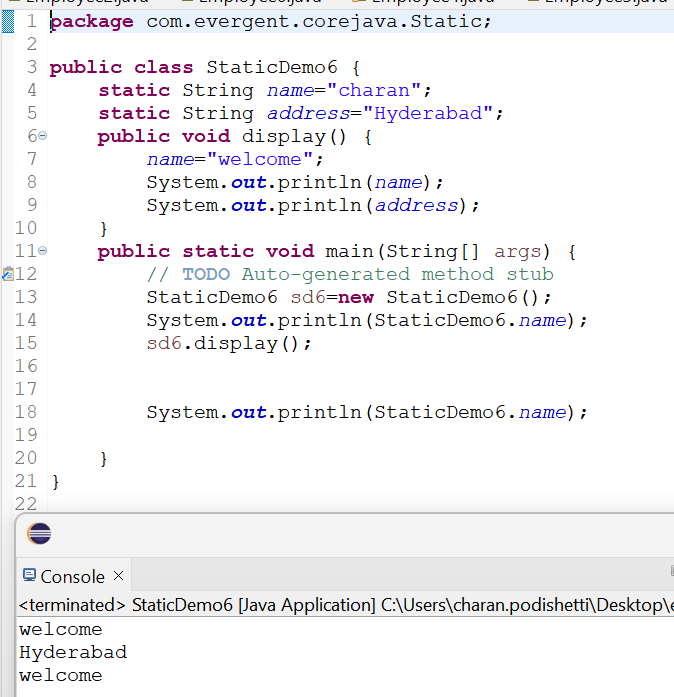
Static block is executed automatically without calling the block in the main method.static block is compiled durring the execution of class.At first static block is executed then remaining main method,classes,etc will be executed.

Static variable can be initialized while declaring it or while constructor, or in static block.

We can initialize “final variables” in static block but it should “static” also. Else it will show compile time error.



Example for static

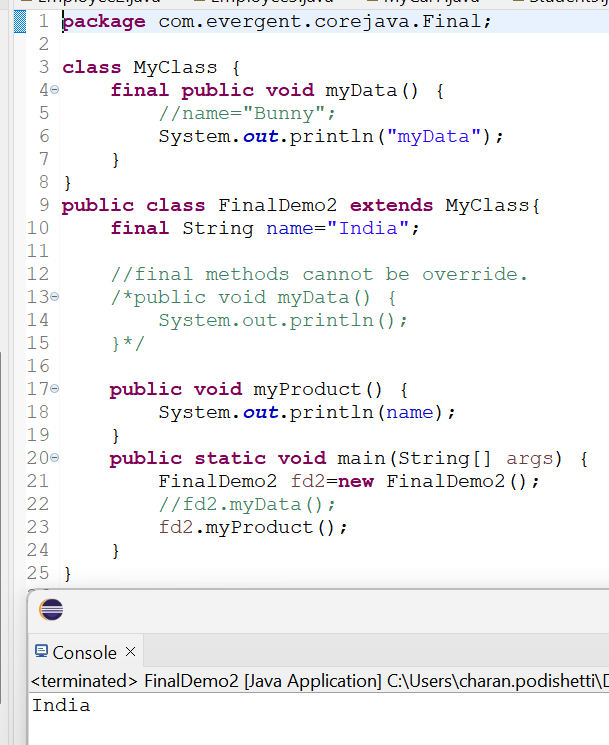


# final

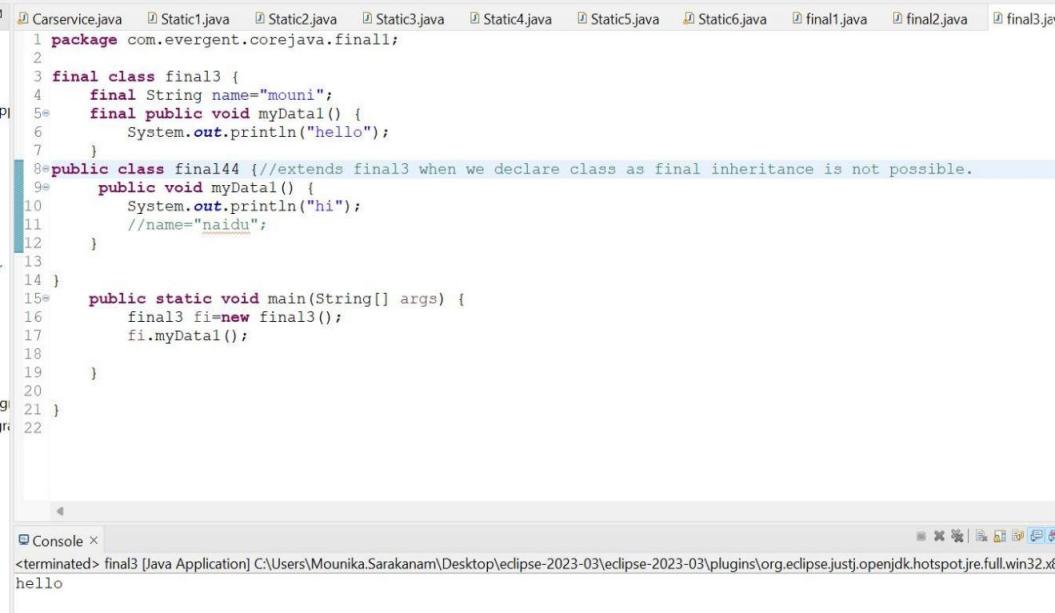
final is a keyword.

final can be declared as variable,method,class

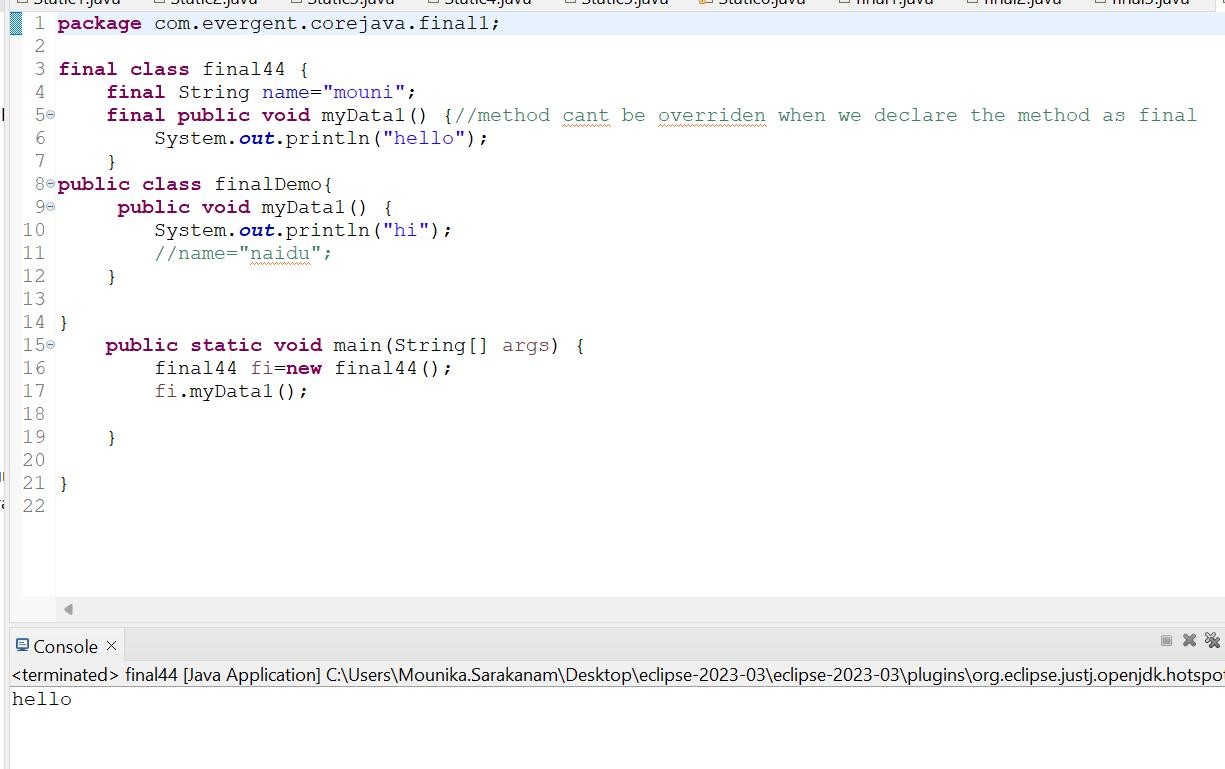
If variables are declared as final those are treated as constants cannot be modified after.



If the class is declared as final we cannot inherit



If the method is declared as final we cannot overriden.



* Final variable can be initialized while declaring it, in instance block, in constructors.

STRINGS

-String is a sequence of characters.

|  |  |  |
| --- | --- | --- |
| String | StringBuffer | StringBuilder |
| Is a final class(can’t extend) | Is a final class(can’t extend) | Is a final class(can’t extend) |
| Immutable(can’t modify) | Mutable(can modify) | Mutable(can modify) |
| Methods in String class are non-synchronized(not-thread safe) | Methods in StringBuffer class are synchronized(thread safe) | Methods in StringBuilder class are non-synchronized(not-thread safe) |
| Strings are used for Single update | StringBuffer is not recommended as it has legacy API | StringBuilder are used for multiple updates.It is started from JDK 1.5 |

String creation:

We can create String in two ways

1. Using new keyword.(Object is created for every new keyword even though the same data is present in the heap area)
2. Using String literal.(Object is created if and only if the keyword is not present in the String Constant pool.)

String str1=new String(“JAVA”);

String str2=new String(“JAVA”);

String s1=”JAVA”;

String s2=”JAVA”;

String constant pool

JAVA

JAVA

JAVA

String class

* String is a final class.
* Strings are immutable.
* String class consists of methods.
* Methods in String class are non-synchronized.

1. Creating String using new keyword.

equals() in String used to check the content.

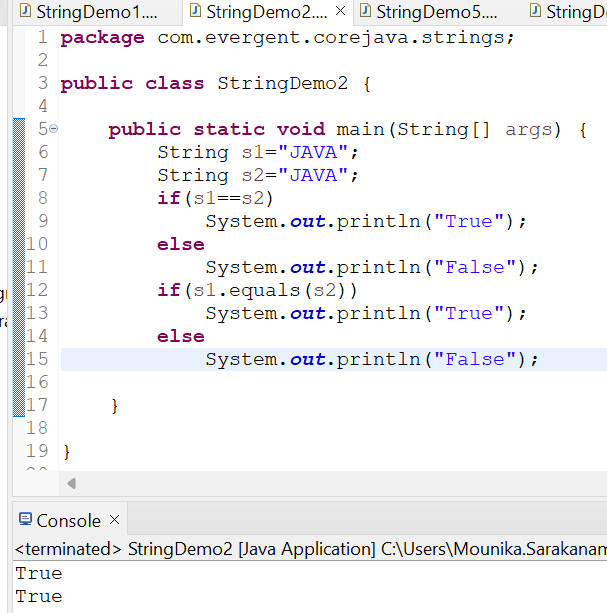
== method in String is used to check the Memory location.



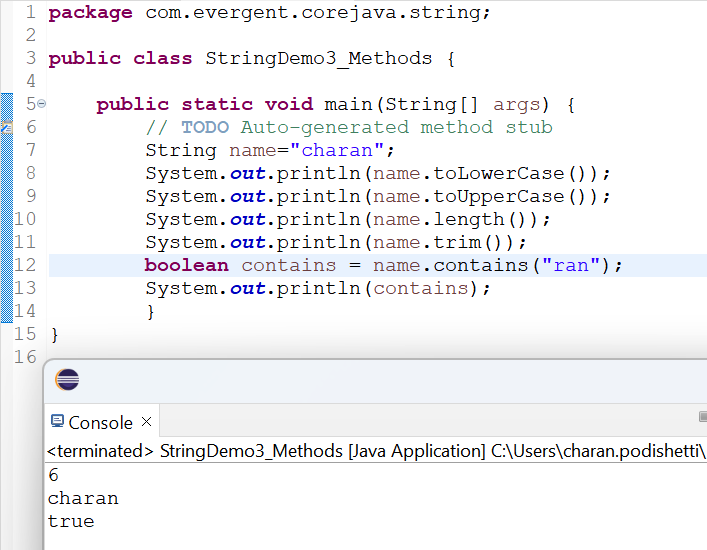
2)Creating String using literals

equals() in String used to check the content.

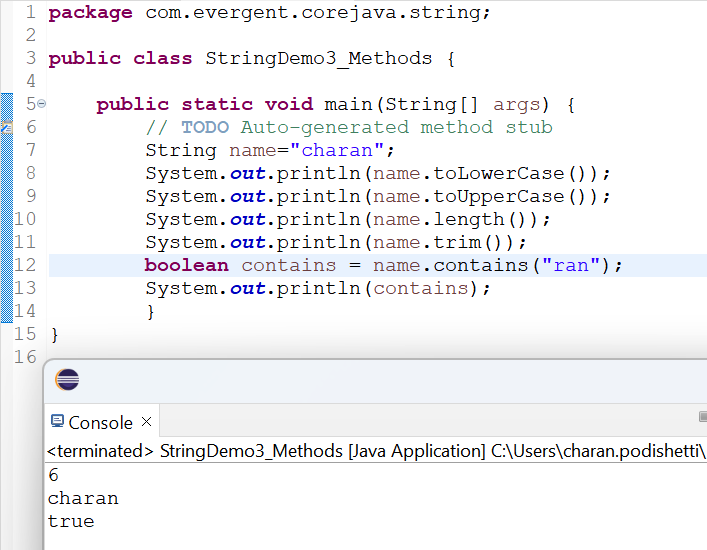
== method in String is used to check the Memory location.



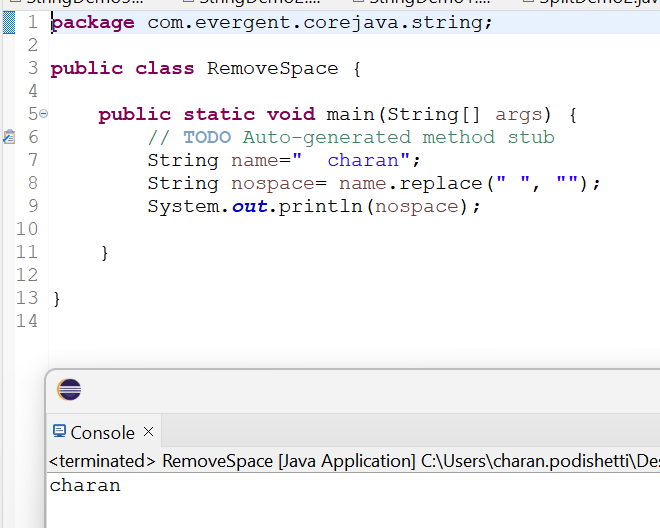
1. String methods (toLowerCase()-Changes the complete String into lowercase,
2. toUpperCase()-Converts the complete String into uppercase,
3. length()-To know the length of the String,
4. trim()-To remove the spaces before the String)



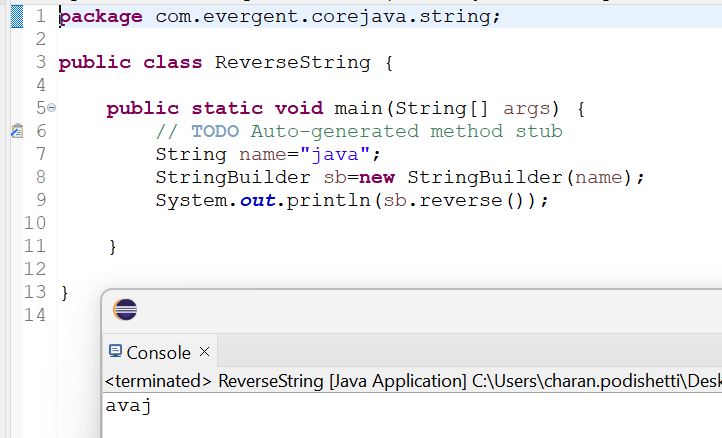
4)String method(contains()-to check whether the substring is present in the string)



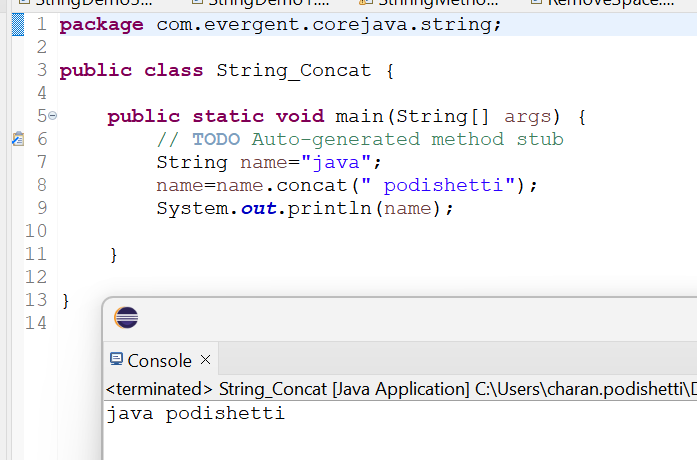
5)String method(replace()-to replace the values with other values)



1. String method(reverse()-to reverse of the String)



7)String method concat()-to combine two strings into one String

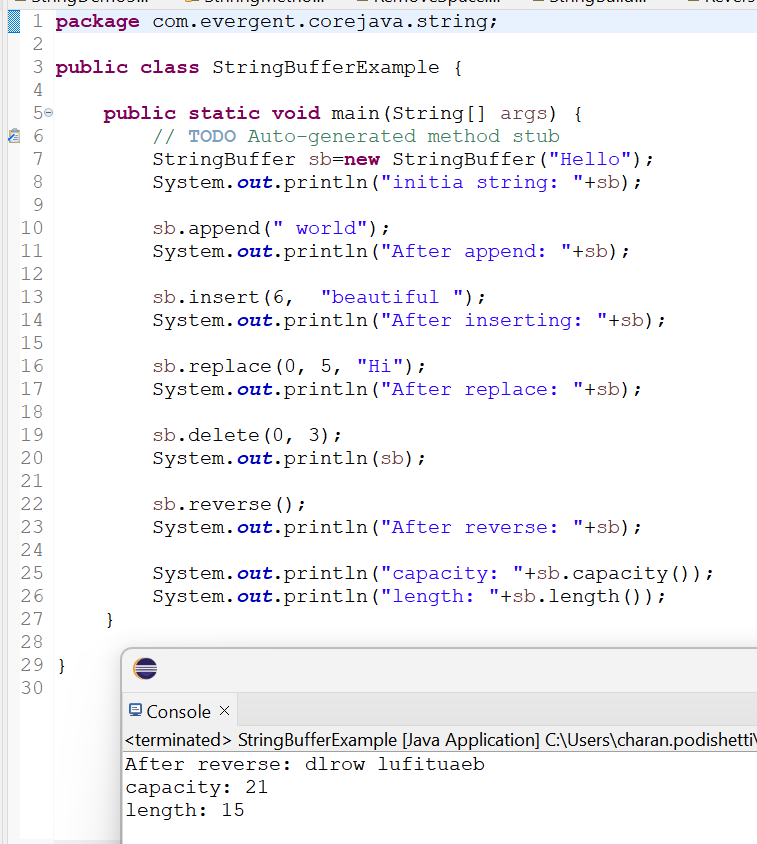


StringBuffer

* StringBuffer is a final class.
* StringBuffer are mutable.
* String class consists of methods.
* Methods in StringBuffer class are synchronized.

1. StringBuffer methods(append()-used to combine the two strings into one String,

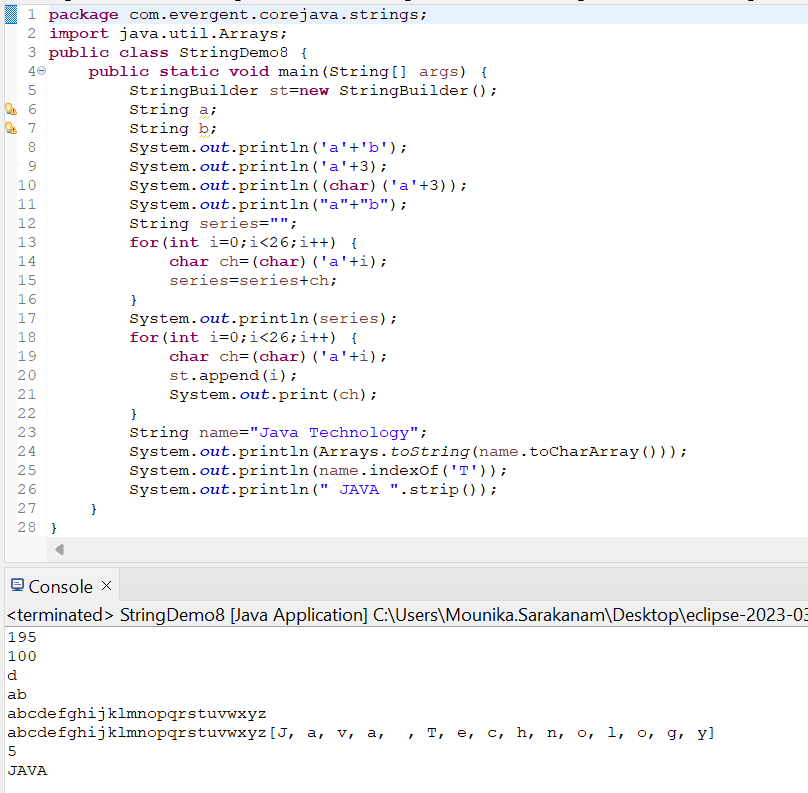
Insert-to insert the string into the present string based on the index value,reverse-to reverse the string ,replace-replace the string with other string based on starting index and ending index,delete-to delete the sub string or string from the existing string based on starting index and ending index.)



StringBuilder

* String Builder is a final class.
* StringBuilder are mutable.
* StringBuilder class consists of methods.
* Methods in String class are non-synchronized.

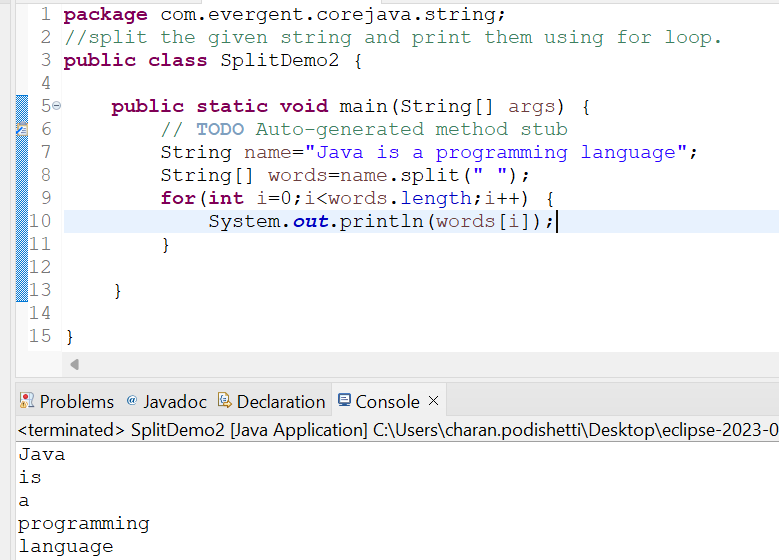
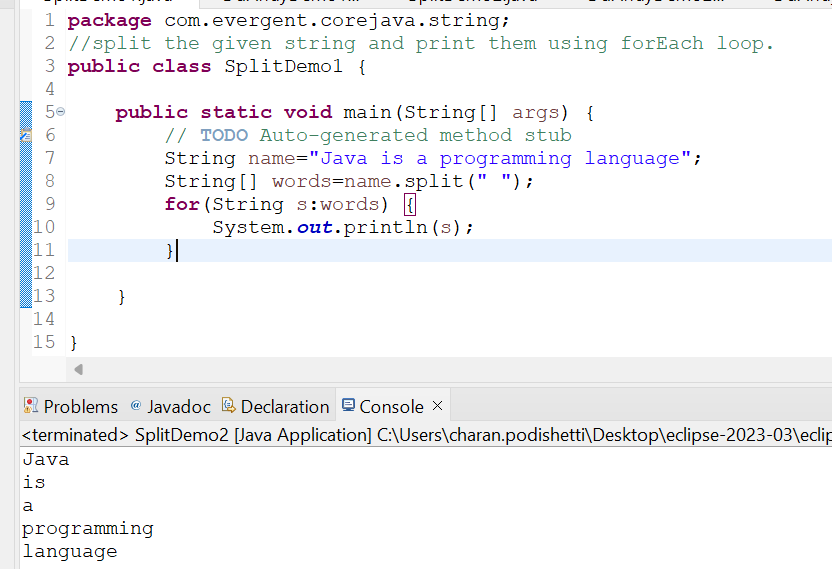
1. 1)StringBuilder methods(append()-used to combine the two strings into one String,Insert-to insert the string into the present string based on the index value,reverse-to reverse the string ,replace-replace the string with other string based on starting index and ending index,delete-to delete the substring or string from the existing string based on starting index and ending index.)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Date 09/08/2024**

**Day-07**

* **Split method program and printing them using for loop.**
* ****
* **Split method program and printing them using forEach loop.**
* ****

**Immutable class:**

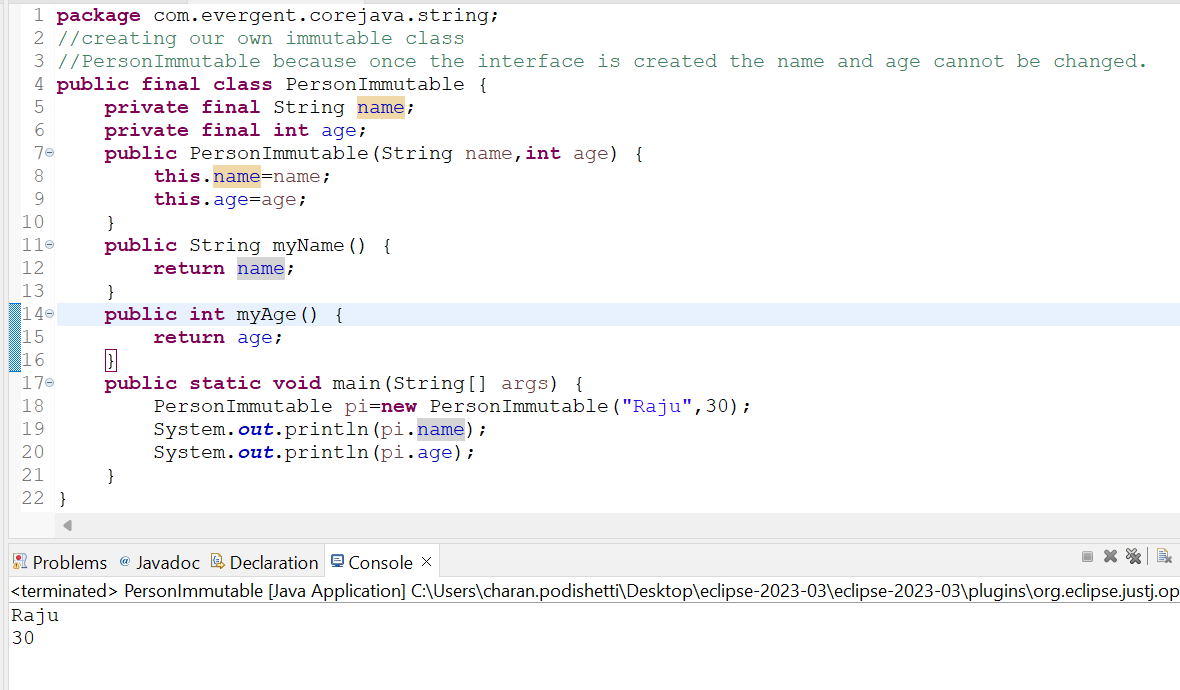
1. We can declare class as final.
2. The class is declared as final so that it cannot be subClassed.

**Private final fields:**

* The fields name and age are private and final.

**Constructor:**

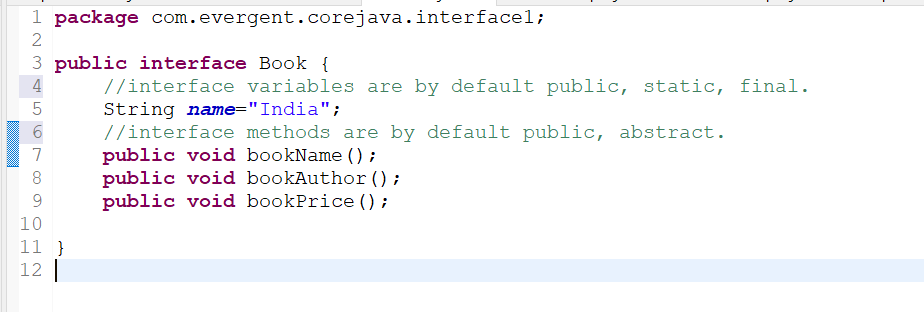
* The constructor initialize the final fields when a person objects is created.

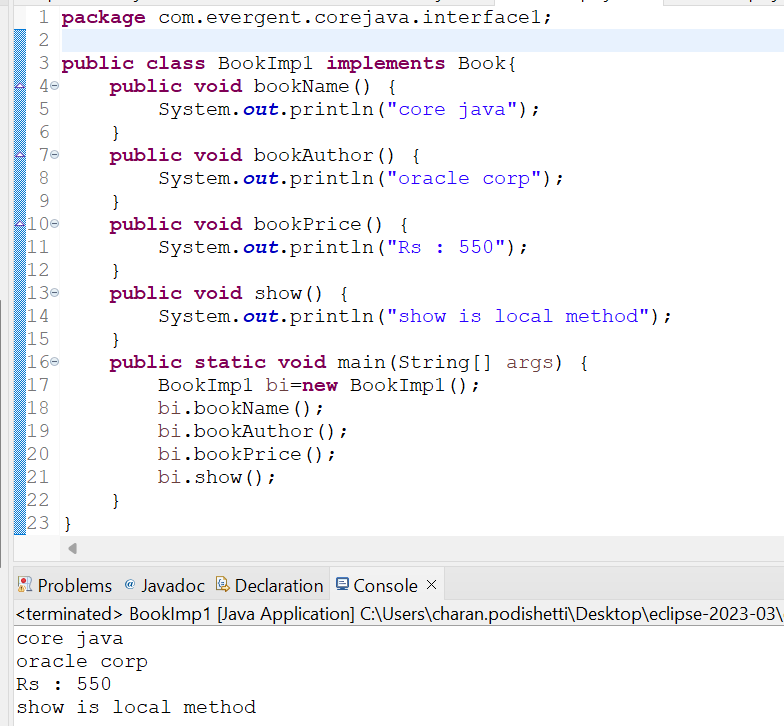


**Interface:**

* Interface is a keyword.
* We can declare method signature only but not the implementation.
* By default all interface methods are abstract.
* If any class implements interface the class should be override all interface methods otherwise the class will be showing compile time error.
* We cannot create object to interface, but we can create reference to interface.
* We can declare variables inside interface, all interface variables are by default public, static, final.
* Java will support multiple inheritance through interface.
* One class can implements interface.
* One interface can extends other interfaces.
* We can declare interface without methods or zero methods.
* Markable interface are clonable, serializable.

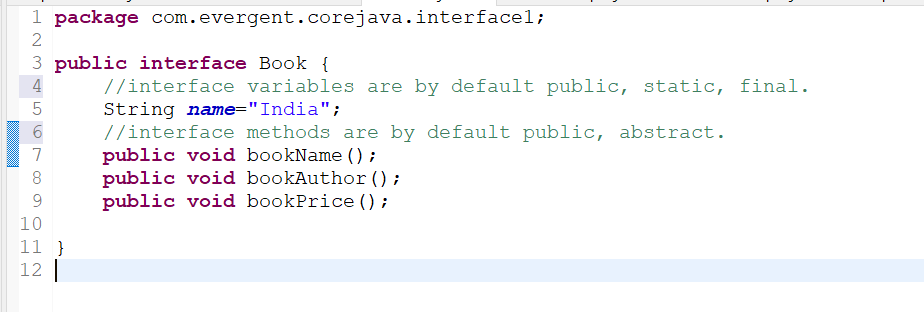
Example 1: program covering first three points.

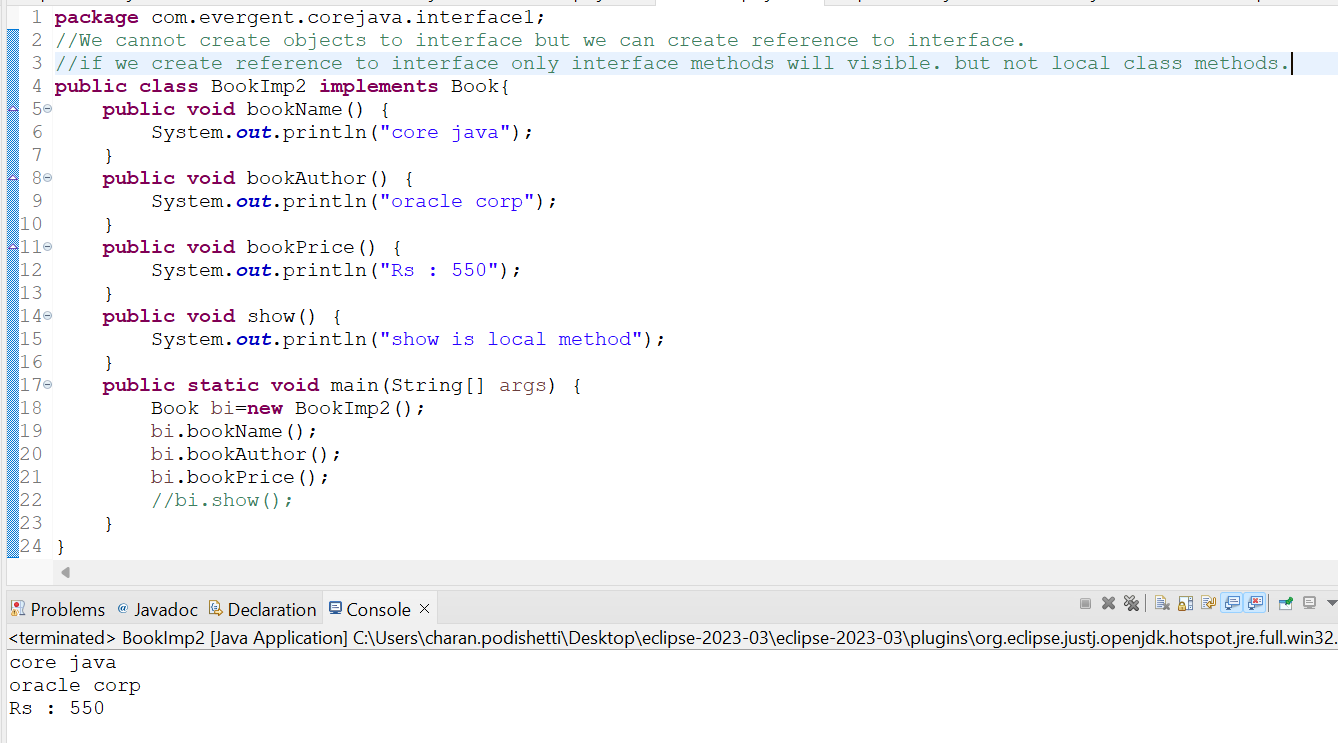




Example 2: We cannot create object to interface but we can create reference to interface.

If we create reference to interface only interface methods will visible.





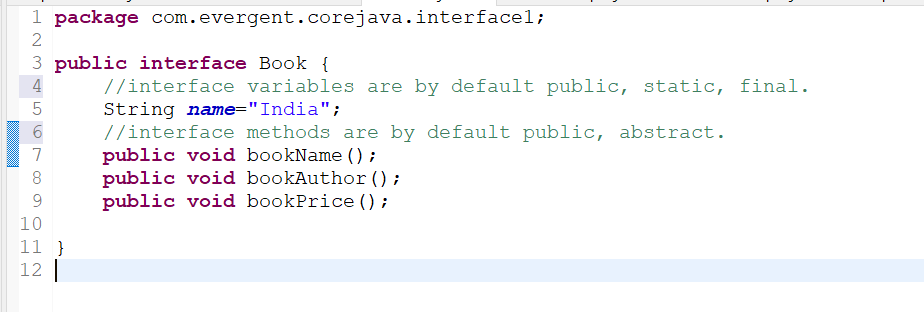
Example 3:

Interface supports multiple inheritance.

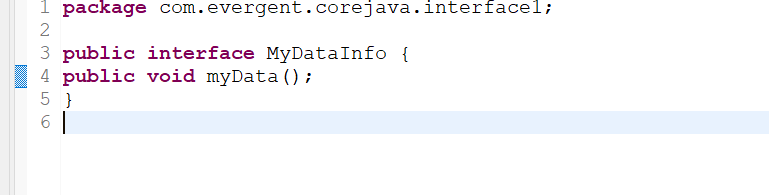
One class can implements interface.

One interface can extends other interface.

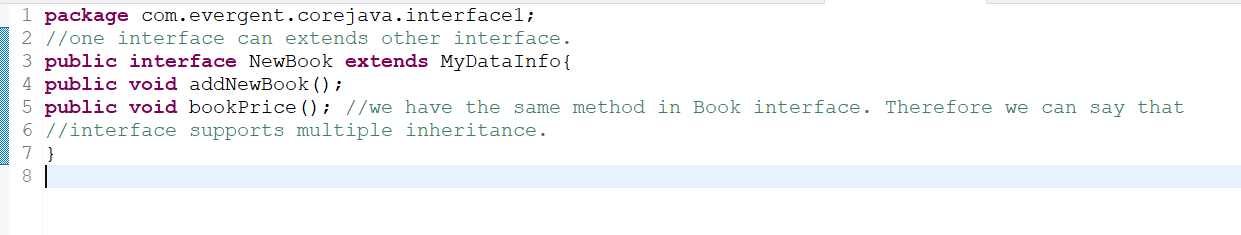
1. Creating one interface.



2.Creating another interface.

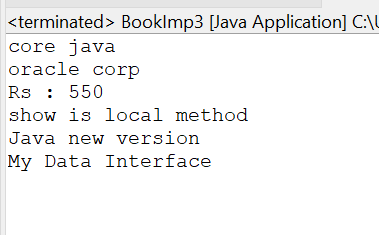


1. One interface extends other interface.





Output:



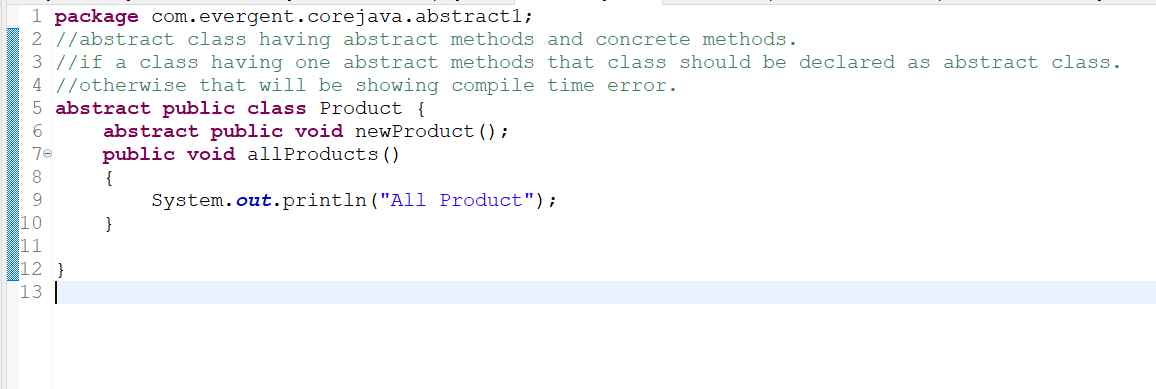
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

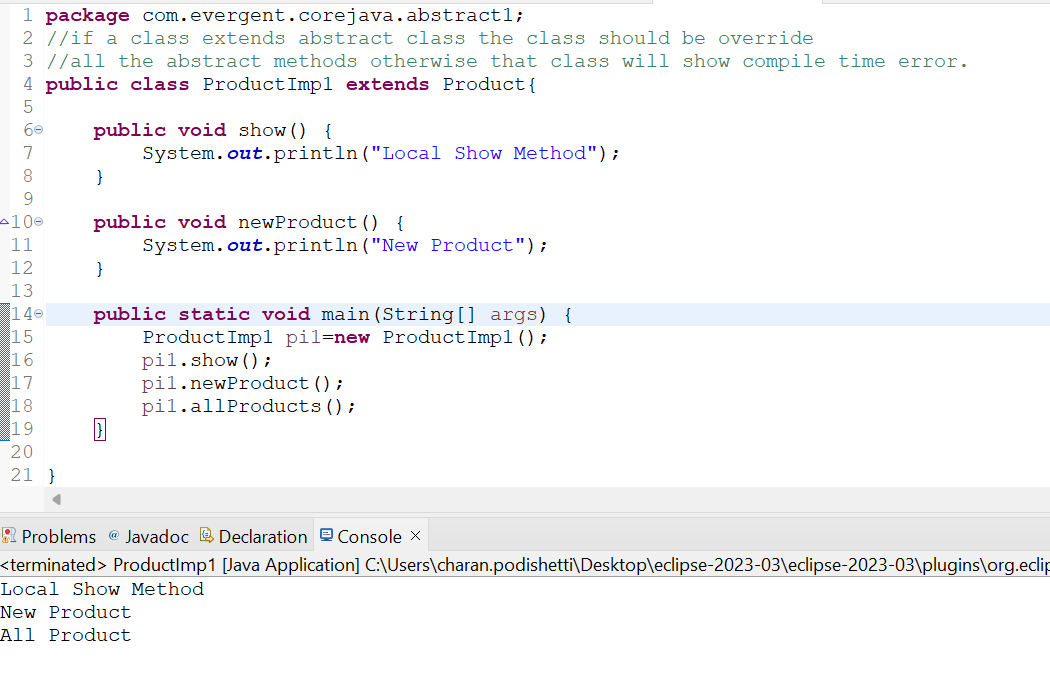
**Date 14/08/2024**

**Day-08**

**ABSTRACT CLASS:**

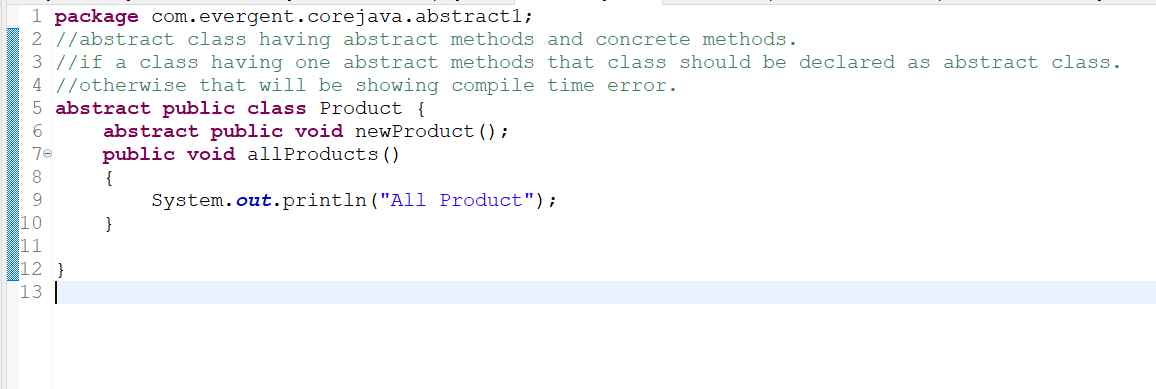
1. Abstract is a keyword.
2. Abstract class having abstract methods and non-abstract methods.
3. If any class having one abstract methods that class should be declared as a abstract keyword otherwise that will be showing compile time error.
4. If any class extends abstract class the class should be override all abstract methods otherwise that class will show compile time error.
5. We cannot create object to abstract class, but we can create reference to abstract class.
6. We can create constructor to abstract class.
7. We can access abstract class constructor through sub class object creation.

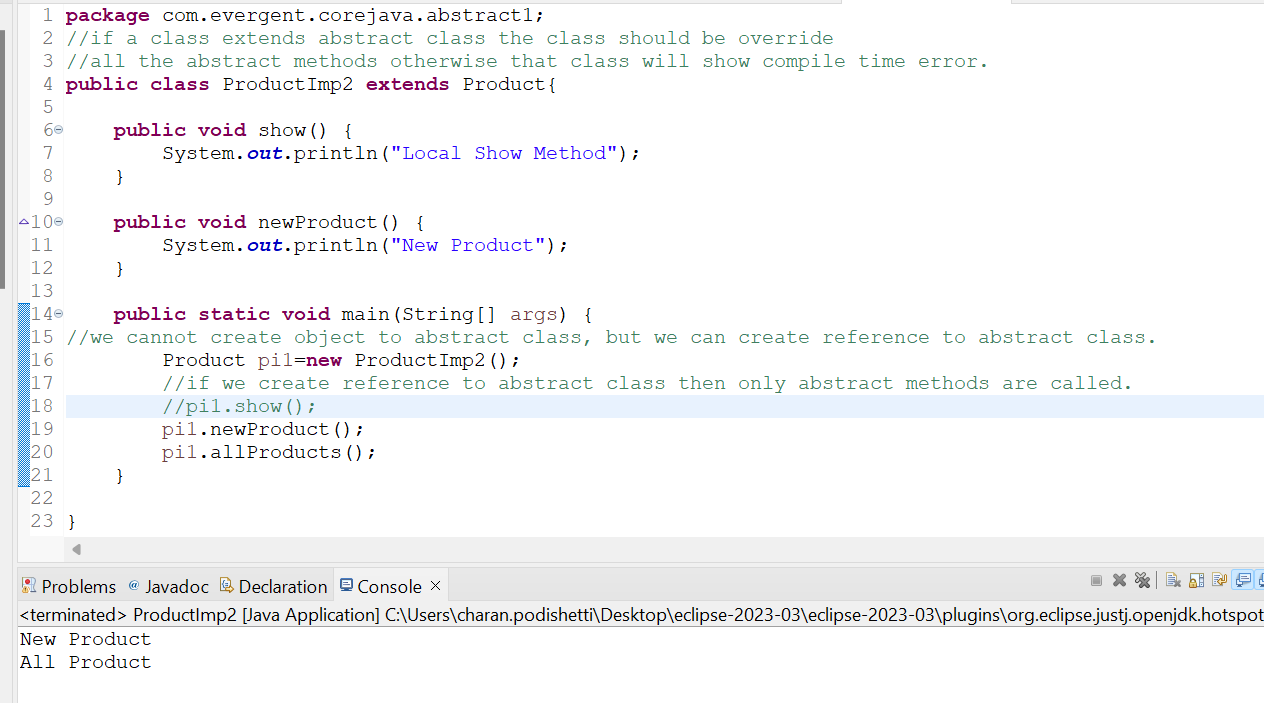




Program on, we cannot create object for abstract class, but we can create reference for abstract class.

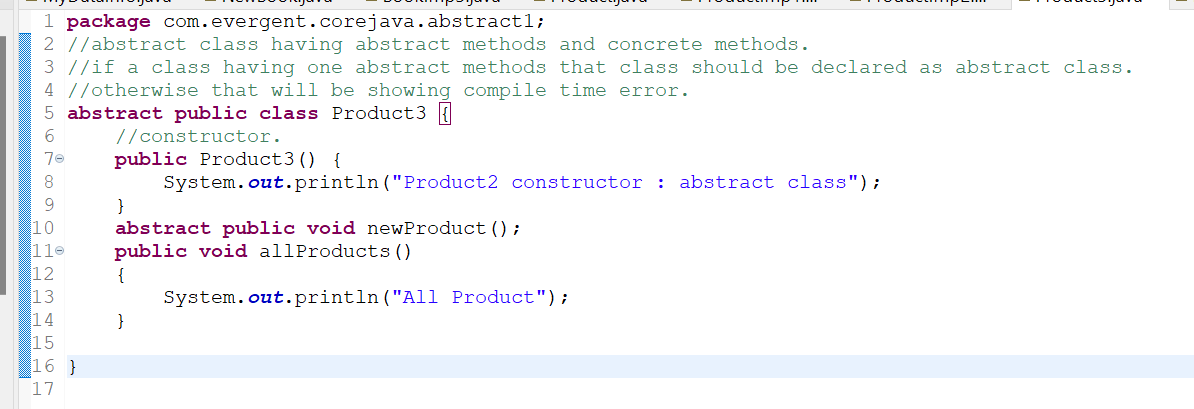
If we create reference for abstract class then all abstract class methods should be override to sub class otherwise it will give compile time error.





Program on, We can create constructor to abstract class.

We can access abstract class constructor through sub class object creation.





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

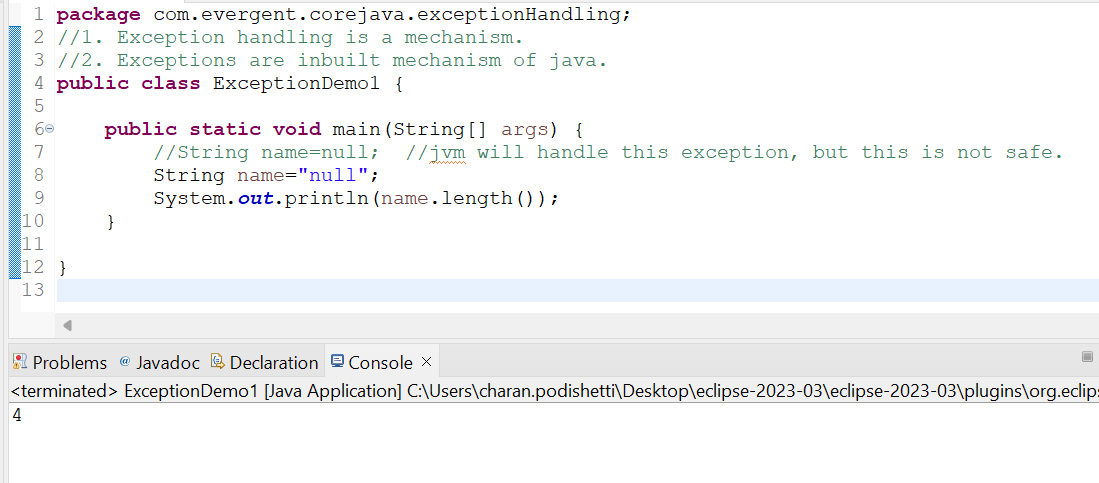
**Date 19/08/2024**

**Day-10**

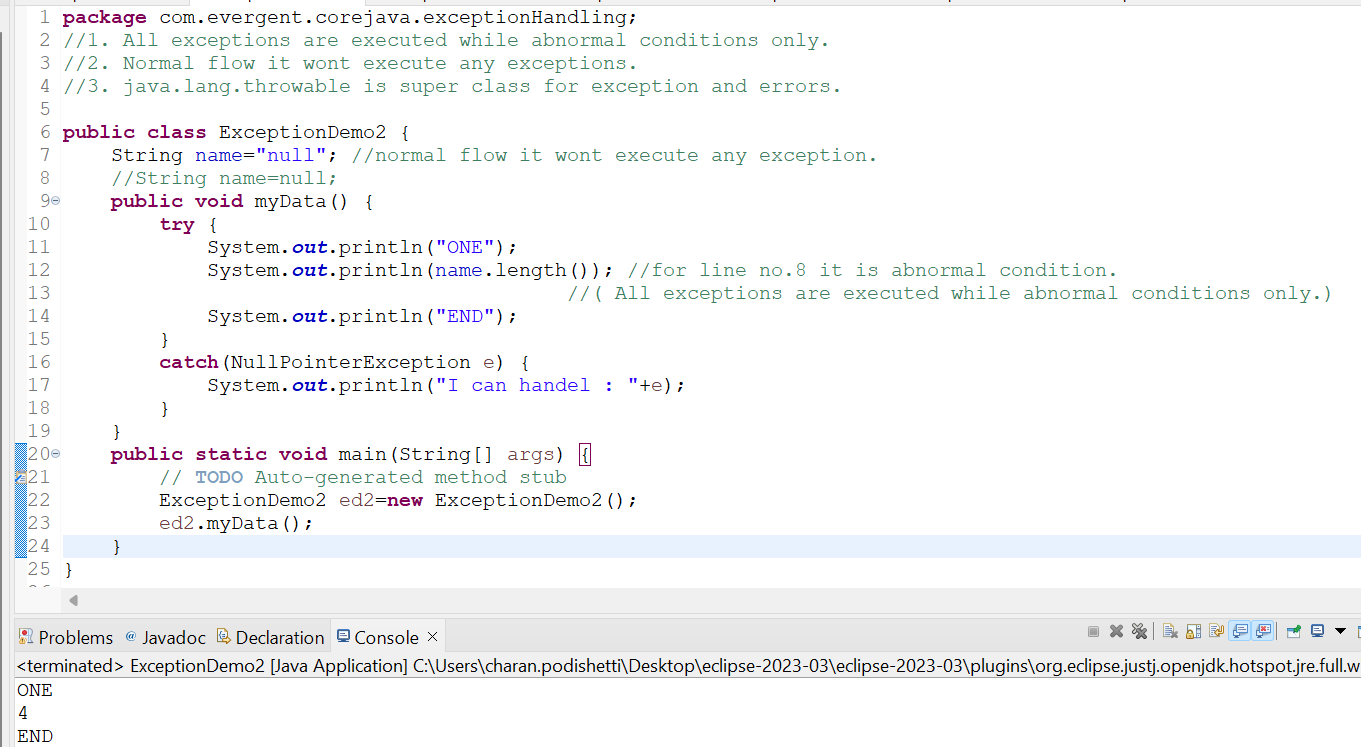
**Exception Handling:**

1. Exception handling is a mechanism.

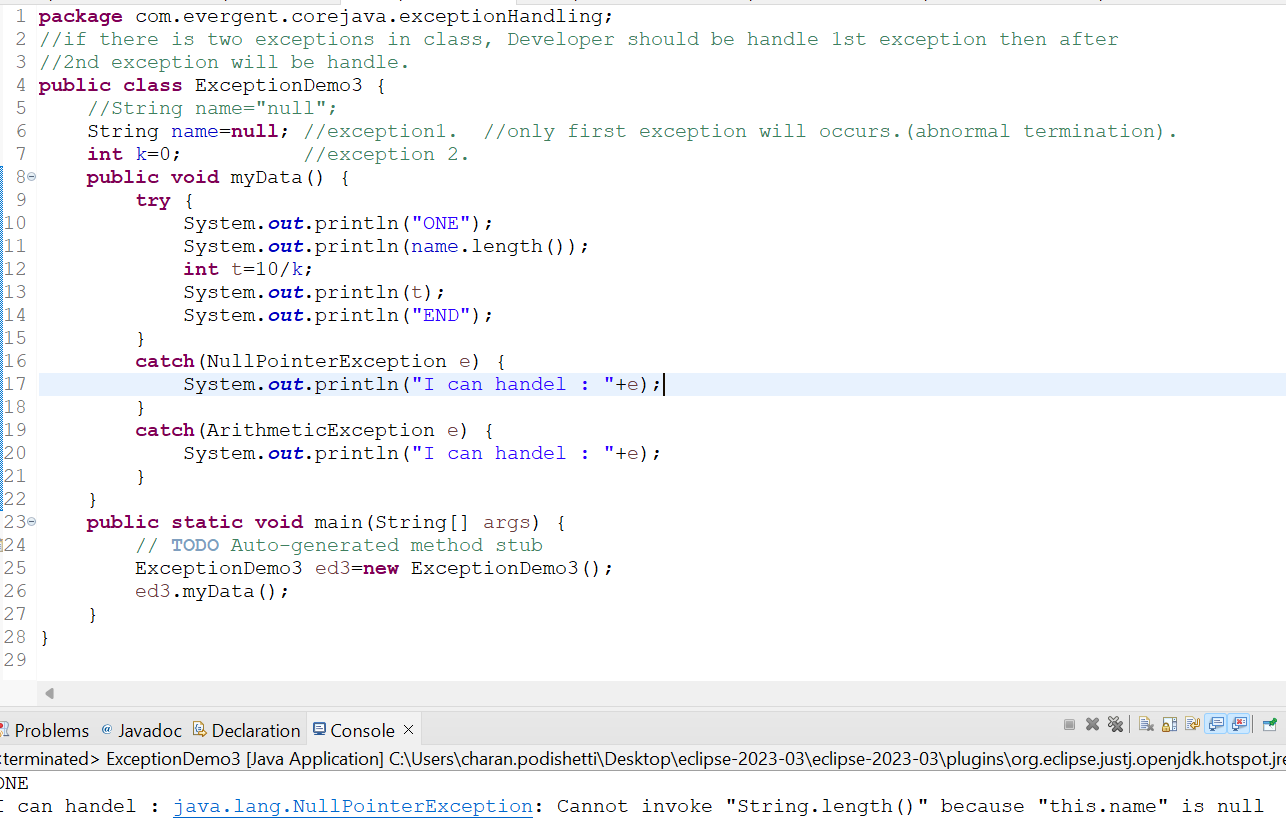
2. Exceptions are inbuilt mechanism of java.



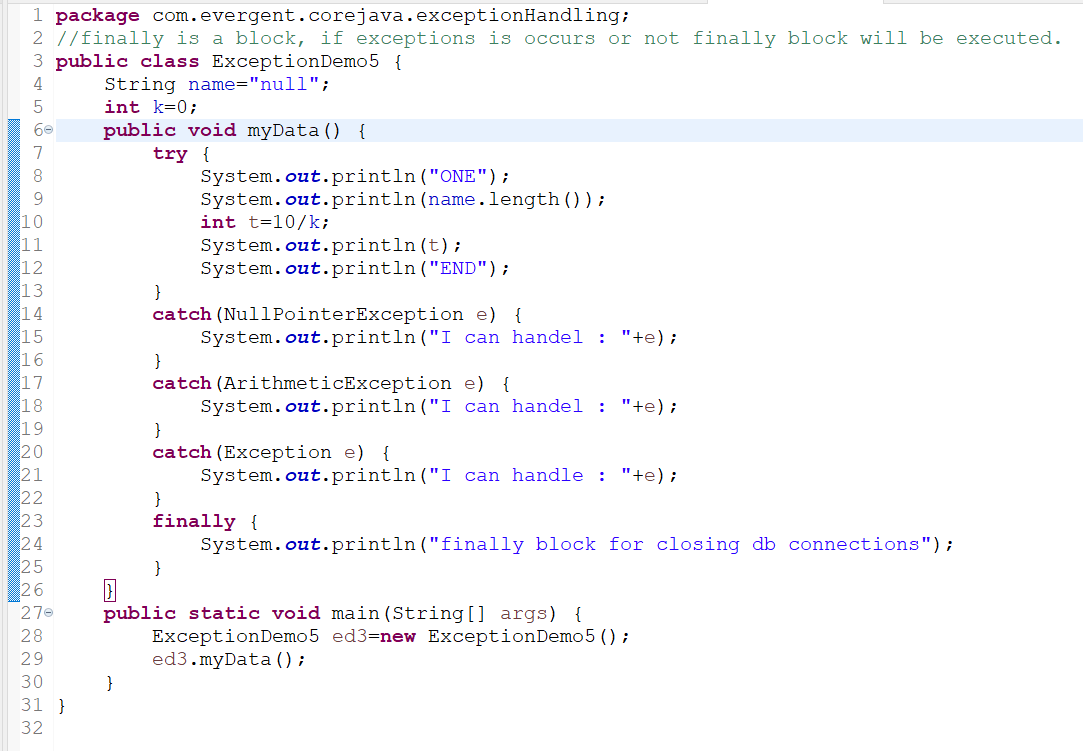
1. All Exceptions are executed while abnormal conditions only.
2. Normal flow it wont execute any exceptions.
3. Java.lang.throwable is super class for exception and errors.



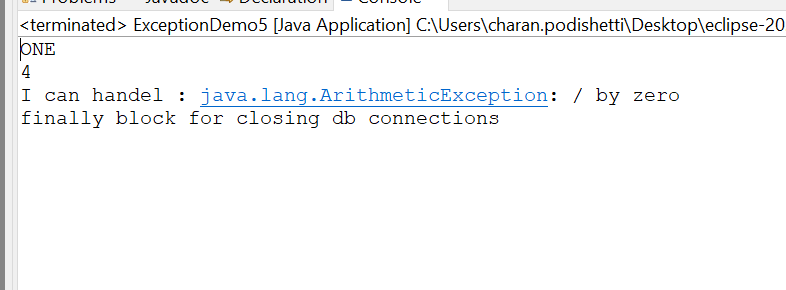
1. if there is two exceptions in class, Developer should be handle 1st exception then after 2nd exception will be handle.



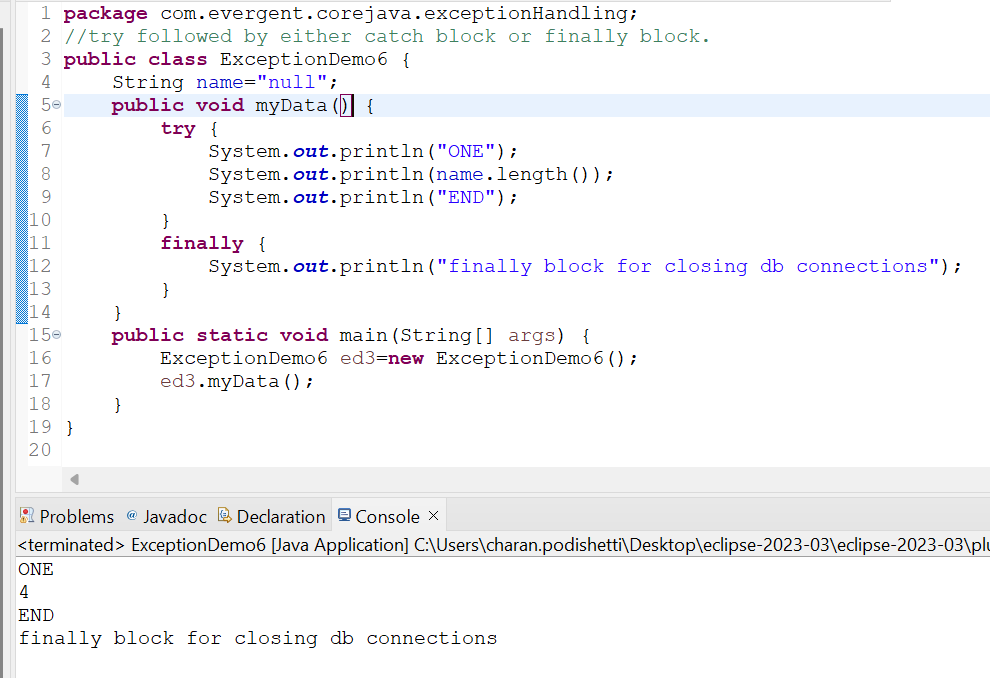
1. finally is a block, if exceptions is occurs or not finally block will be executed.



OUTPUT:



1. Try followed by either catch block or finally block.



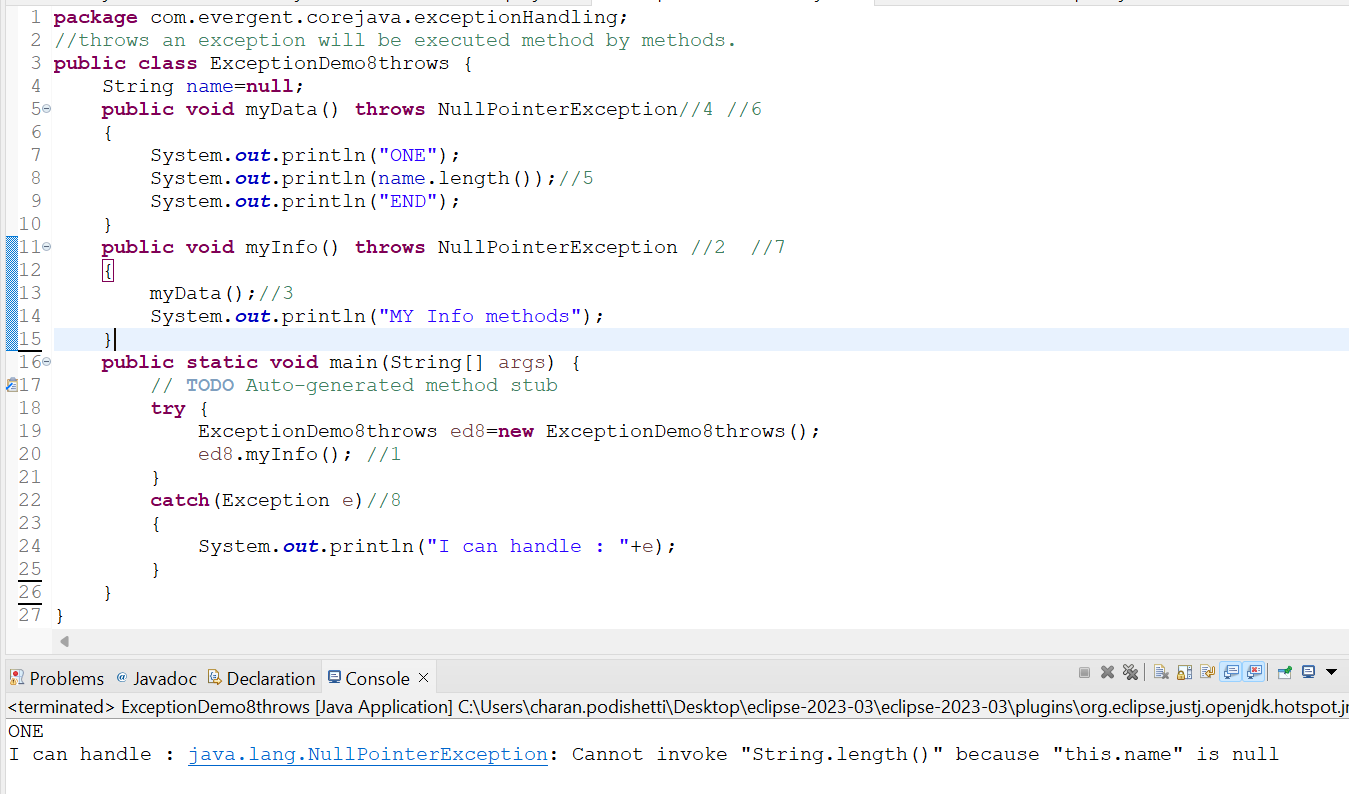
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Date 20/08/2024**

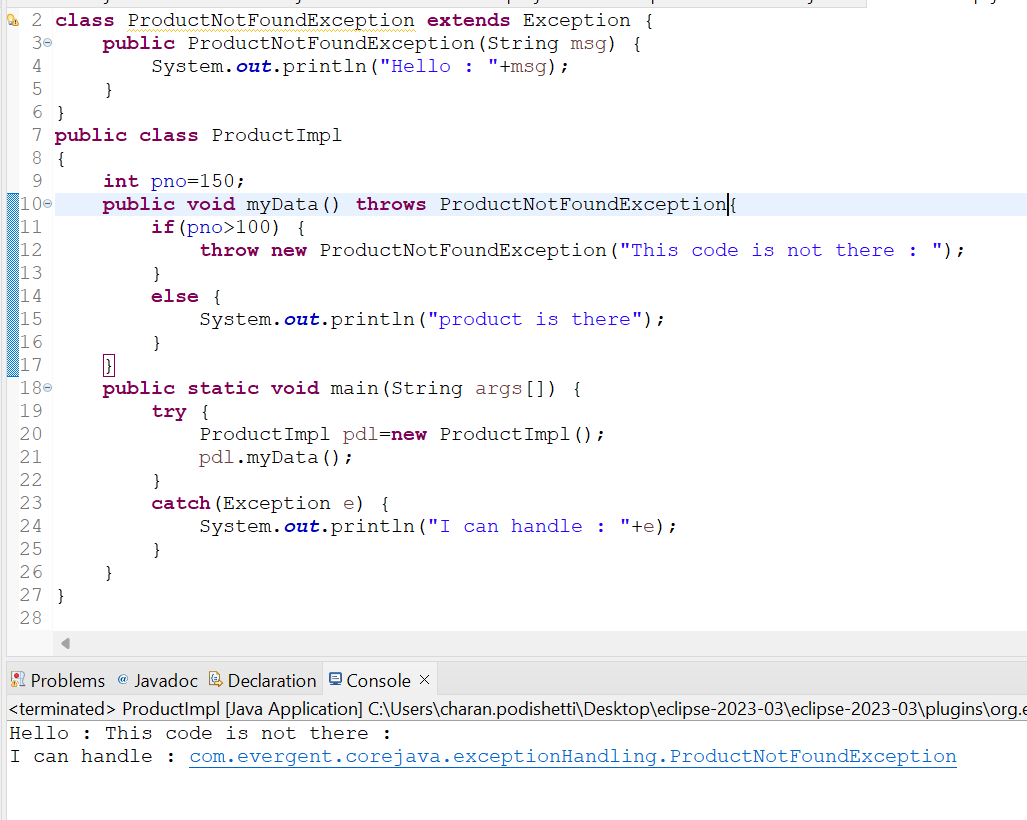
**Day-11**

**Exception Handling:**

1. Throws an exception will be executed method by method.



1. We can create our own (userDefined) Exception.

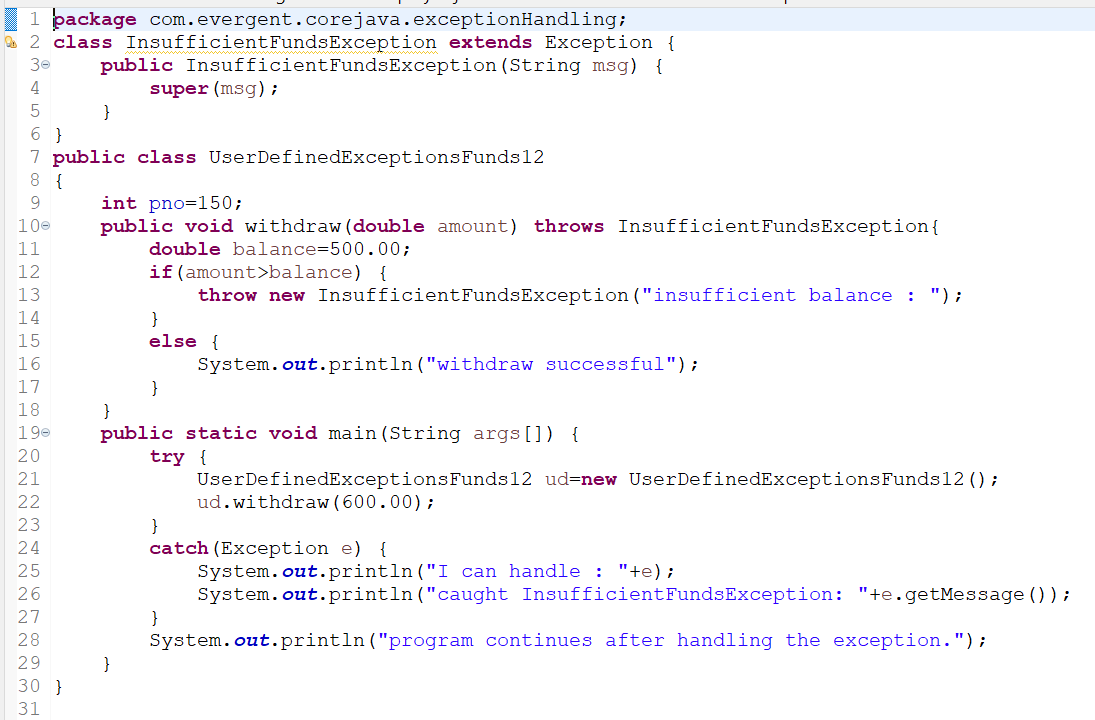


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

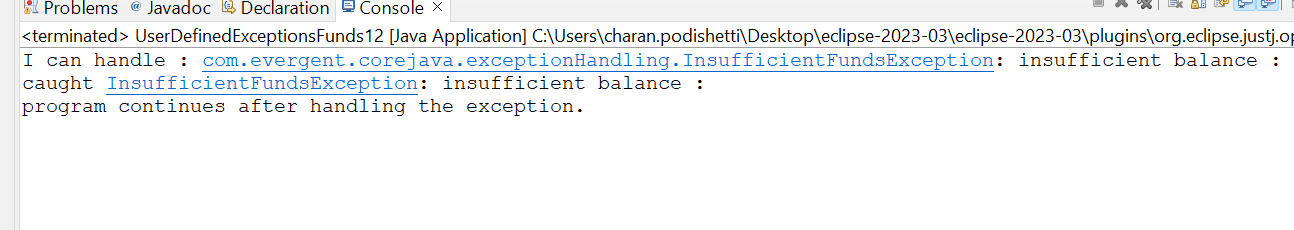
**Date 21/08/2024**

**Day-12**

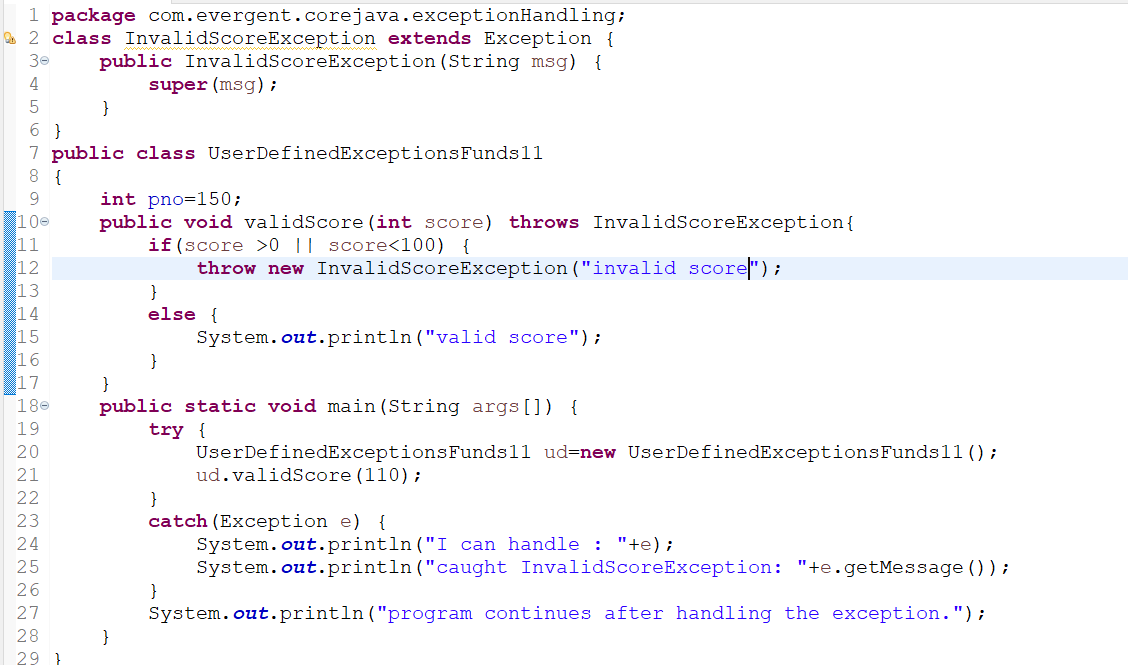
Program for InSufficientFundsException.



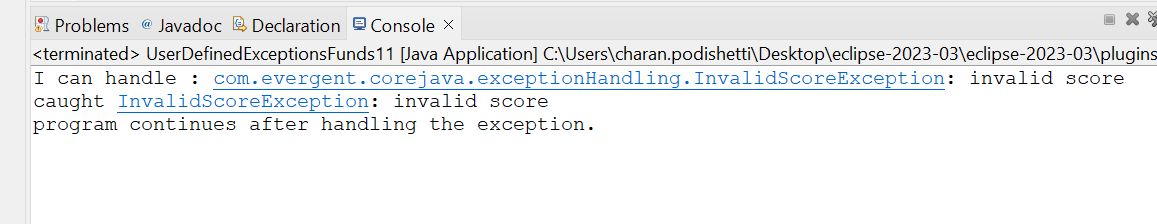
Output:



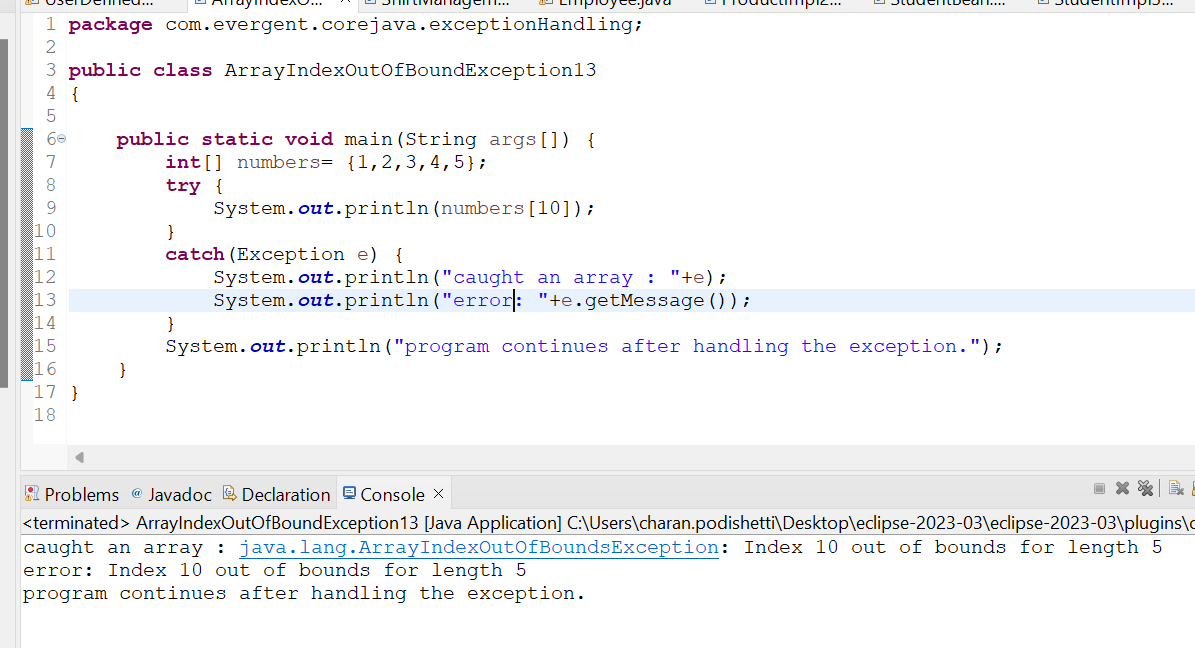
Program for InvalidScoreException.



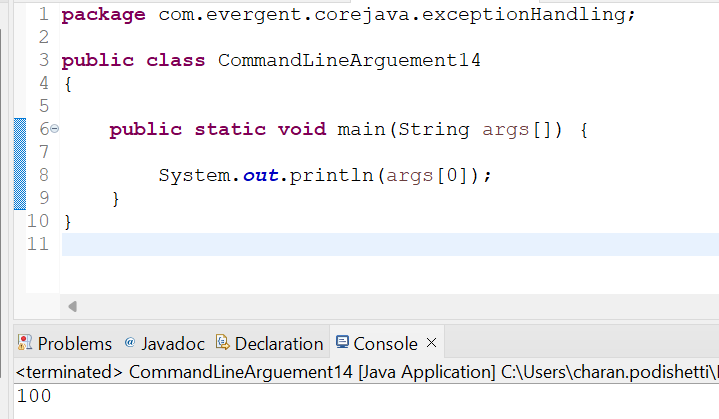
Output:



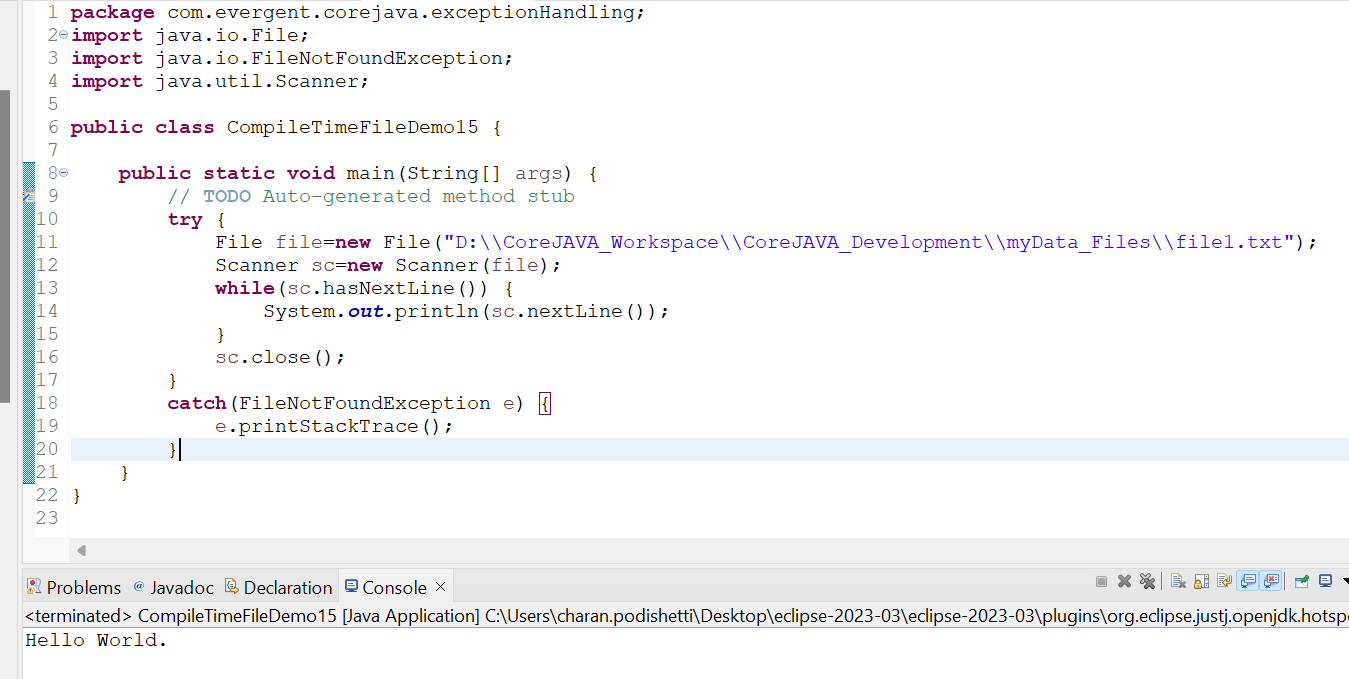
Program on ArrayIndexOutOfBound.



Program on commandLineArguement.



Program on fileNotFoundException.



JAVA BEANS:-

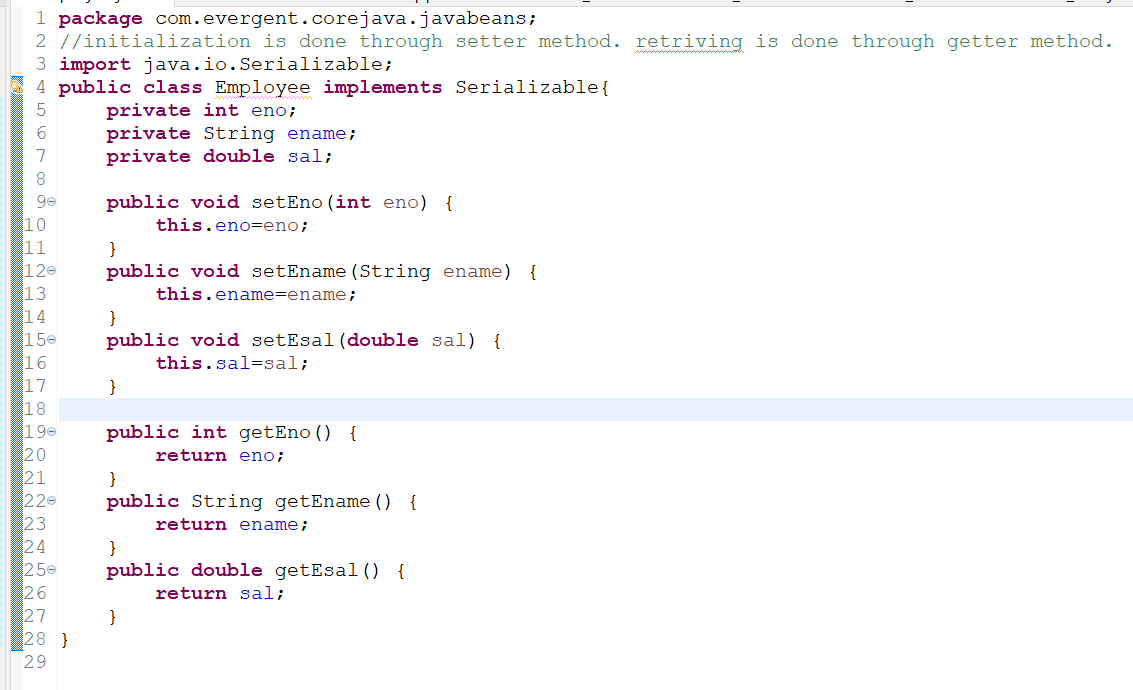
Java bean is a mechanism.

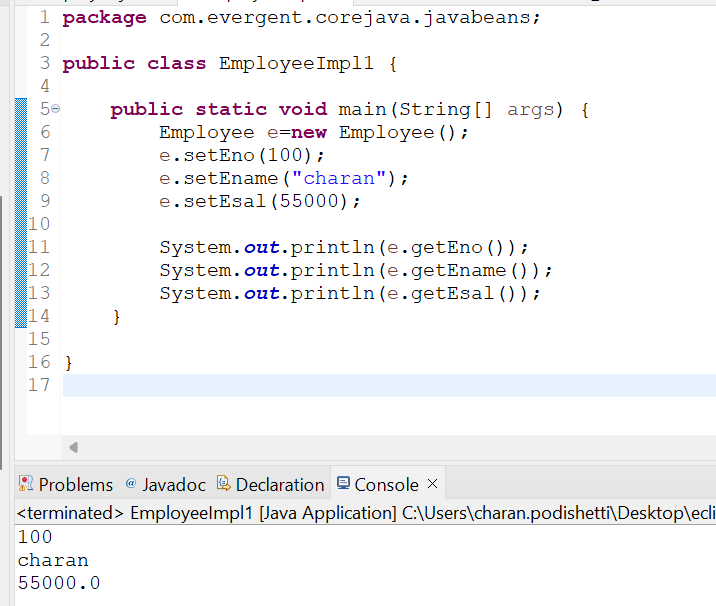
Java bean is a light weight.

All attributes are private.

Get/set methods are implementes java.io.serializable interface.

We can achieve tightly encapsulation through java bean.



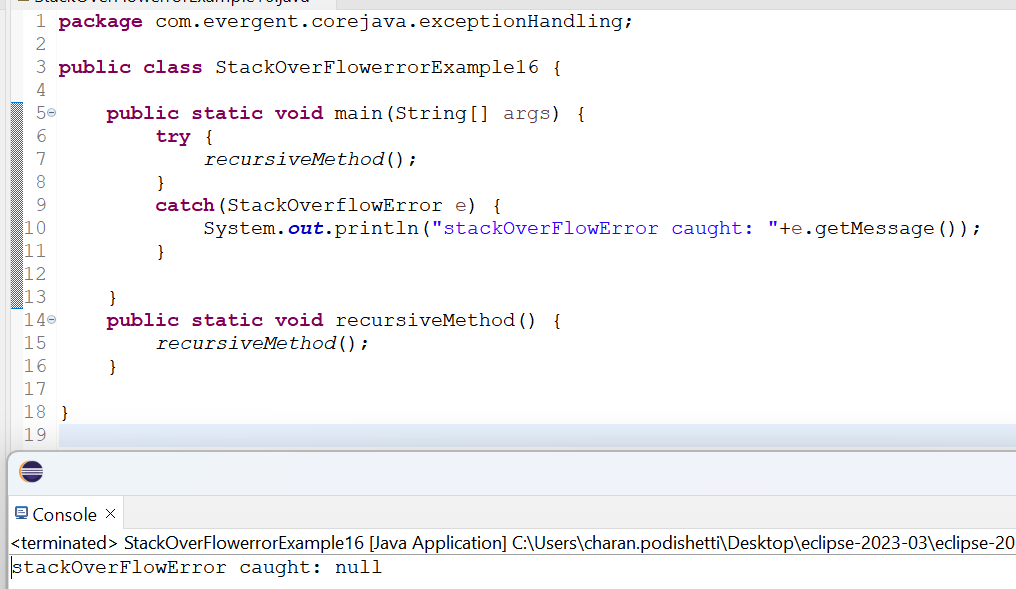


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

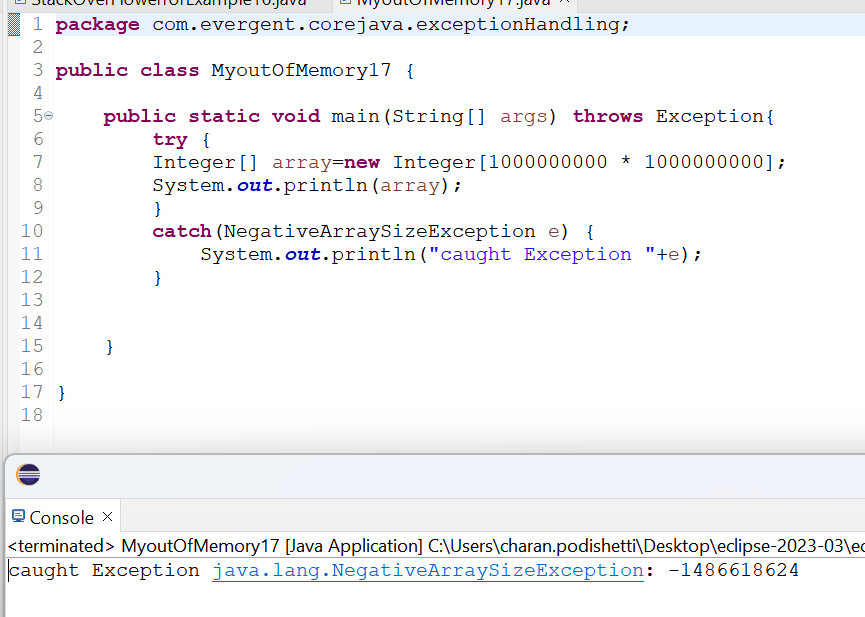
**Date 22/08/2024**

**Day-13**

**Program on Stack overflow.**

****

**Program on out of memory.**

****

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Date 23/08/2024**

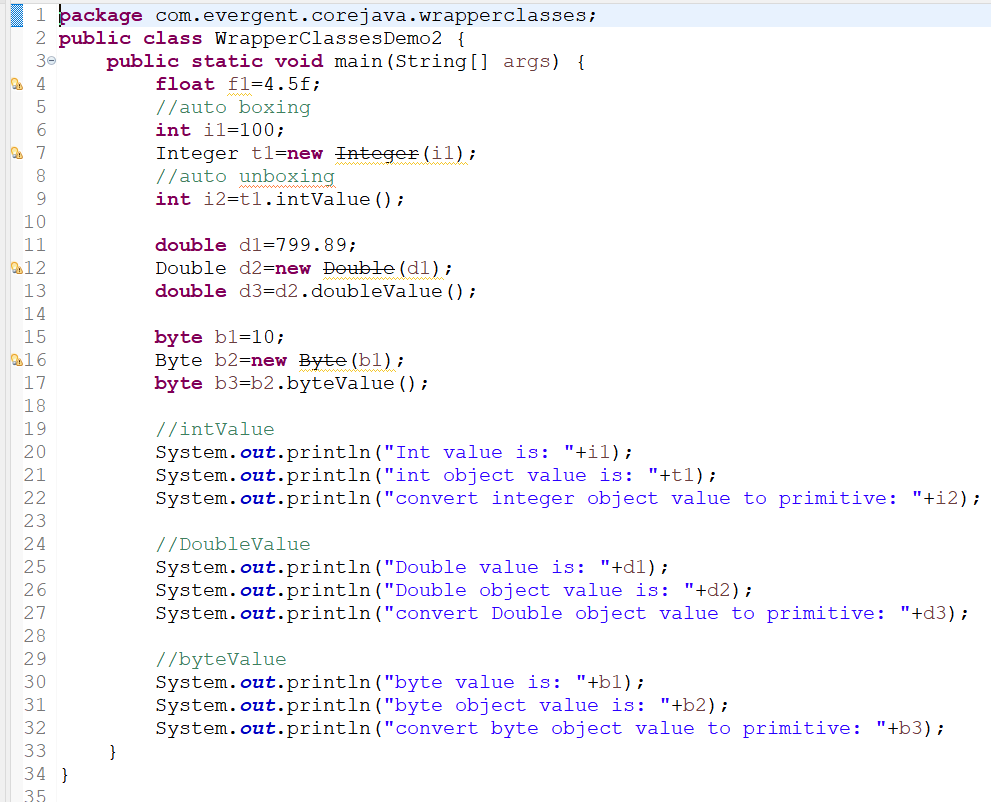
**Day-14**

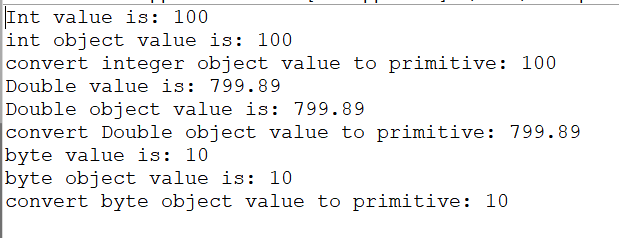
**Wrapper classes:** it provides a way to use the primitive data types as objects.

**Auto boxing:-** it converts the primitive data type to object value.

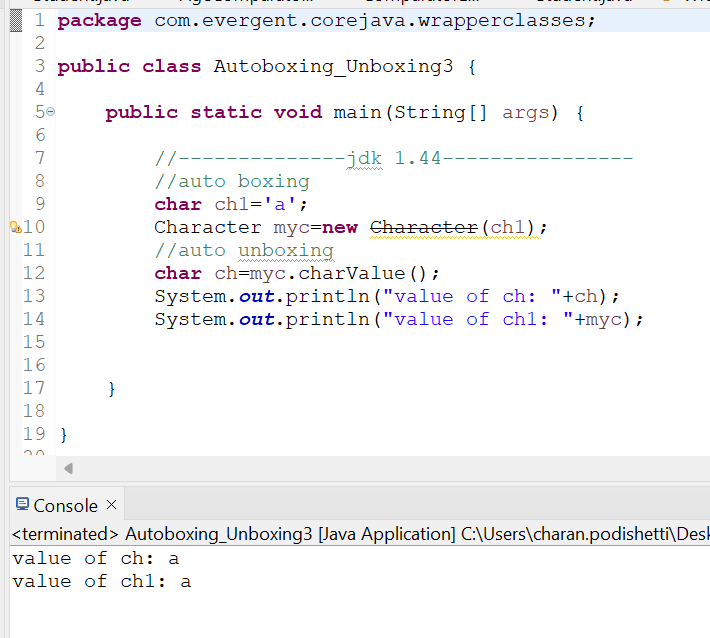
**Auto unboxing**:- it converts the object value to primitive data type.

**Program on wrapper class.**

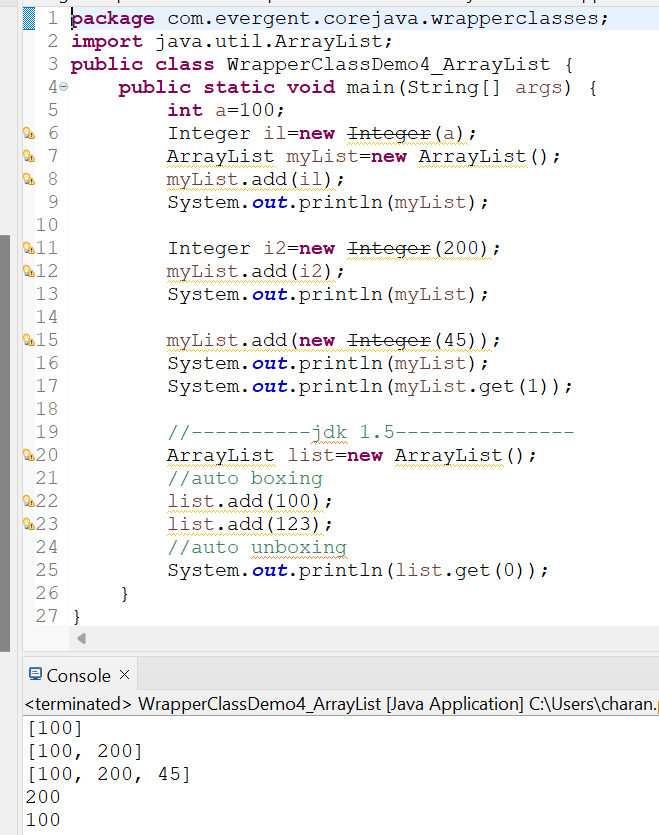
****

****

**Example program for Auto boxing and auto unboxing. In JDK 1.44 version.**

****

**Example program using wrapper class auto boxing and auto unboxing.**

****