**INDEX**

**Date:5/08/2024**

**Day-1**

1.languages and applications

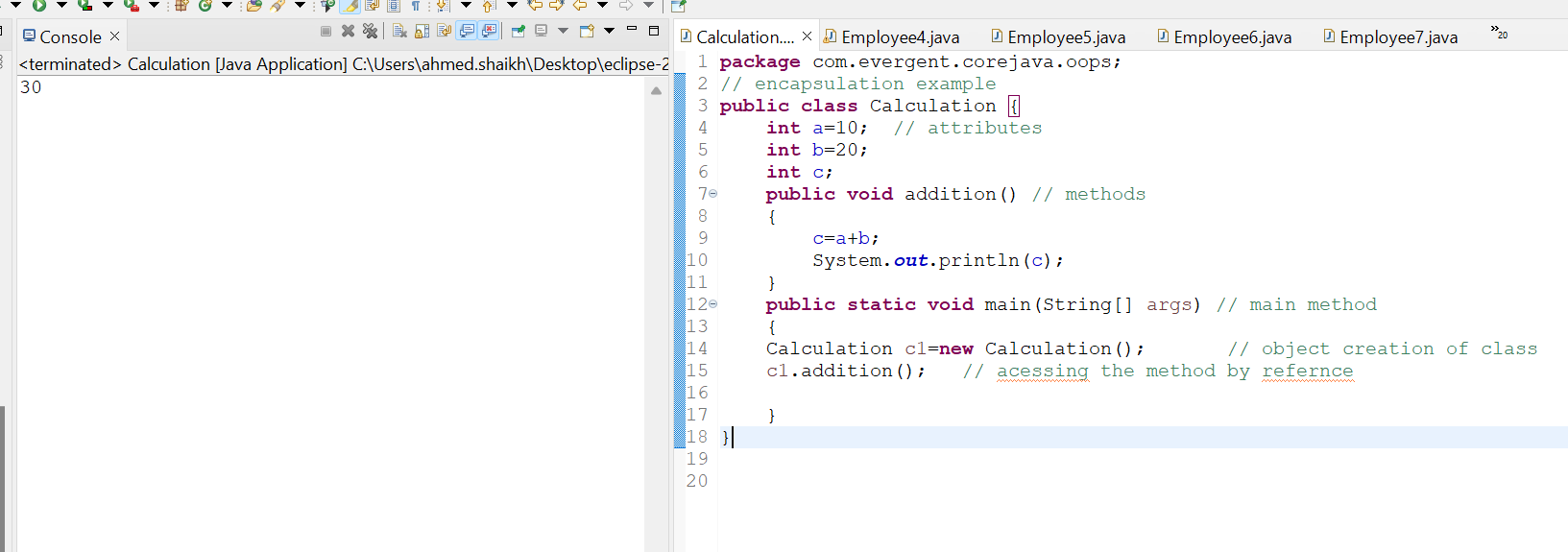
2.Java Features

a.Why Java is platform Independent

b.oops

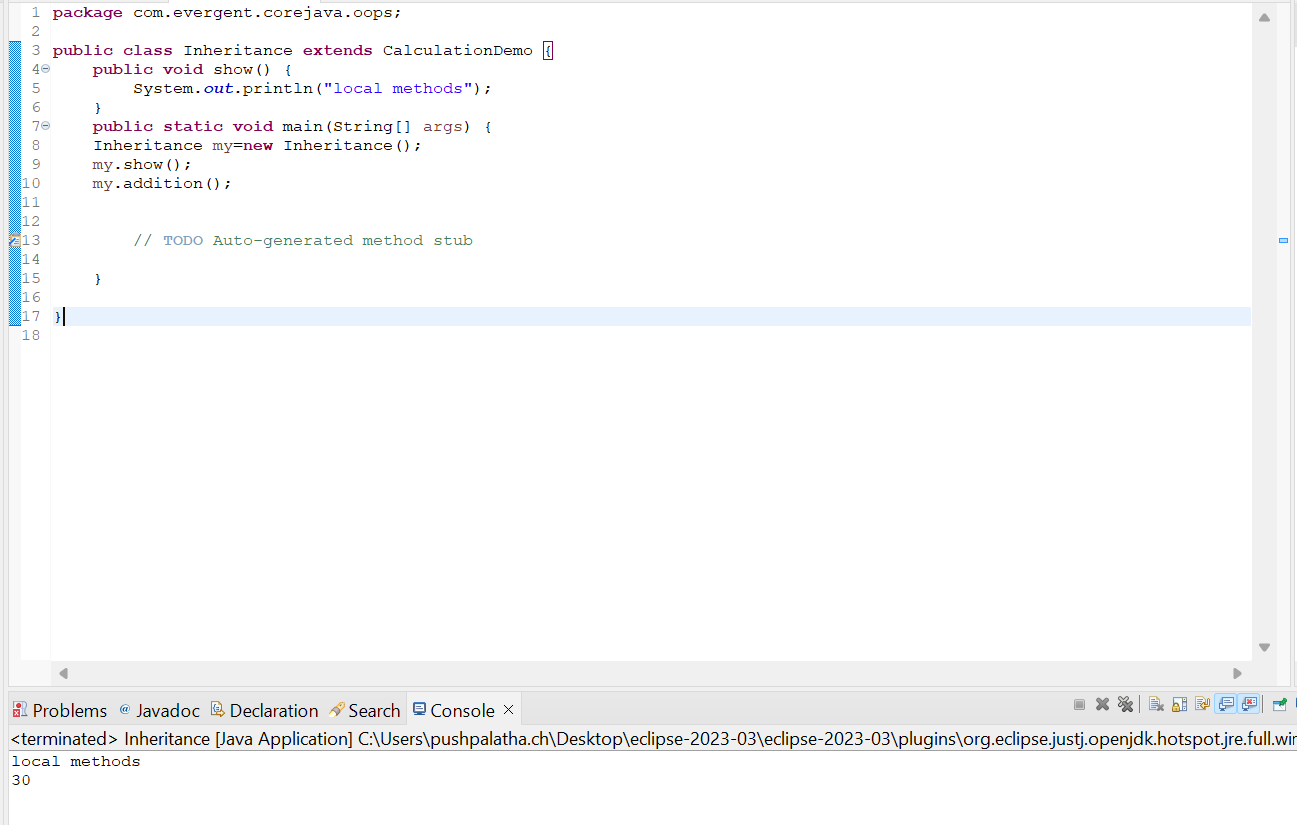
Encapsulation:

It is a mechanism of binding of attribute and method inside an object creation

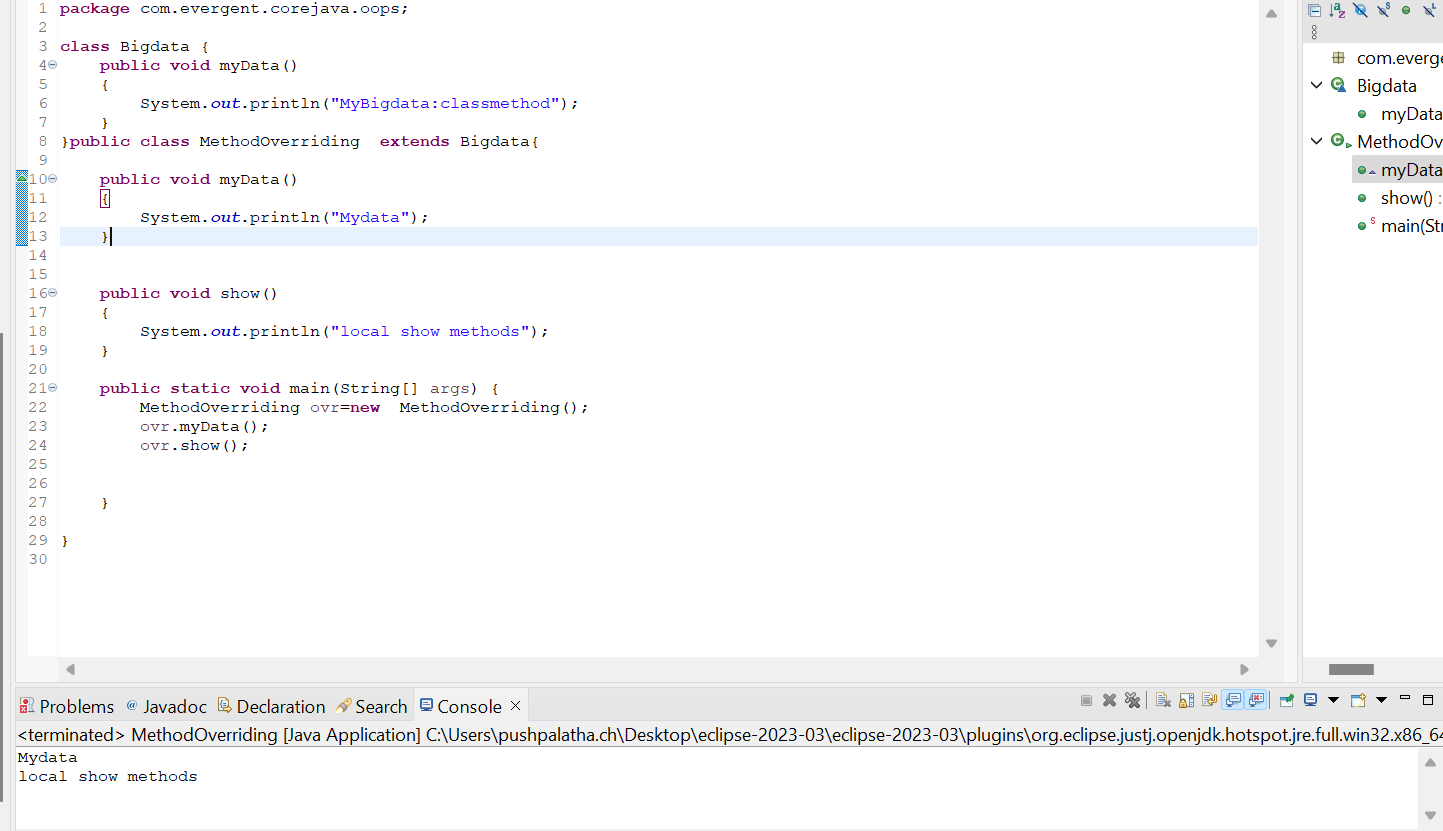


3.Example of encapsultion

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

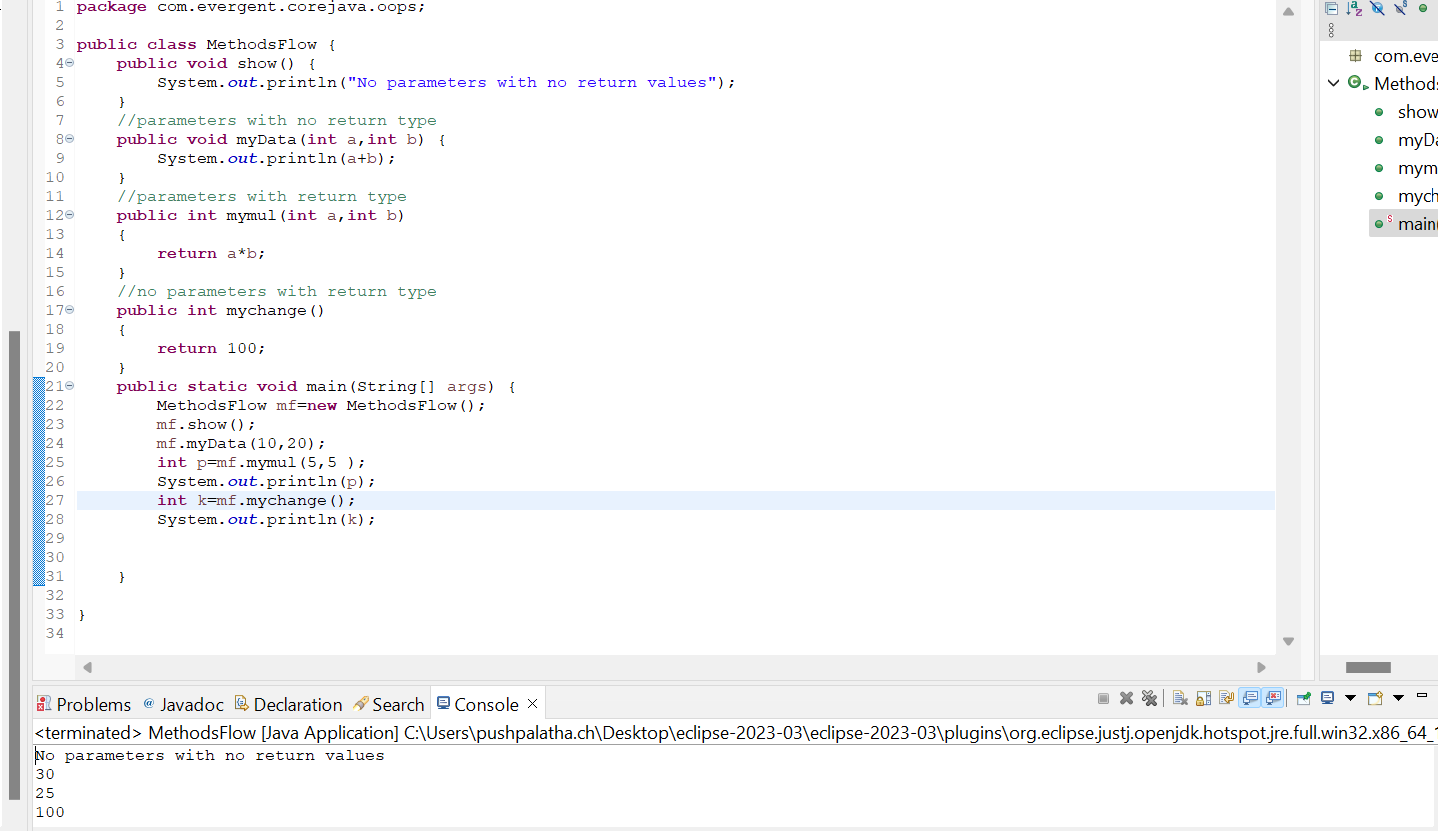
Basically we can call methods in four ways:

1)Methods with parameters without return type.

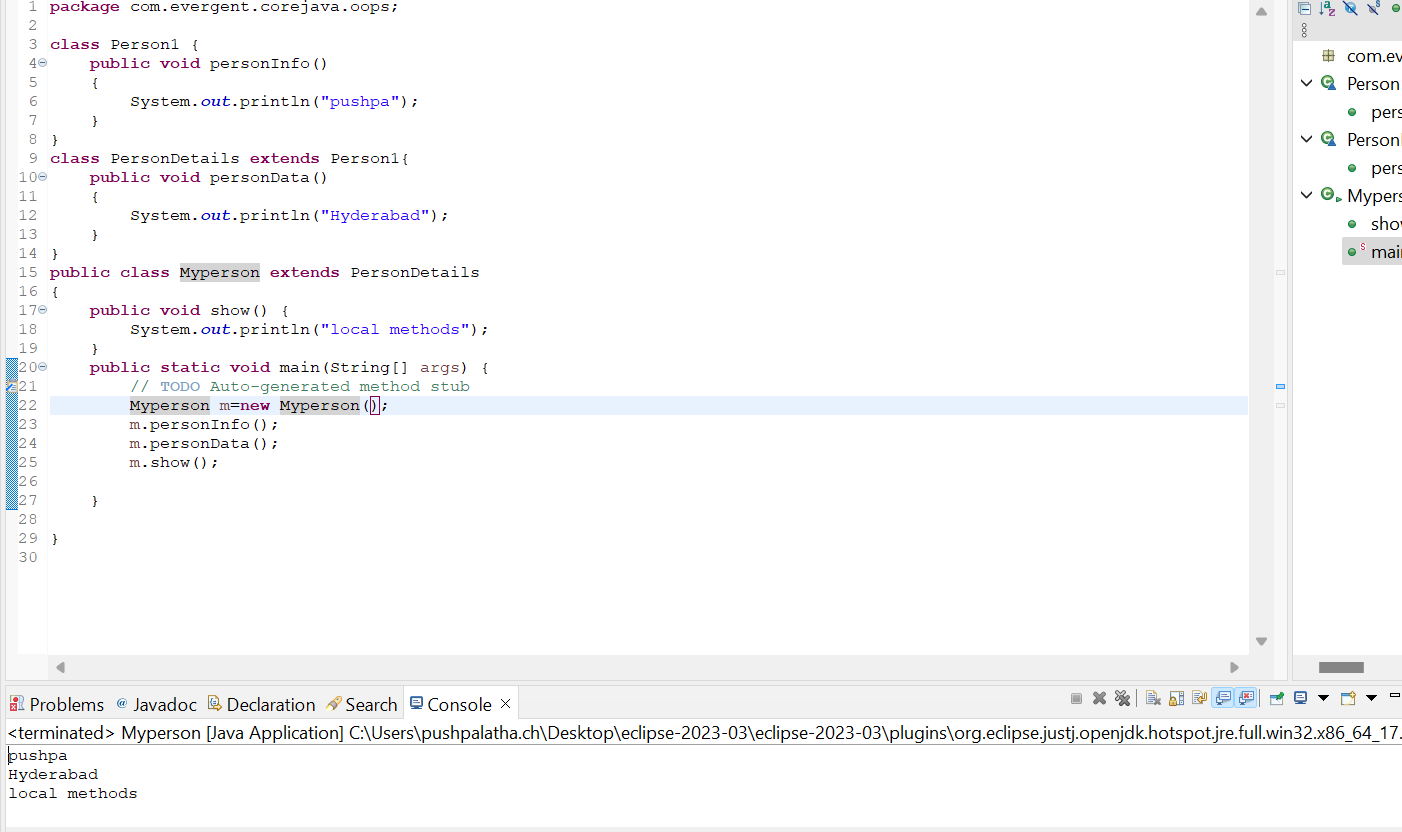
2)Methods with parameters with return type.

3)Methods without parameters without return type.

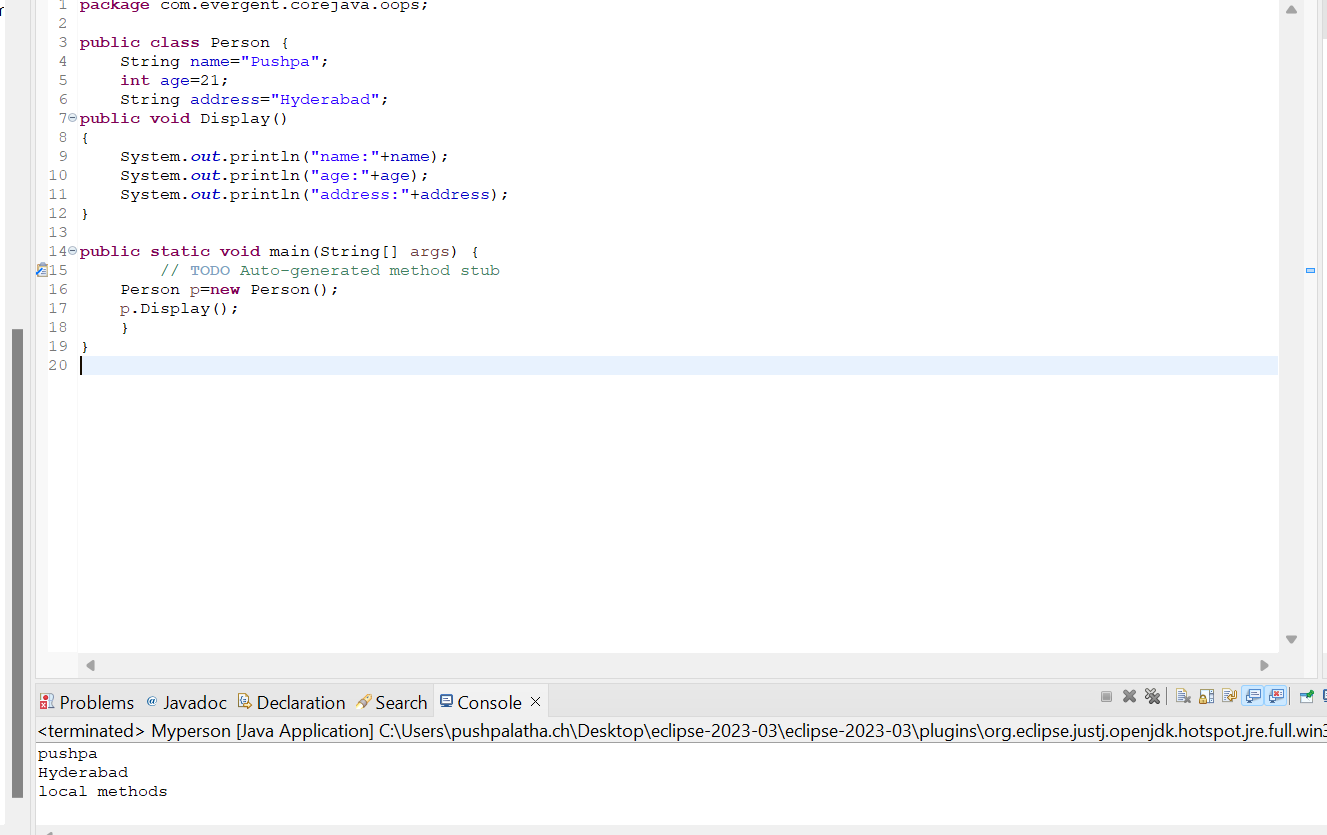
4)Methods without parameters with return type.



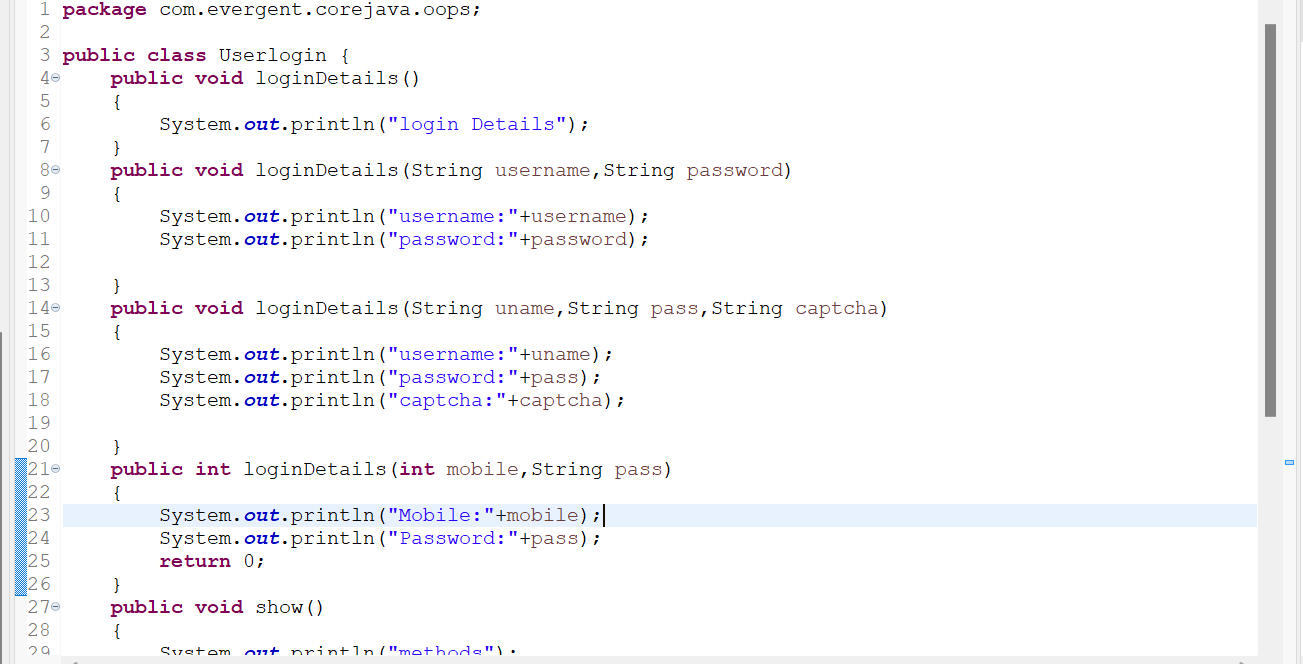
6.Multi Level Inheritance

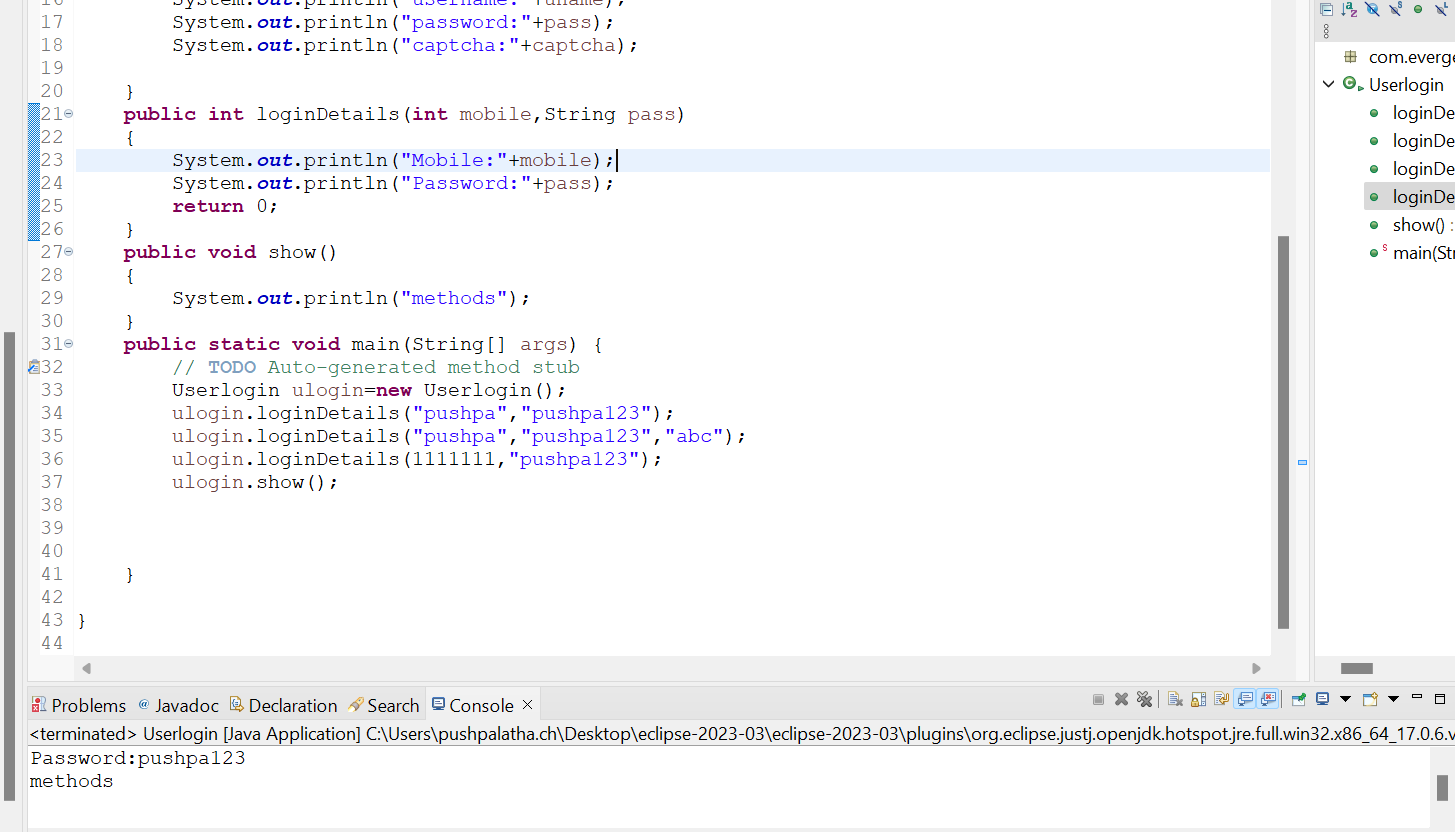


1. OOPS concept Example



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





c.Exception handling

d.Multithreading

e.web application

f.open source

g.security

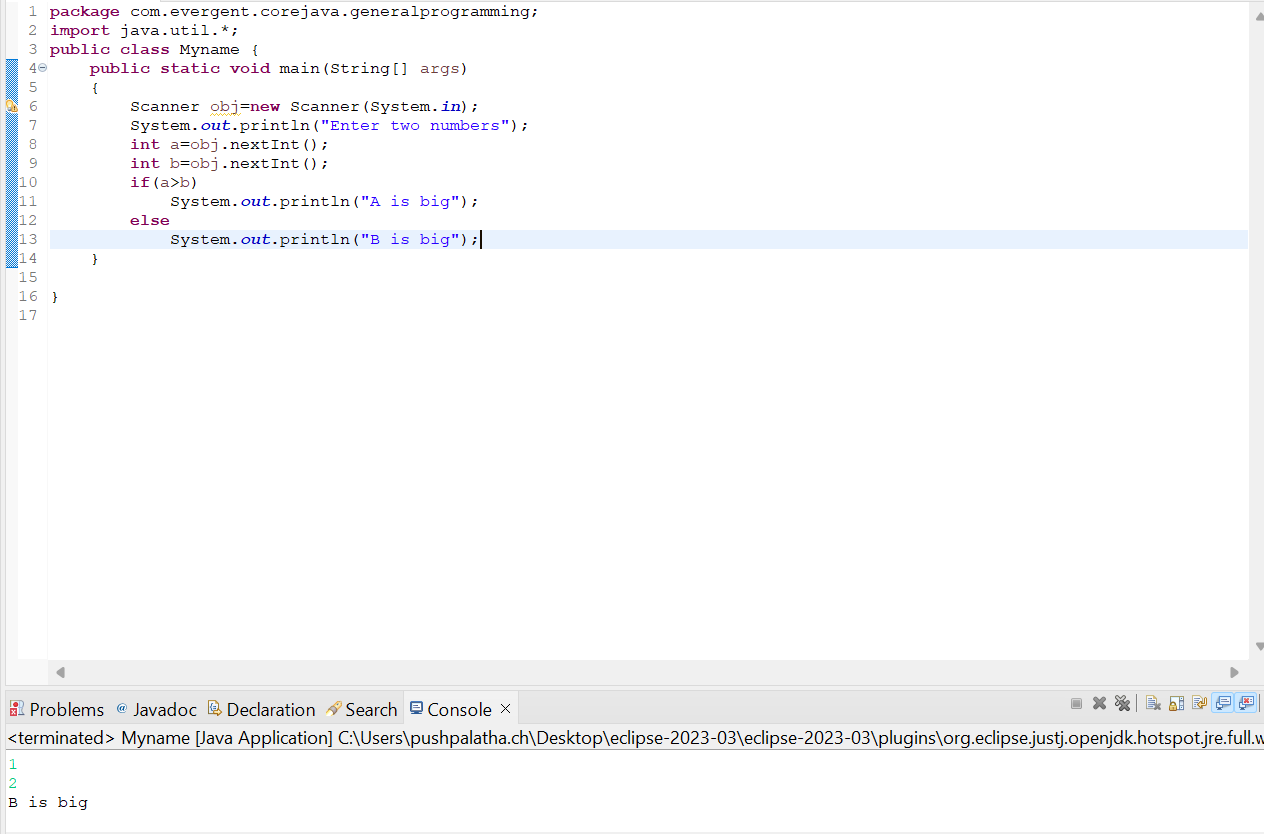
h.supports networking

i.memory management

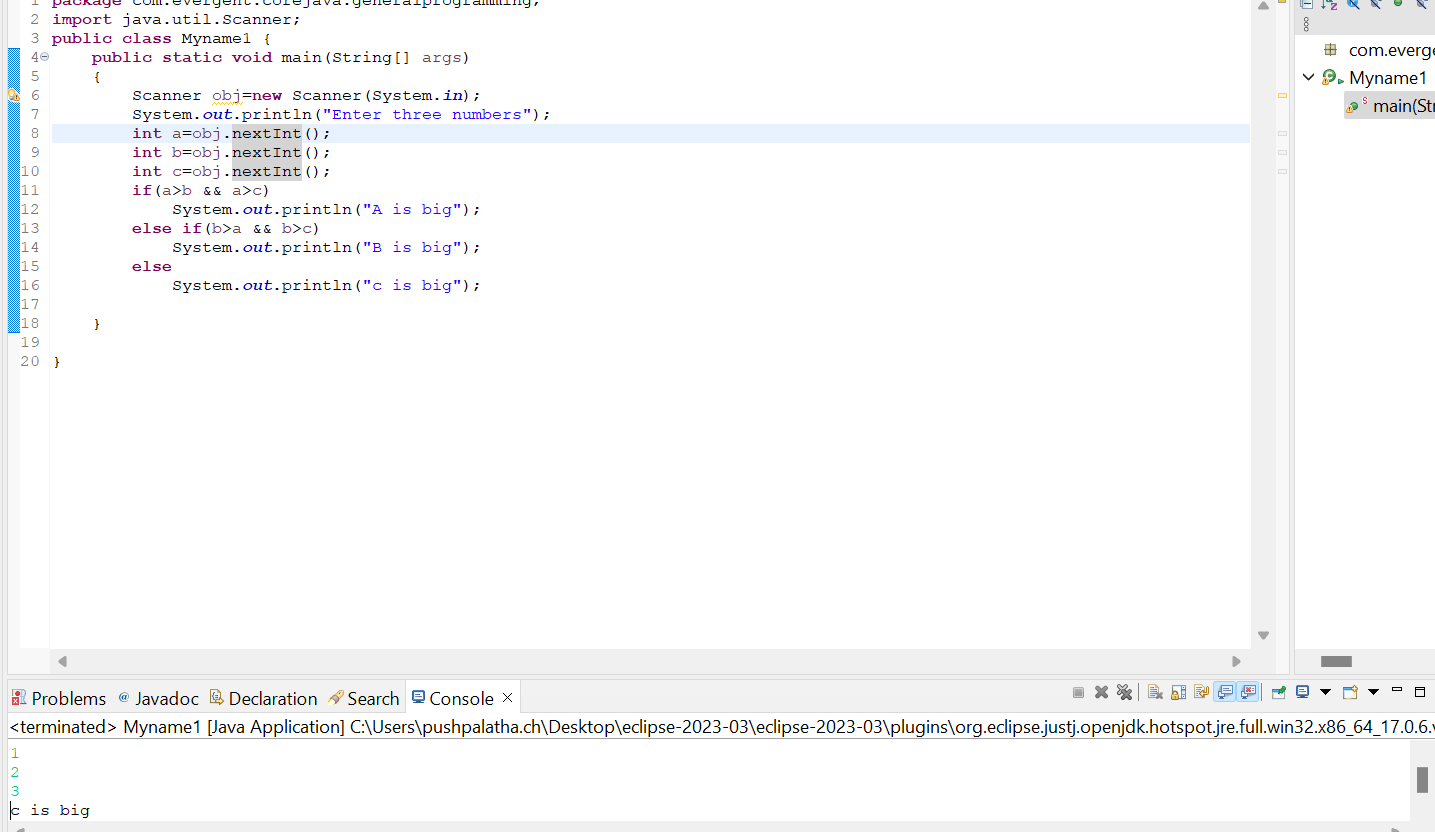
3. JDK,JRE,JVM

4.Basic programming

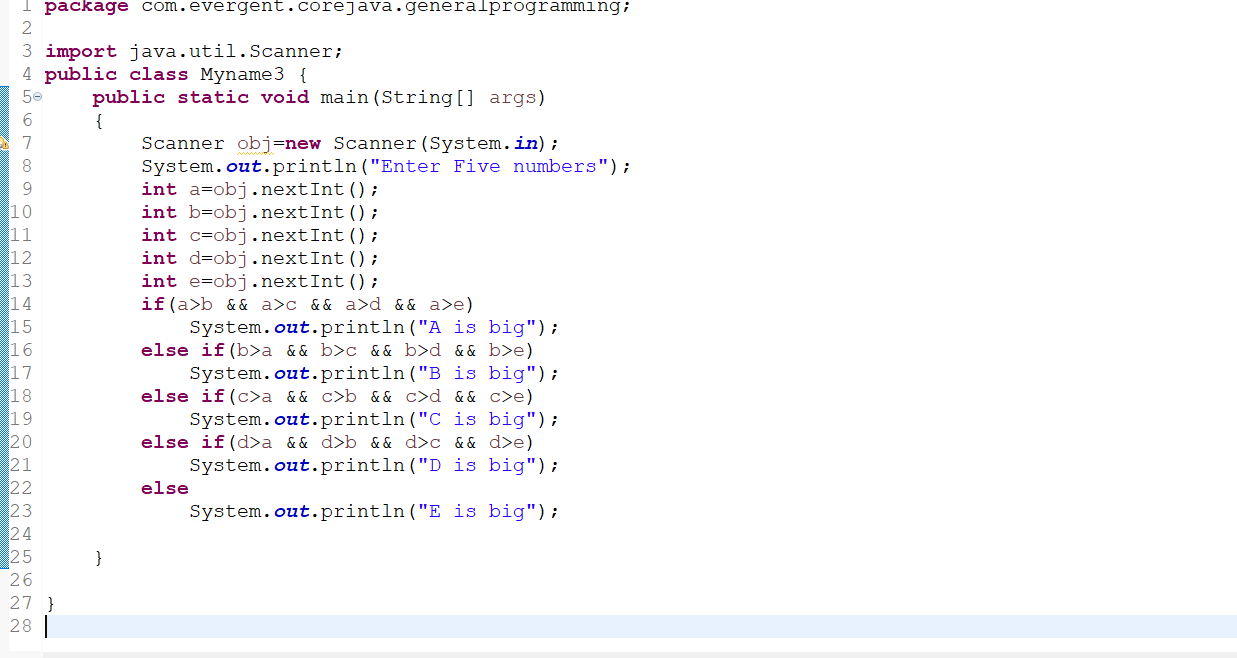
Biggest of 2 numbers

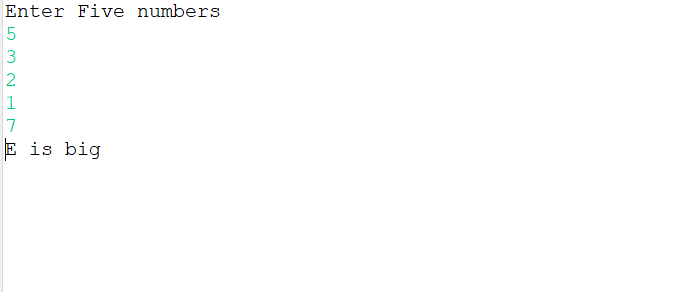


Biggest of 3 numbers

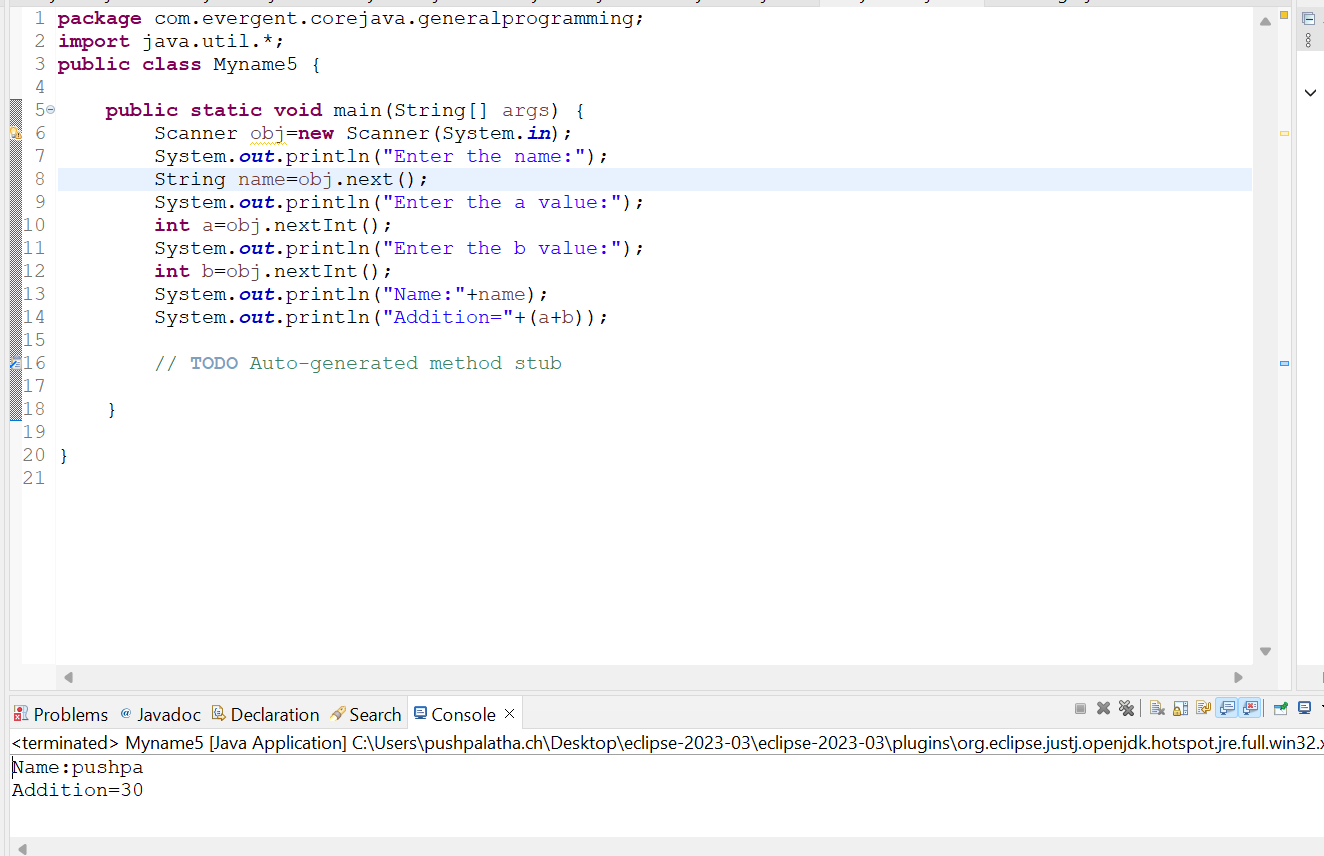


Biggest of 5 numbers





Print Addition and name



Swapping of 2 numbers without and with temp



5.Packages.

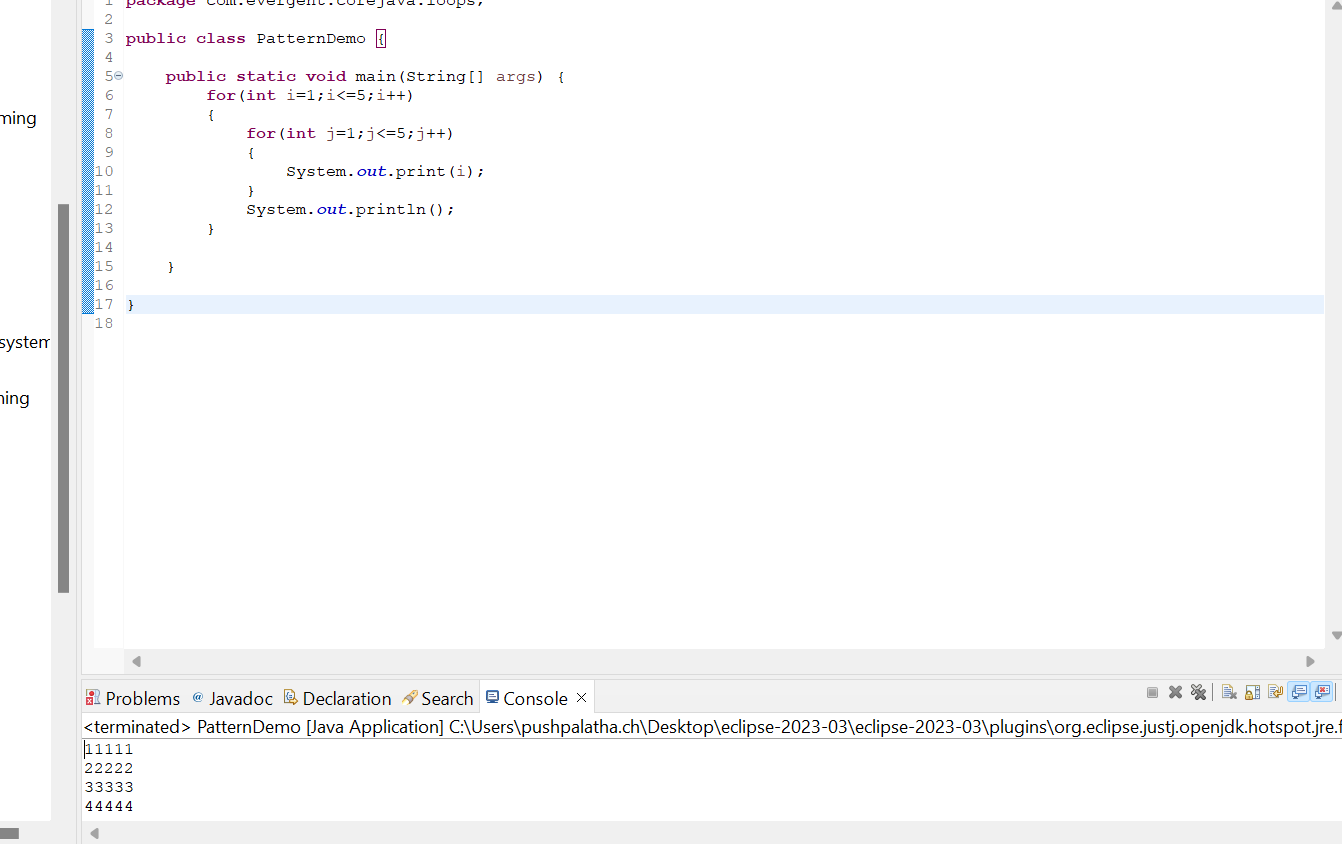
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Date:06/08/2024**

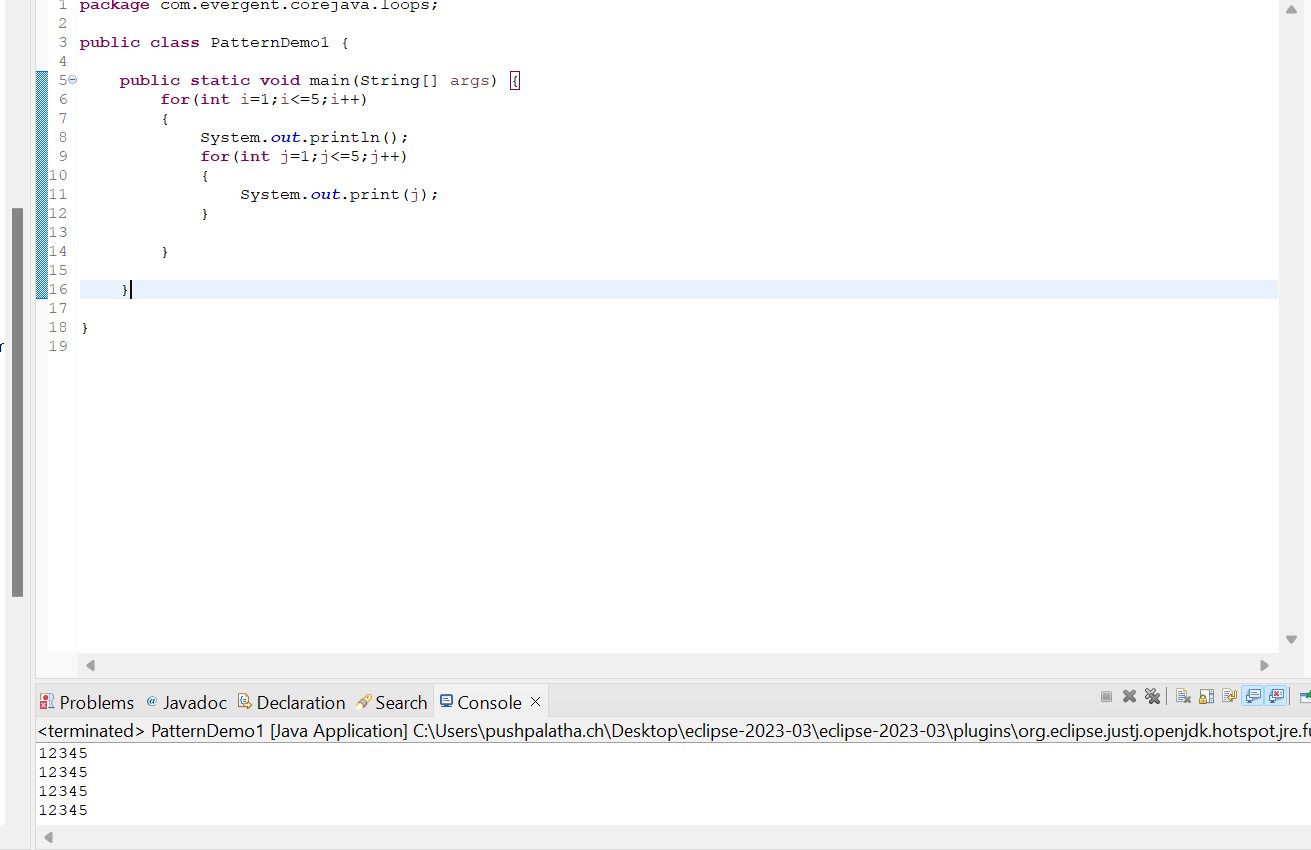
**Day-2**

1. Nested loops

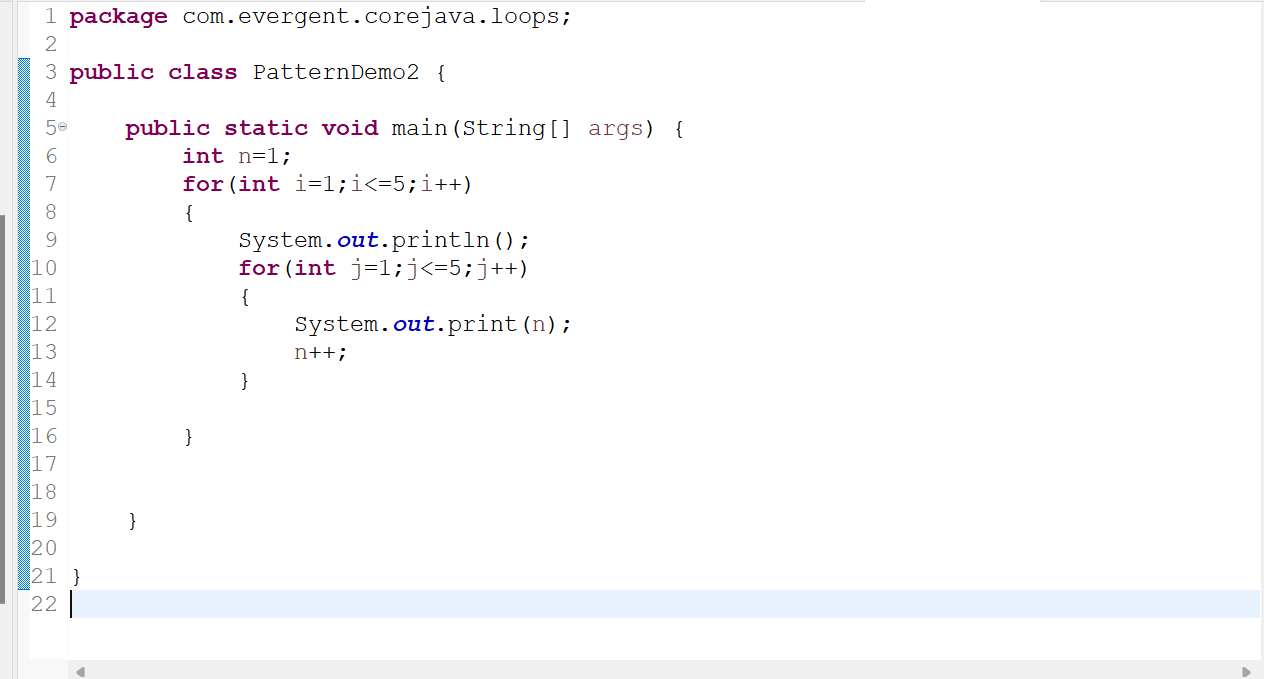
1.Pattern1

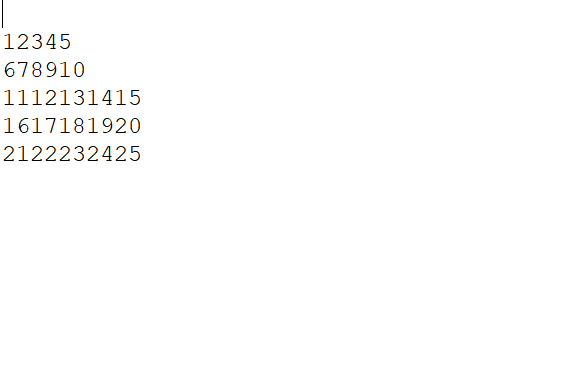


Pattern 2



Pattern 3

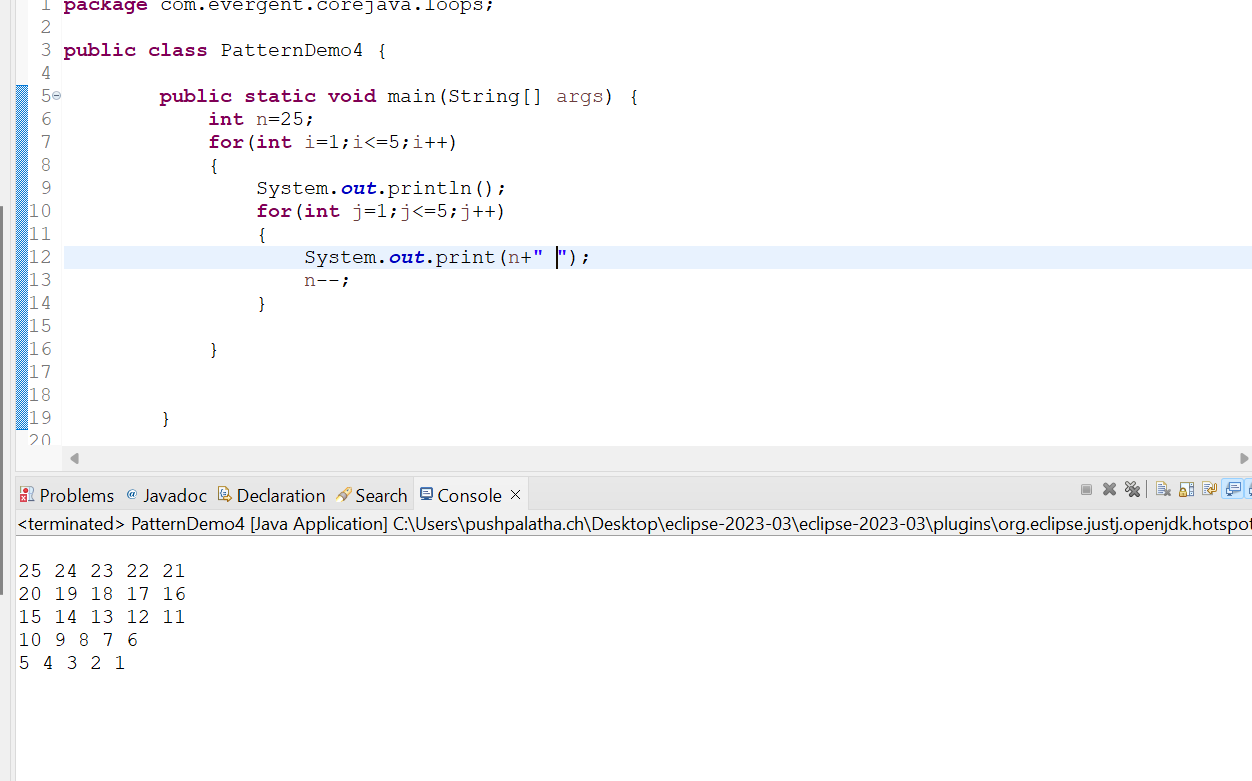




Pattern 4



Pattern 5

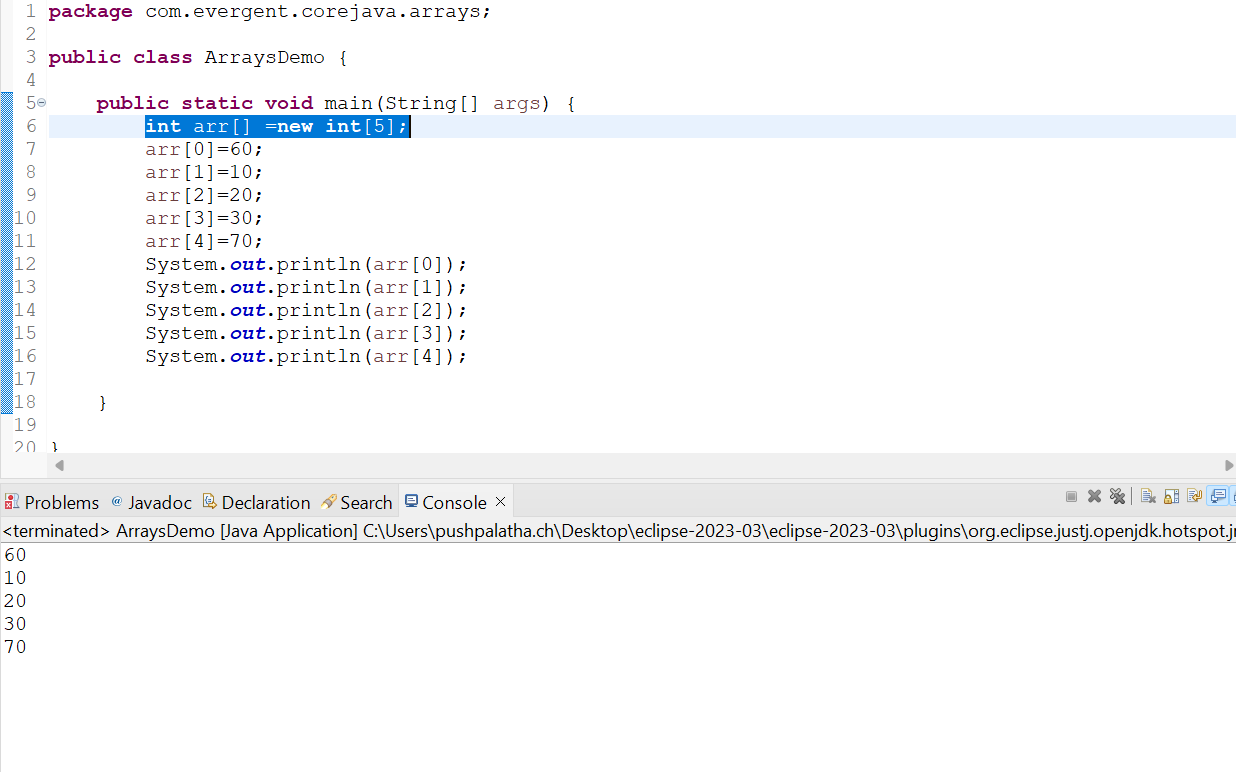


2.Arrays(collection of Homogenous data types stored in contiguous memory allocation with fixed size.)

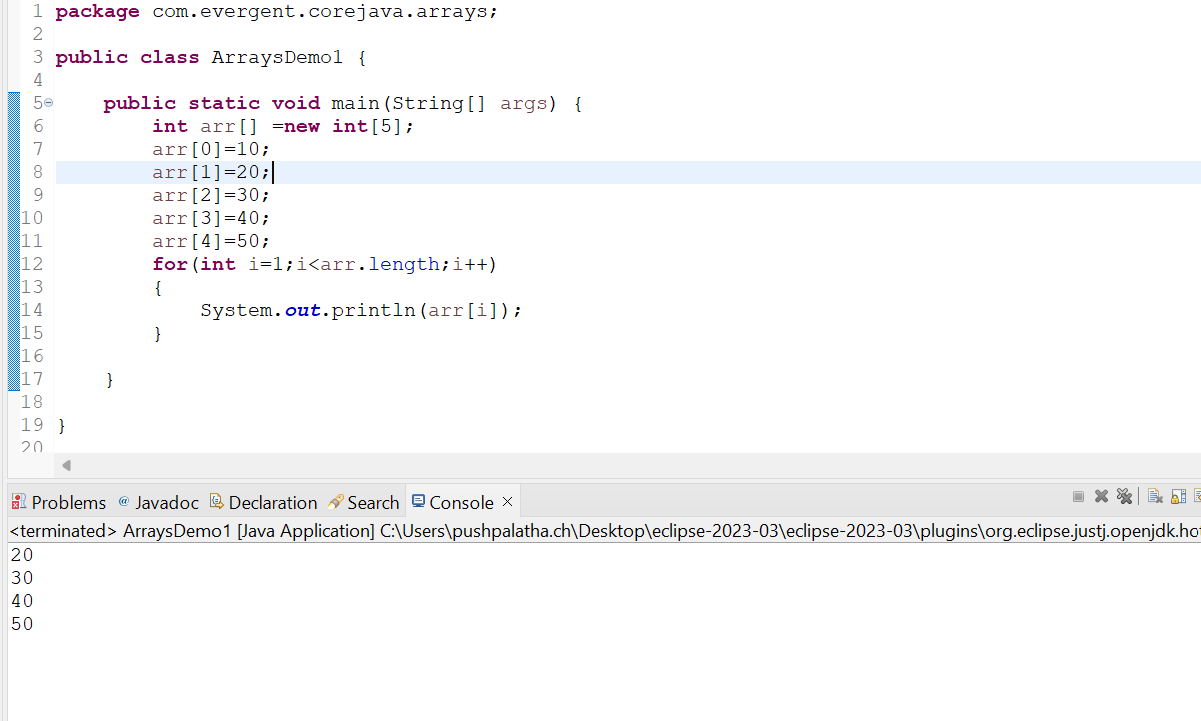
a.single dimensional array

b.double dimensional array

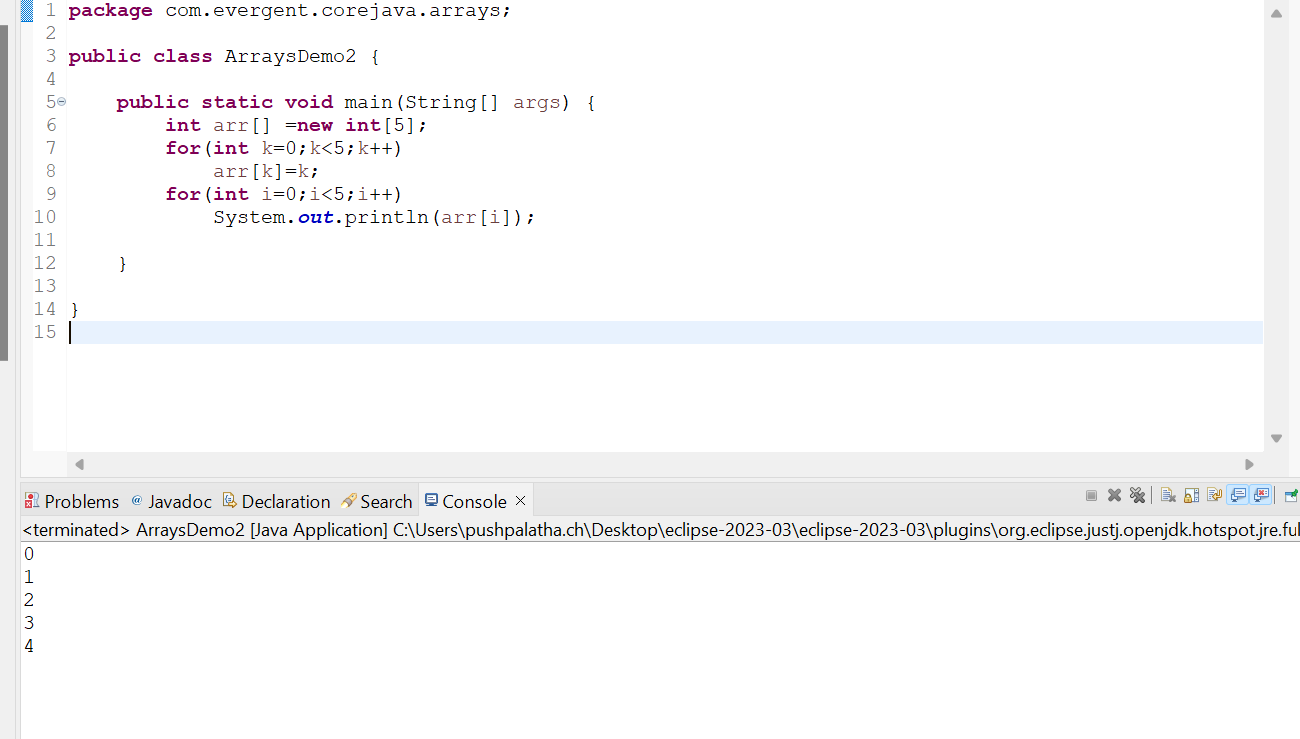
Array1(initializing,retriving of data in single dimensional array)



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



Array3



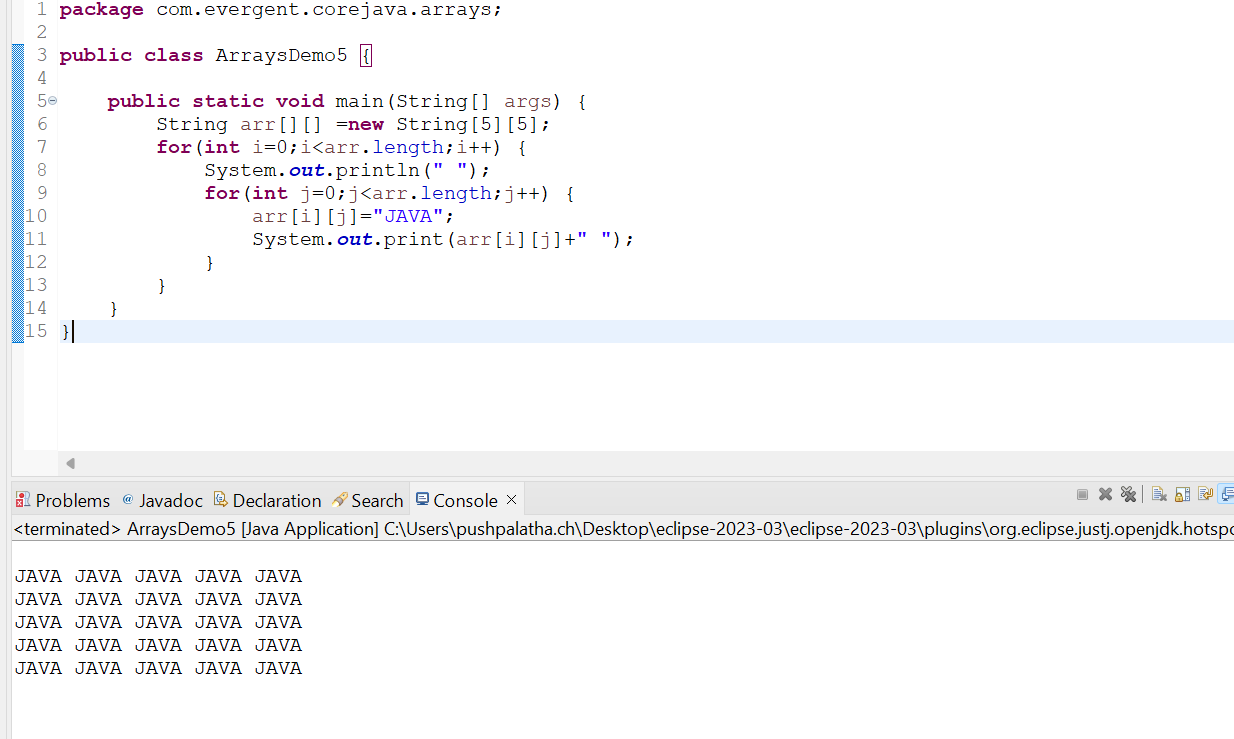
Array4



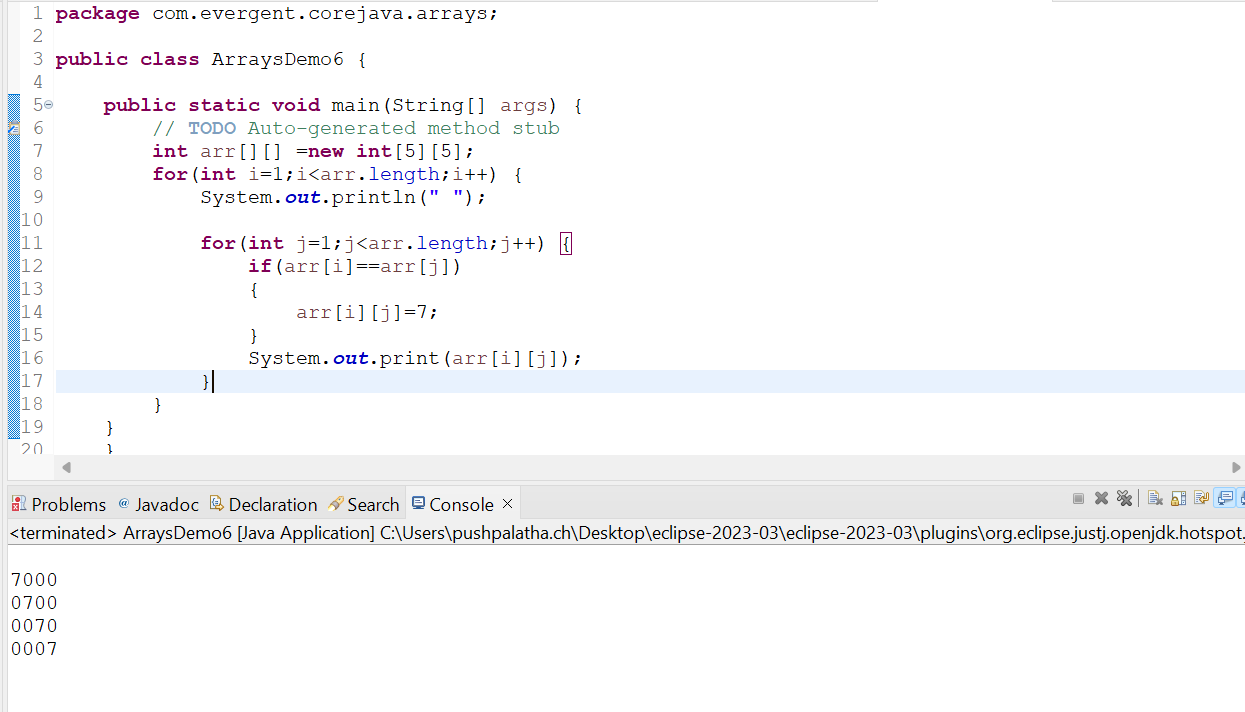
Array5



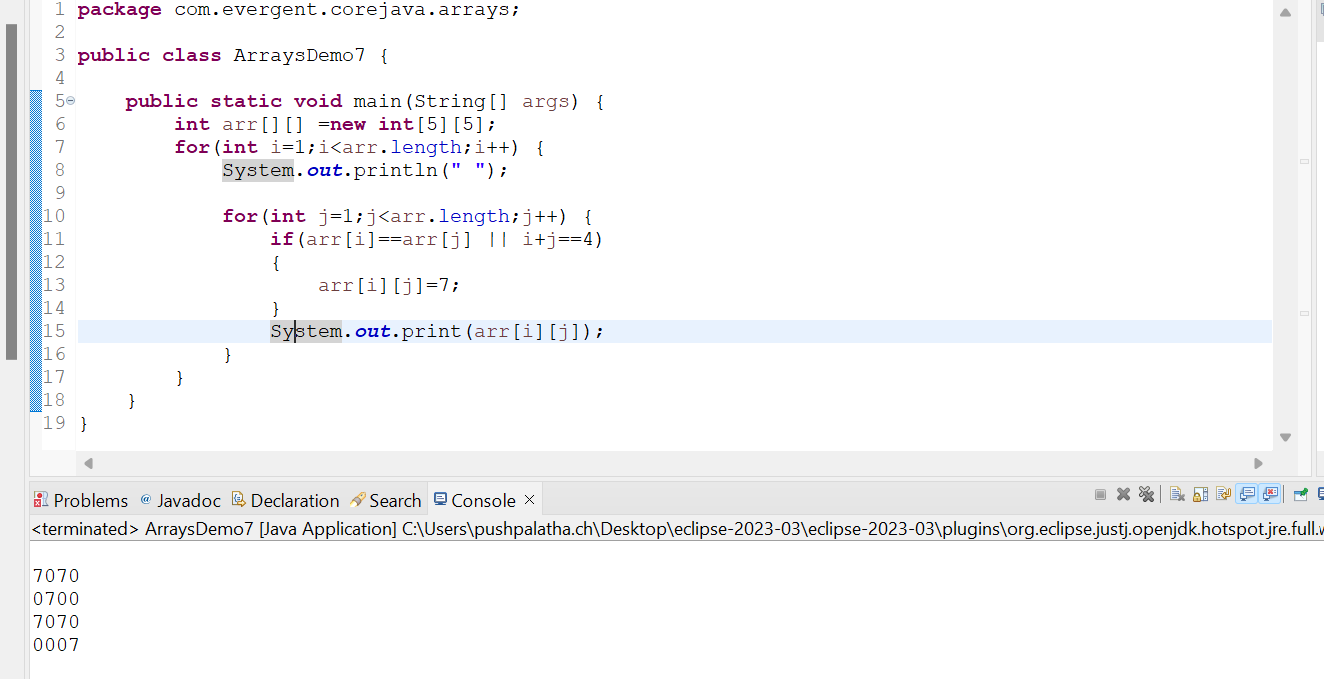
Array 6(Multidimensional Array)



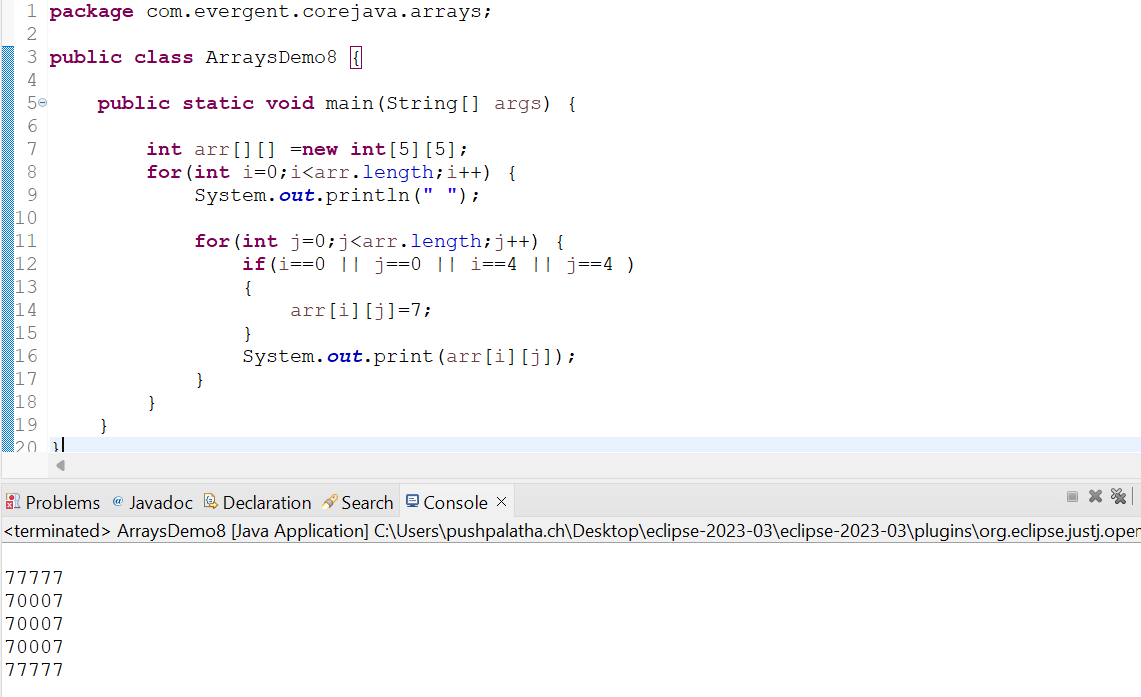
Array 7(Multi dimensional array)



Array 8(Multi dimensional Array)



Array 9(Multi dimensional Array)

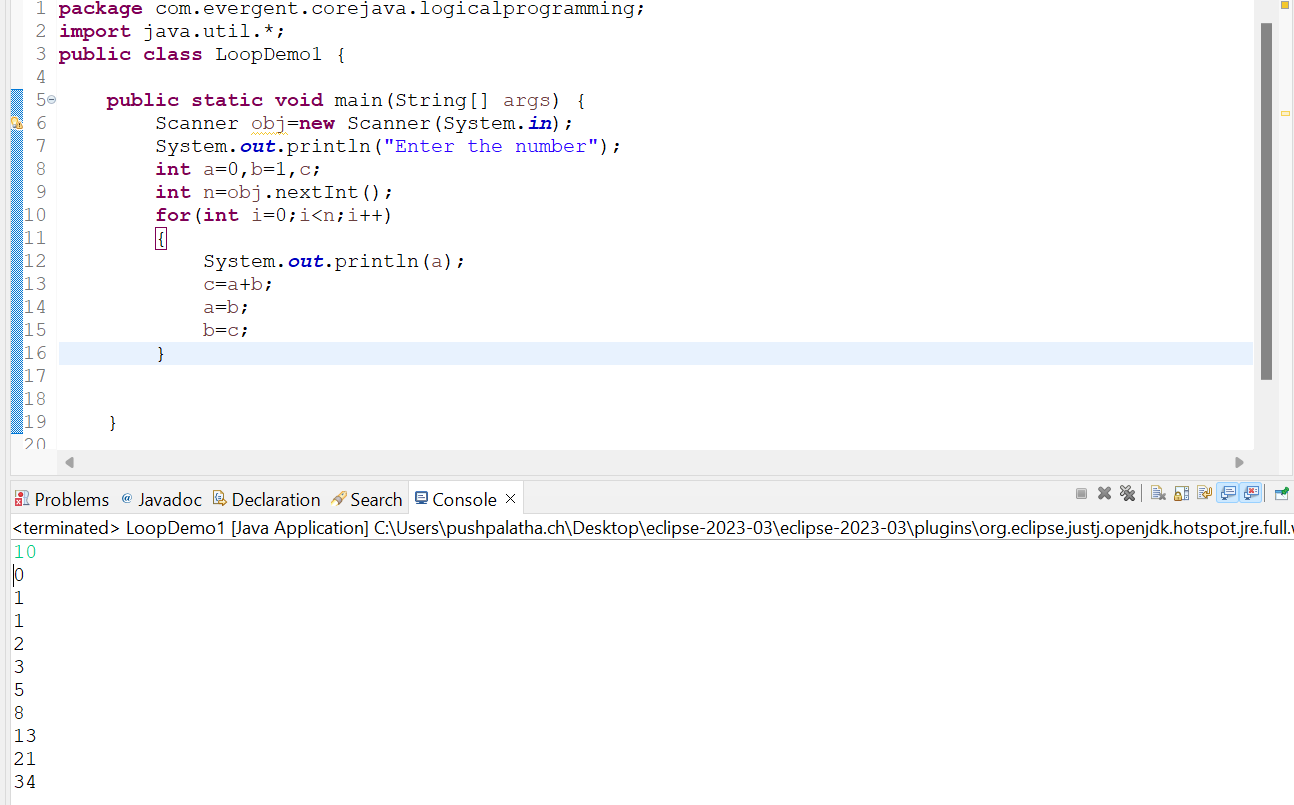


1. Logical programming

Factorial program



Fibonacci series program

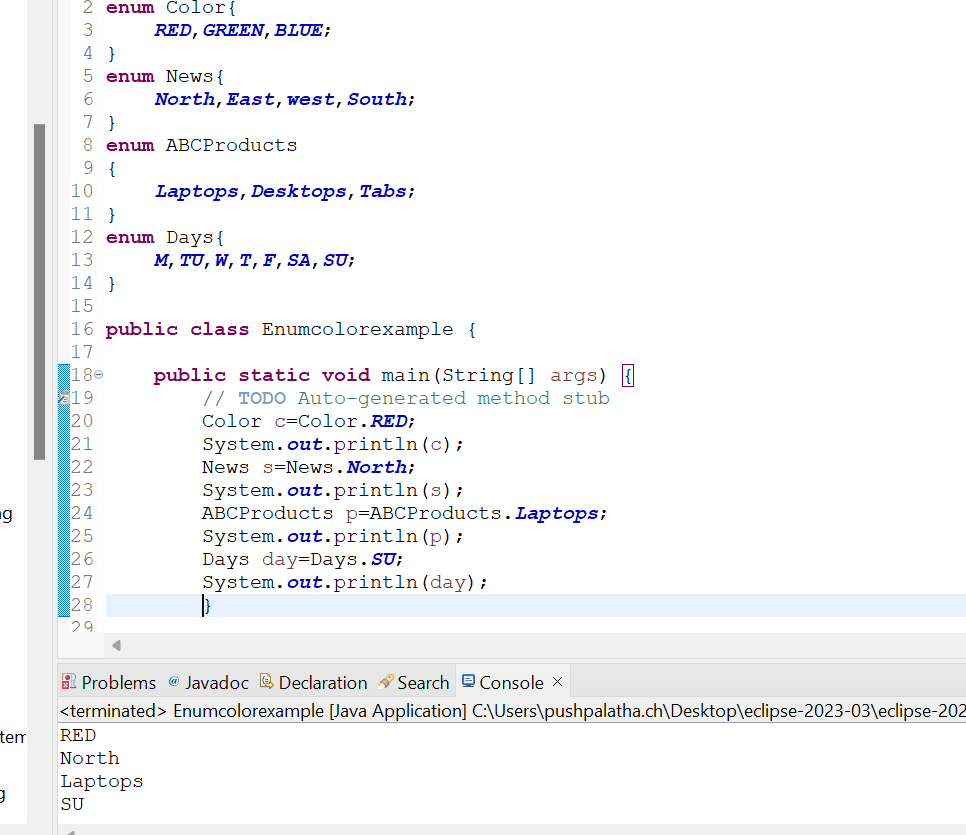


1. Switch case

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.

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

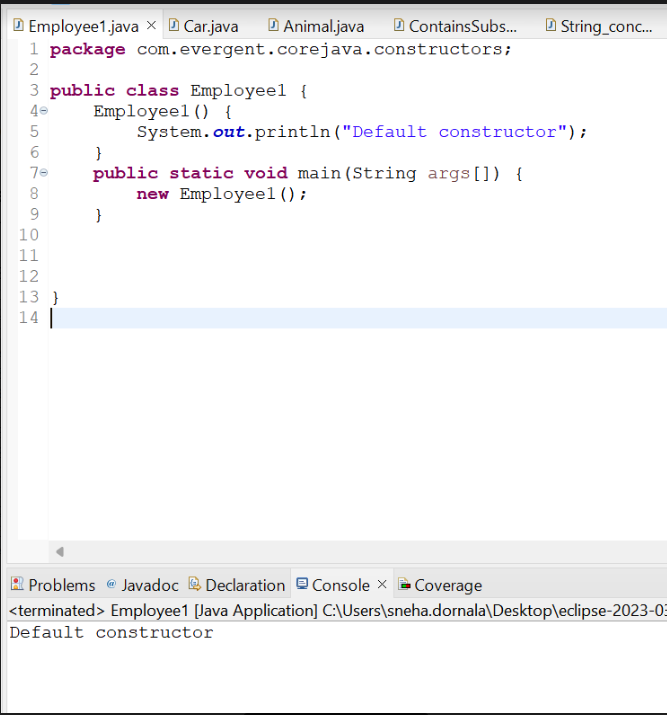
**Date 08/08/2024**

**Day-4**

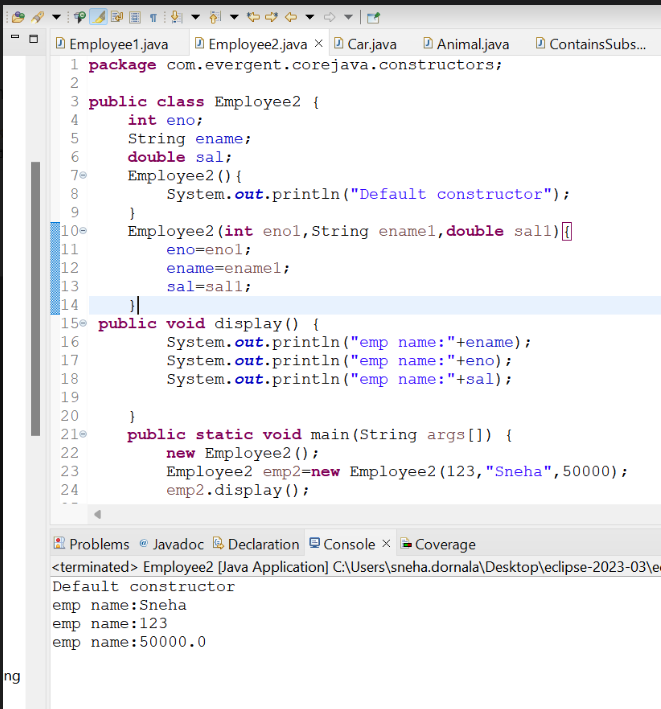
**Constructors**

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

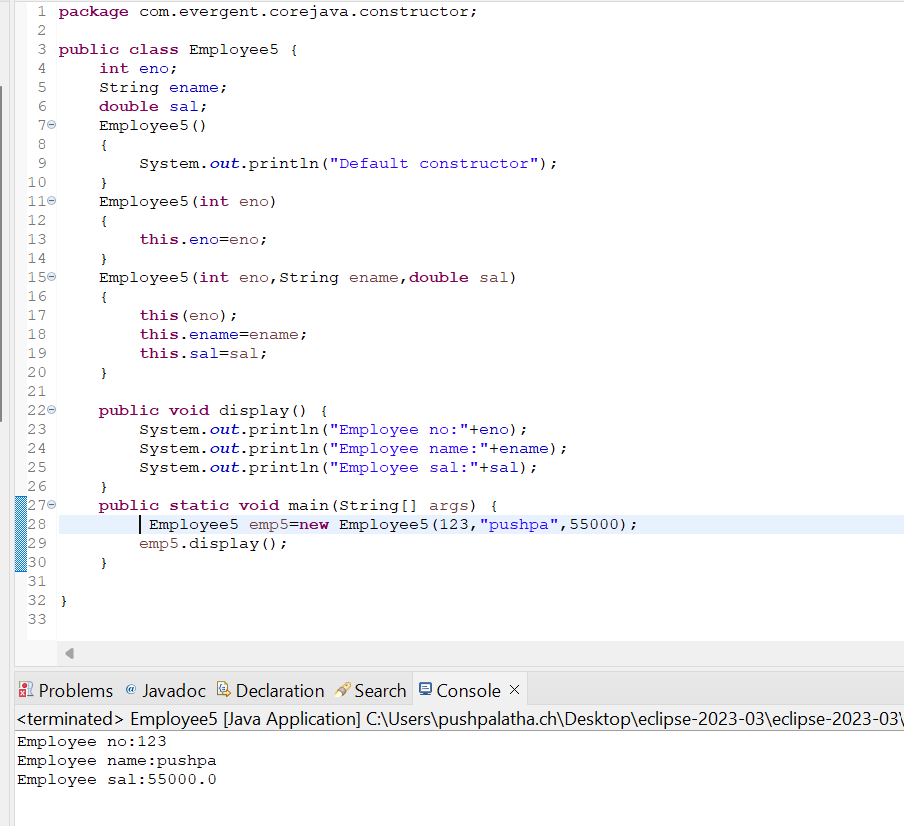
Default constructor



Parametrized constructor

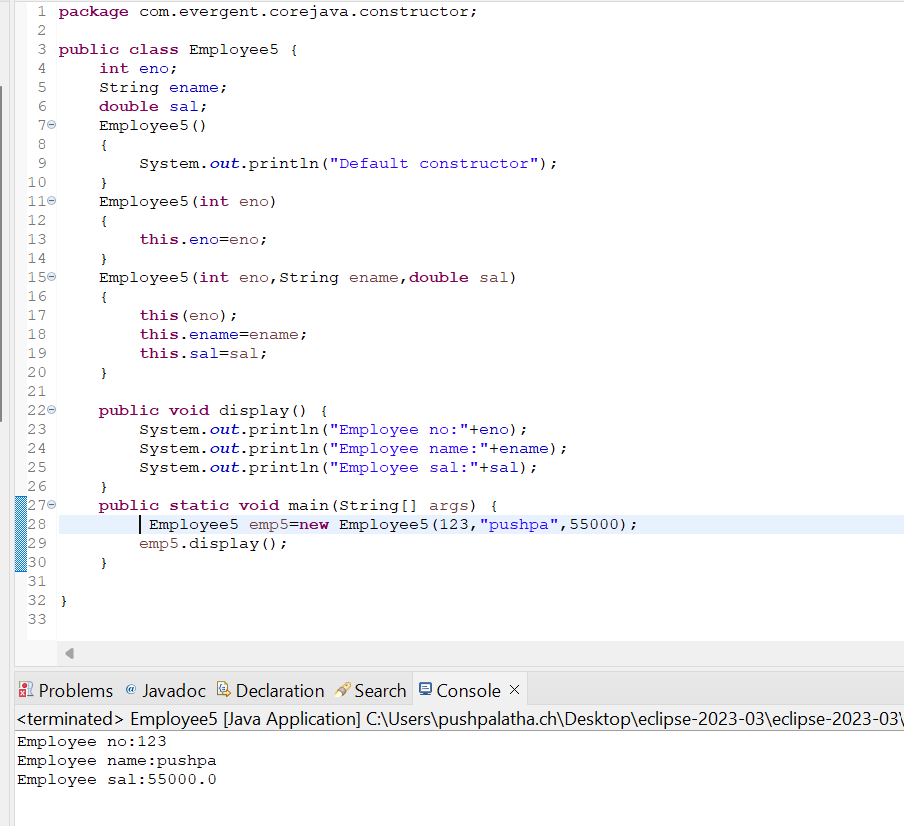


1. We can access constructors while creation of objects.
2. Constructors are mainly for intializing the objects.
3. Constructors does not any return type not even void also .If we declare as a void it will consider as a method not as constructor.

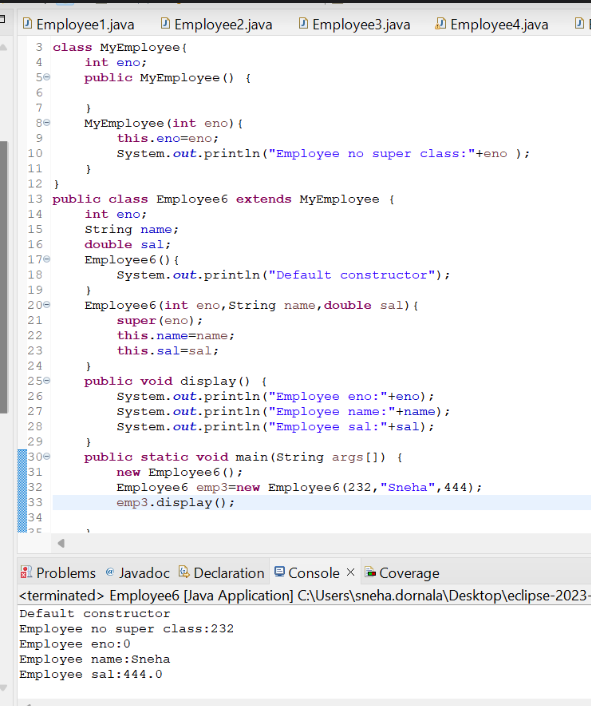


1. Every Class Contains atleast one default constructor Either we will create or JVM will generate at the time of compilation.
2. this and super Keywords.

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)

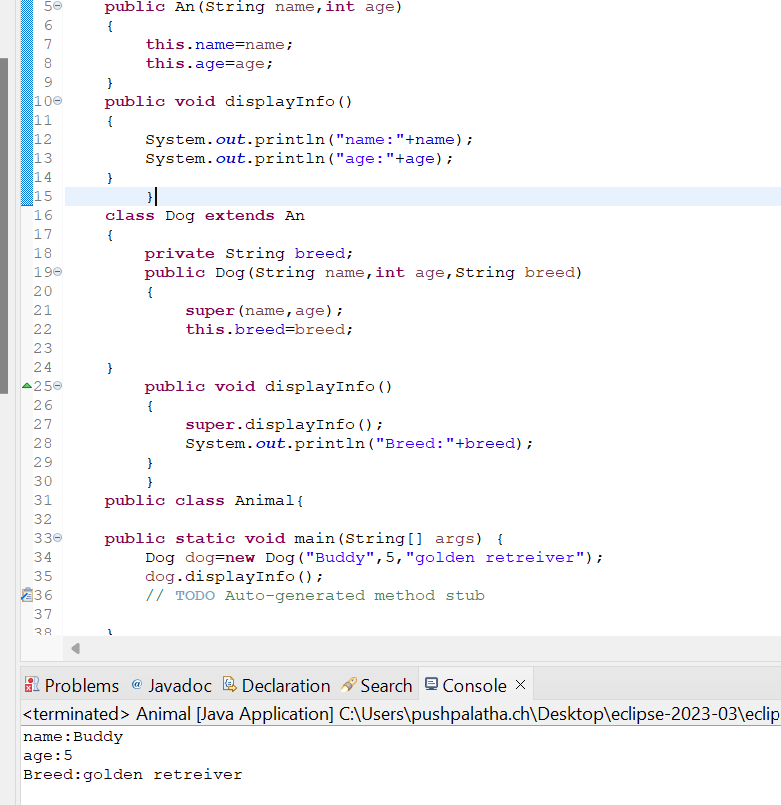


1. Object copy constructor.



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.

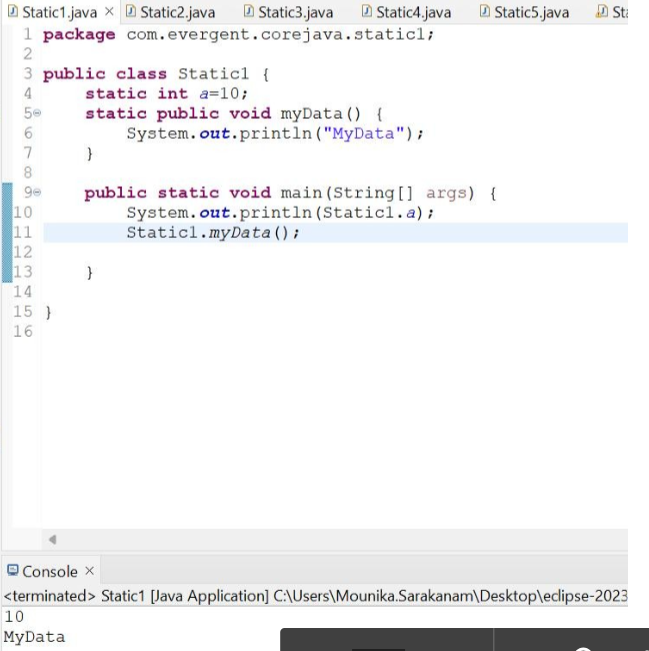


ConstructorOverloaded

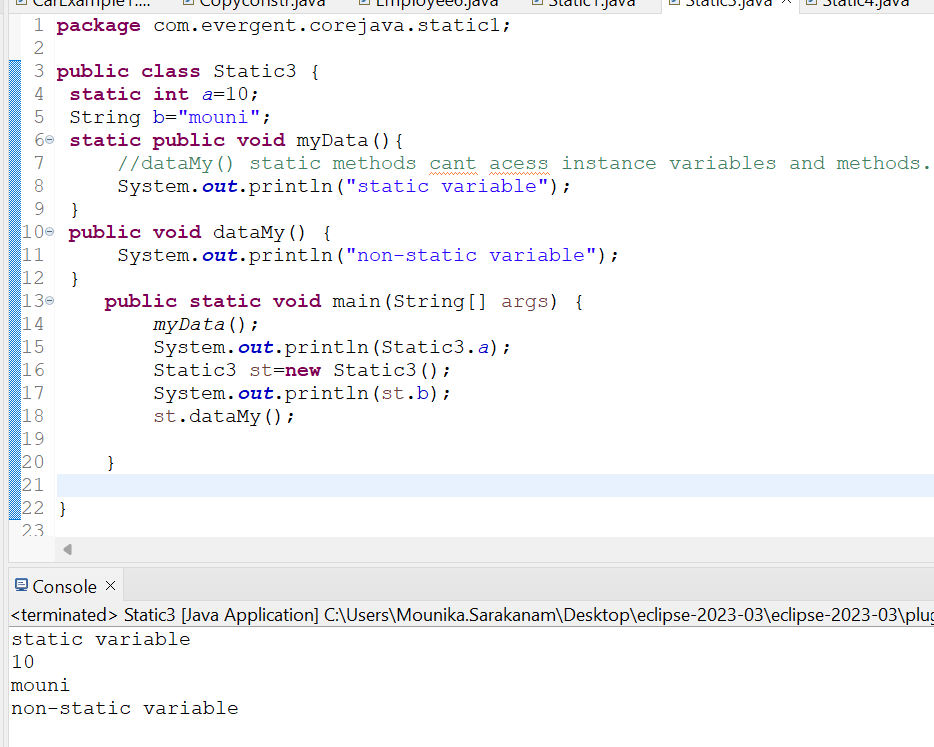
**Static keyword**

Static is a variable

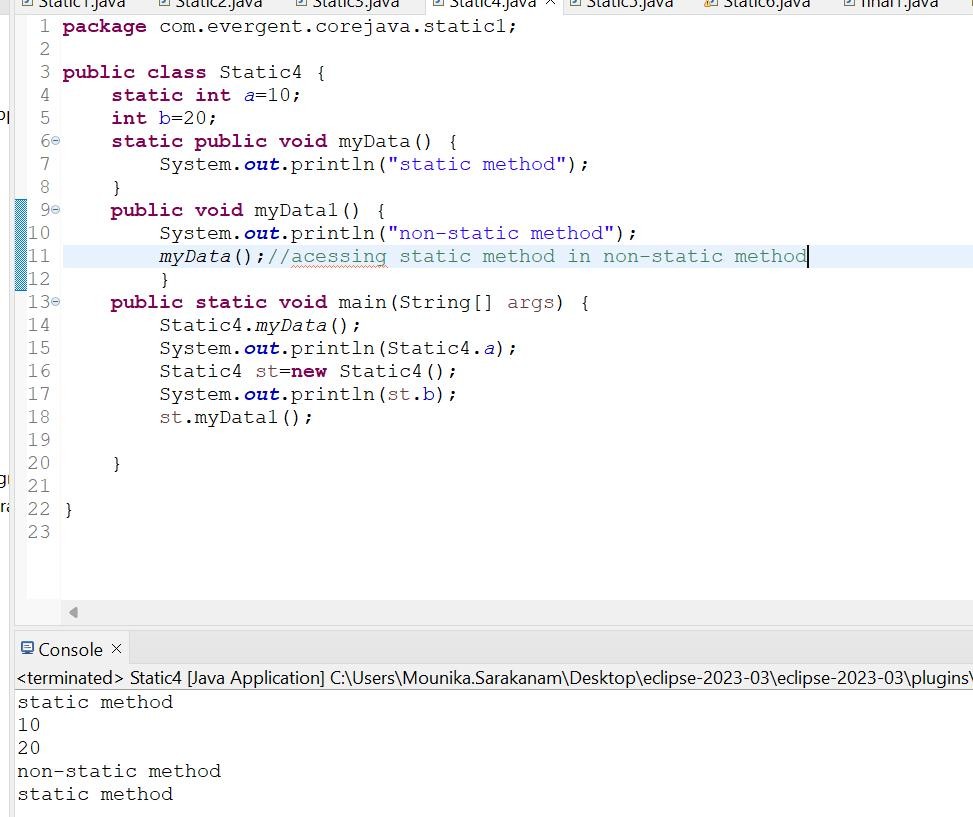
We can declare static as variables,methods



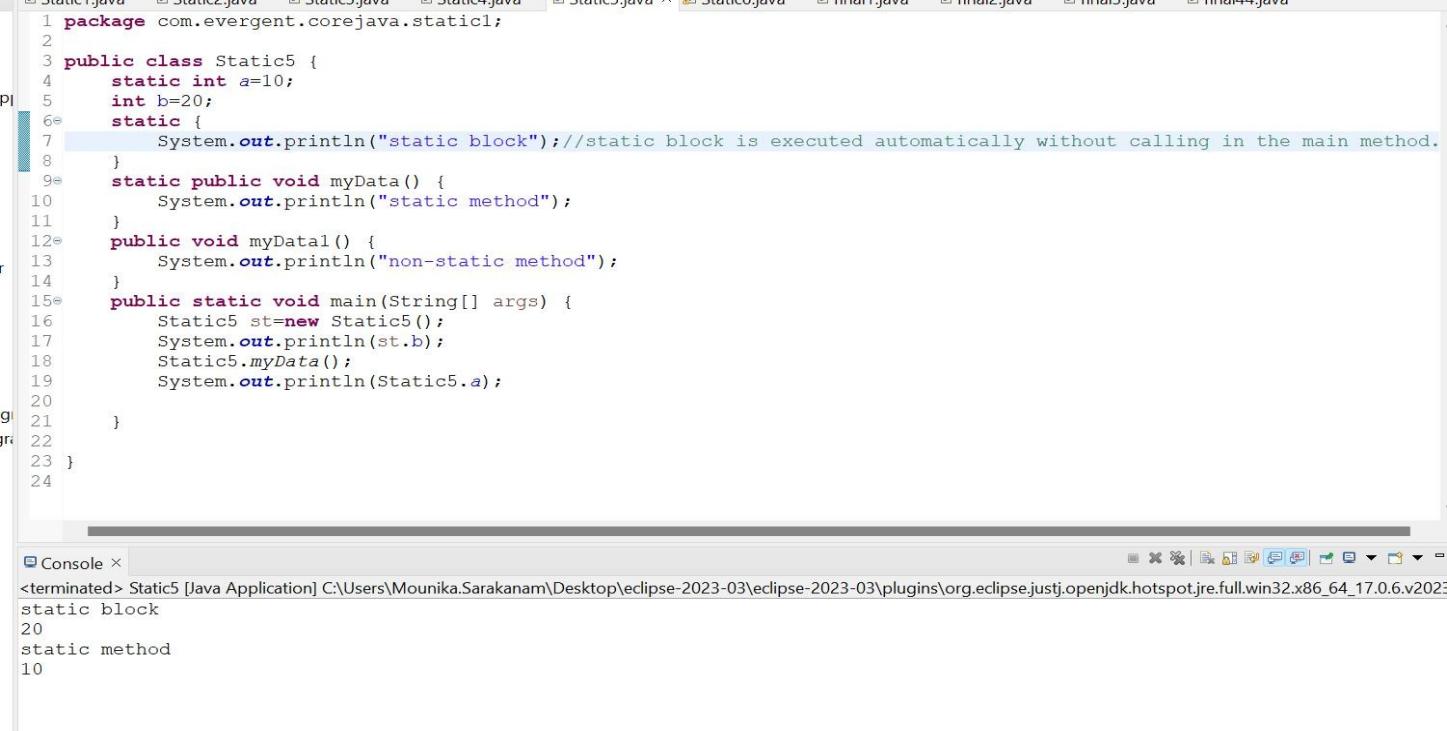
Inside static method we cannot access non static variables or methods.



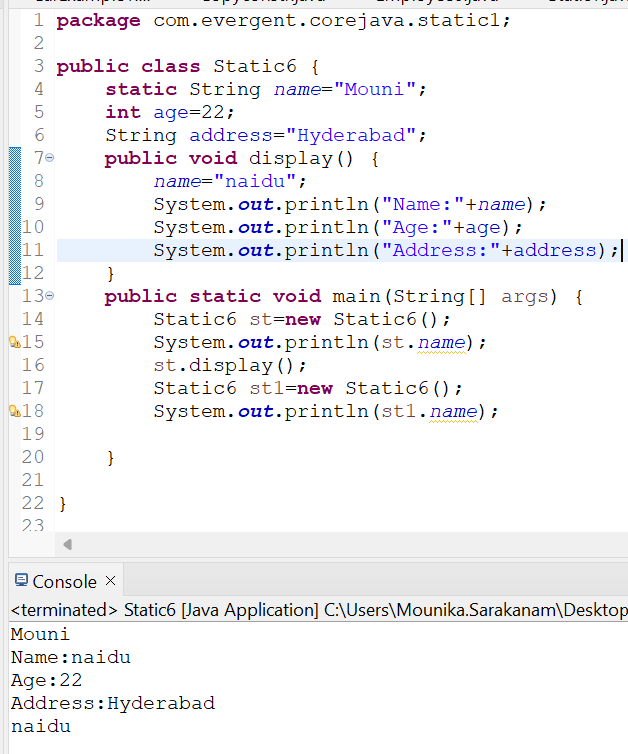
Inside non static method we can use static methods or variables.



Static block 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.



Example for static

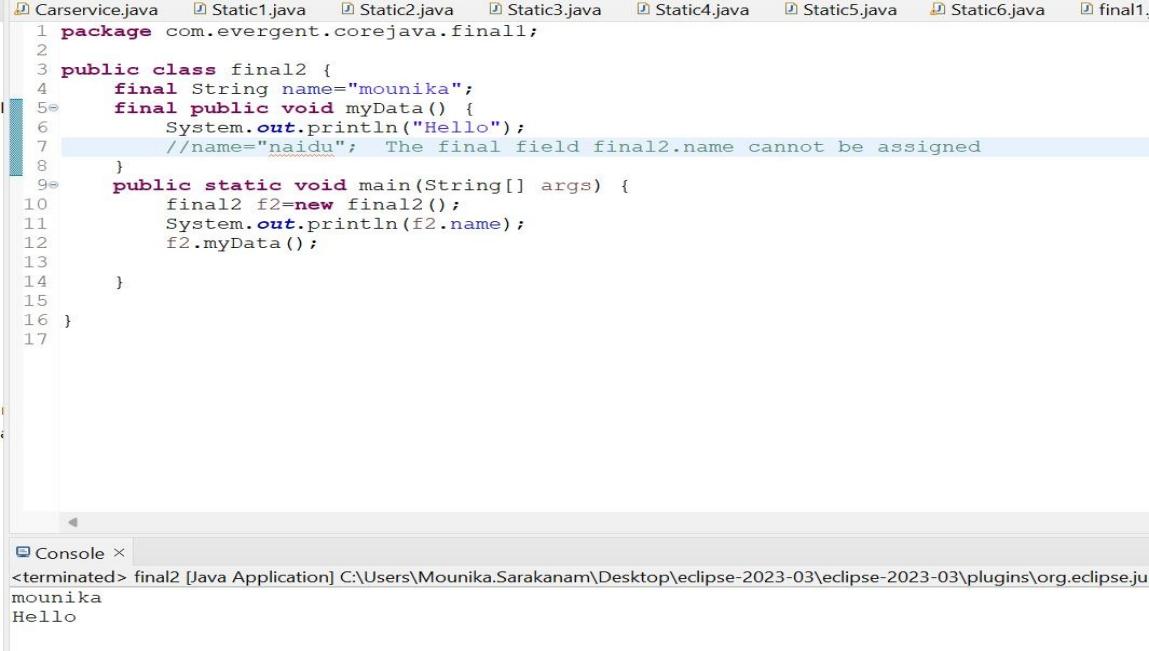


# final

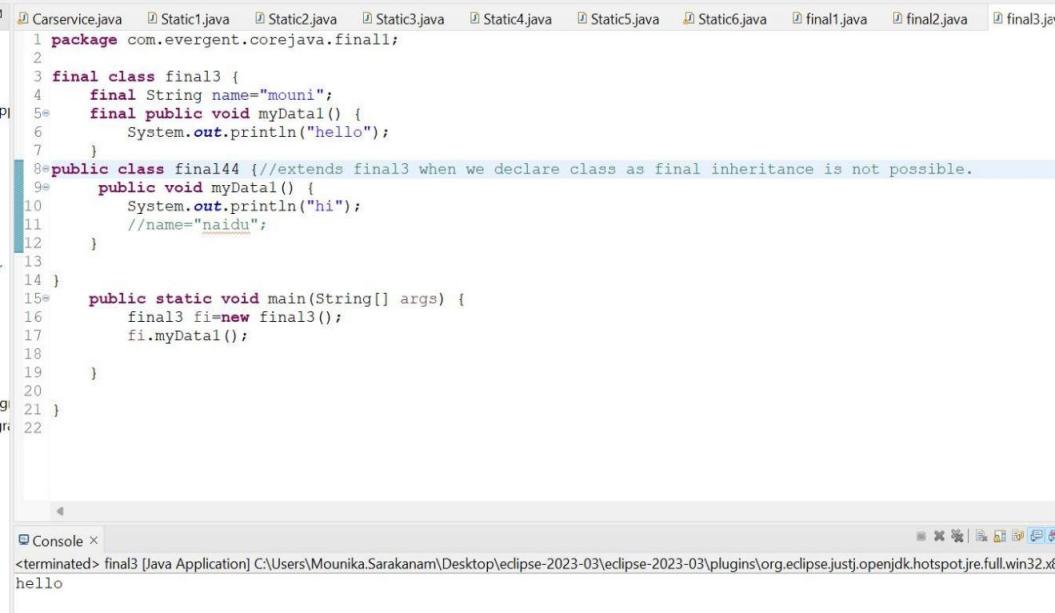
final is a keyword

final can be declared as variable,method,class

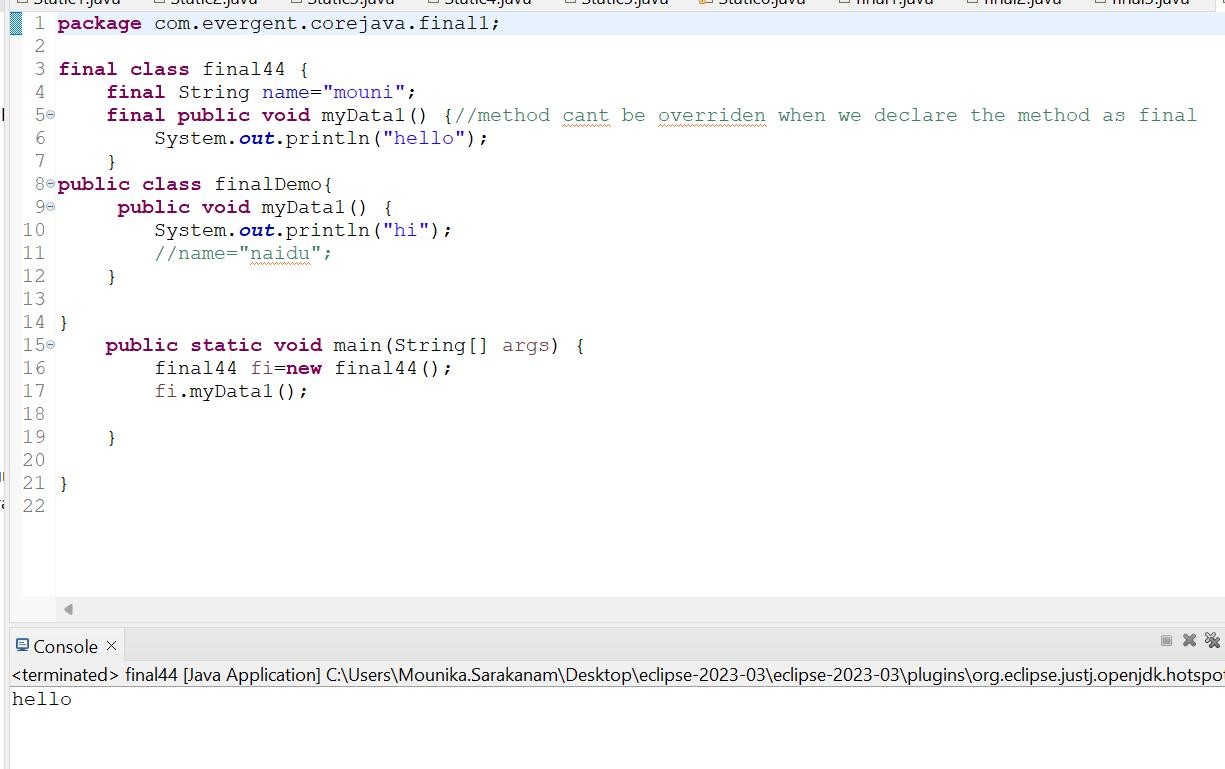
If variables are declared as final those are treated as constants cannot be modified after.



If the class is declared as final we cannot inherit



If the method is declared as final we cannot overriden.



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.

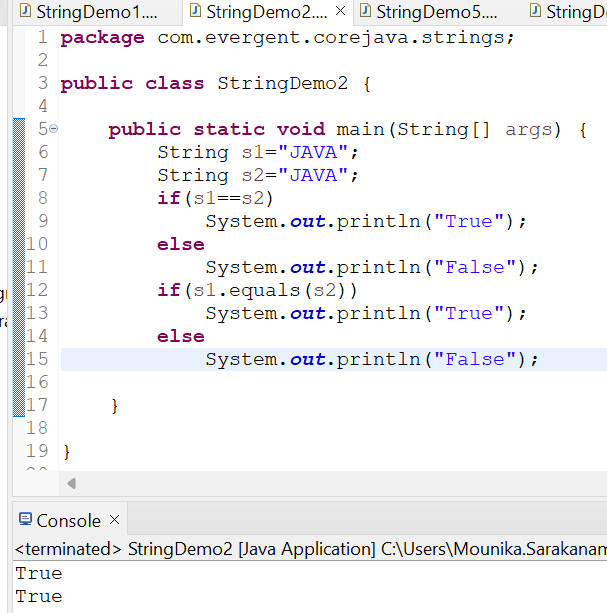
== method in String is used to check the Memory location.



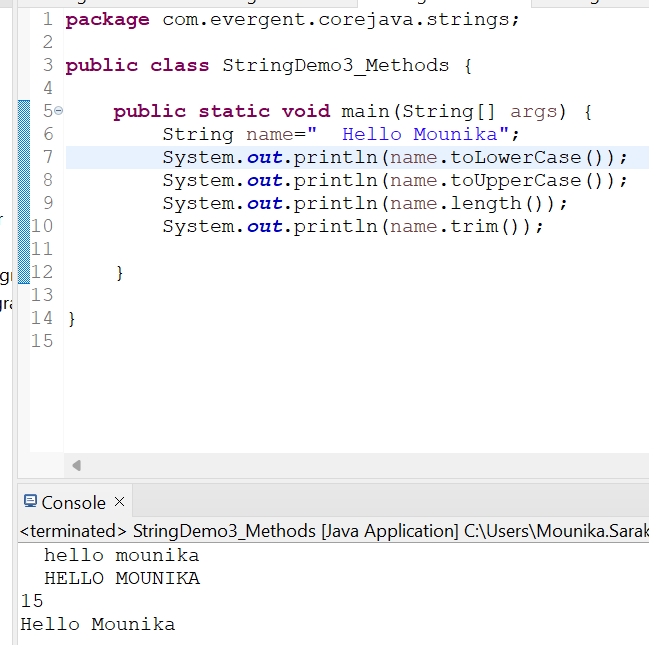
2)Creating String using literals

equals() in String used to check the content.

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



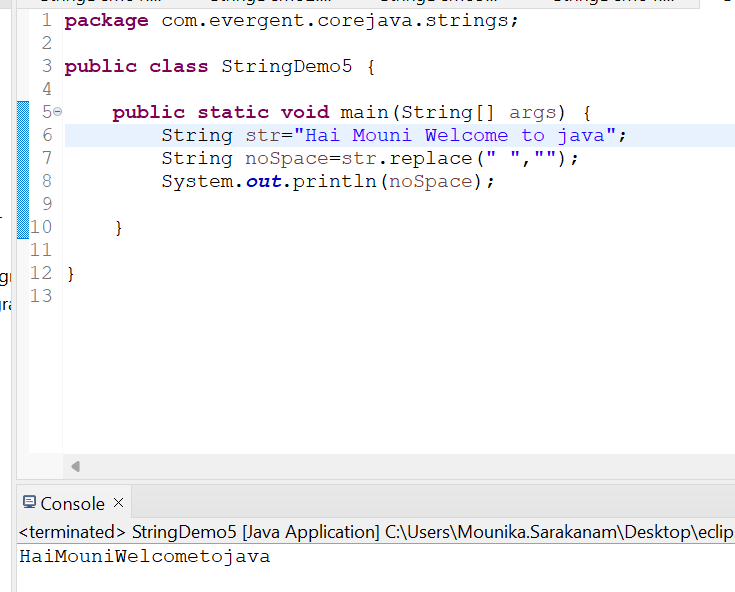
1. String methods (toLowerCase()-Changes the complete String into lowercase,
2. toUpperCase()-Converts the complete String into uppercase,
3. length()-To know the length of the String,
4. trim()-To remove the spaces before the String)



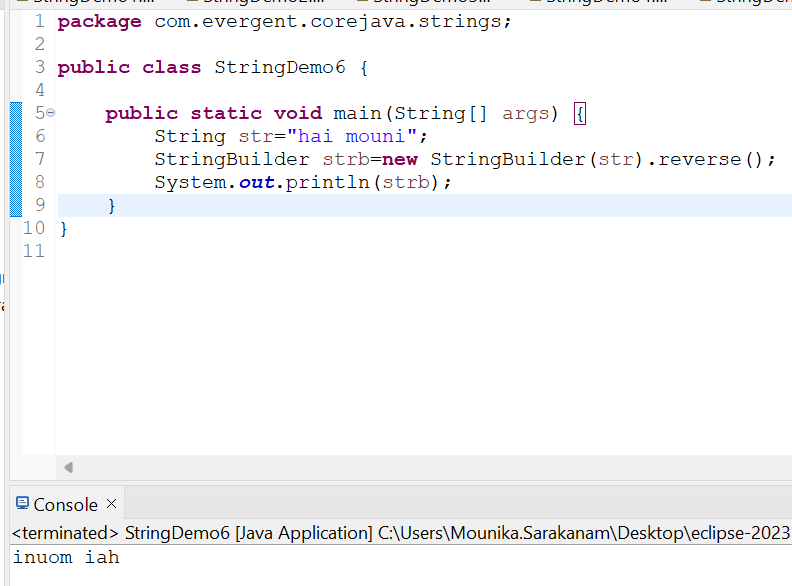
4)String method(contains()-to check whether the substring is present in the string)



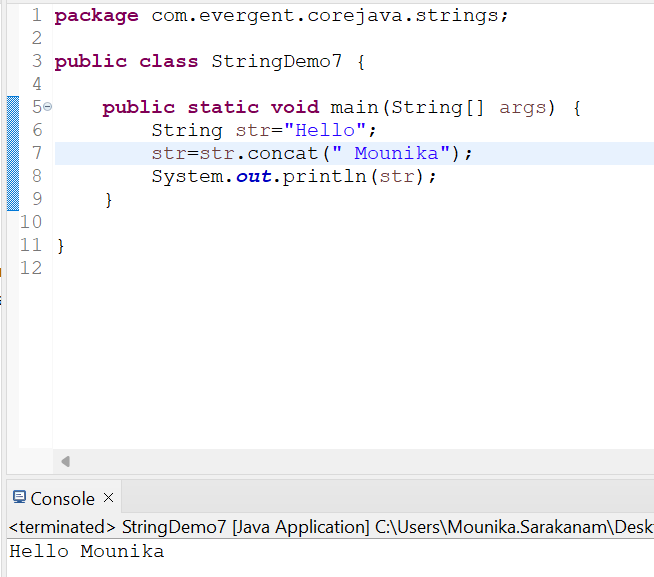
5)String method(replace()-to replace the values with other values)



6)String method(reverse()-to reverse of the String)



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

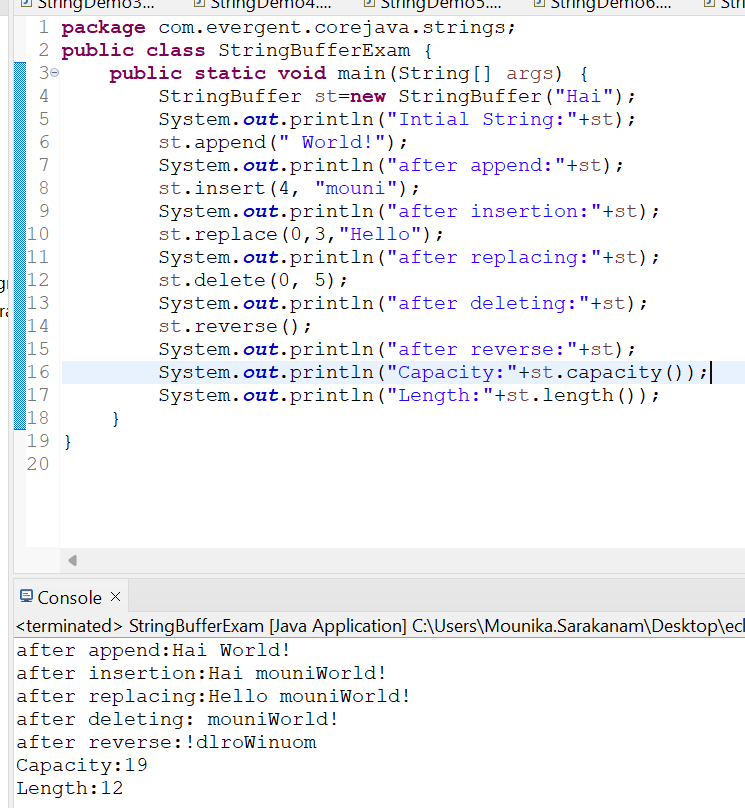


StringBuffer

* 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,

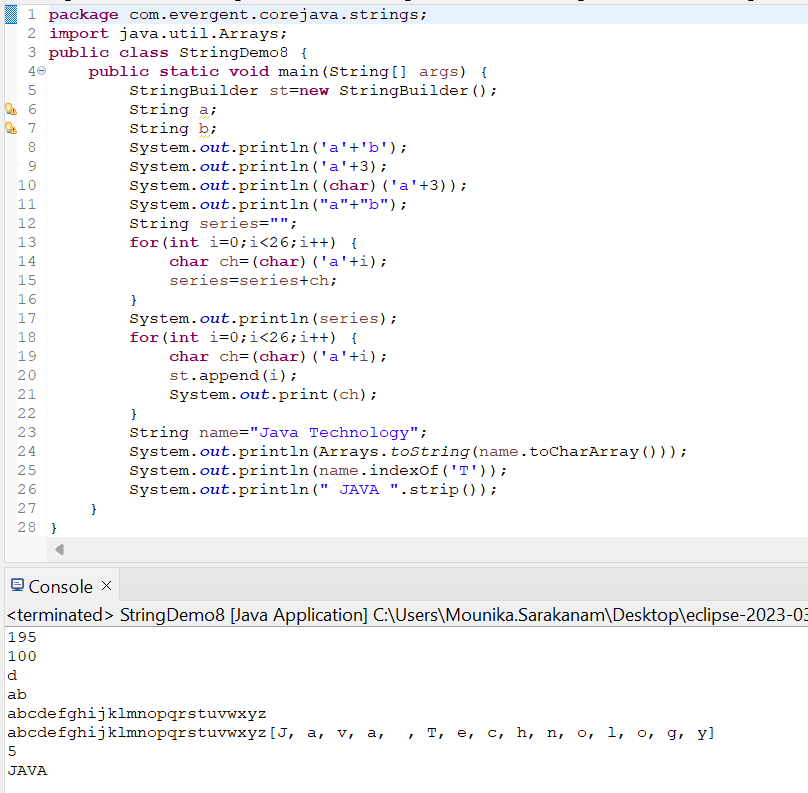
Insert-to insert the string into the present string based on the index value,reverse-to reverse the string ,replace-replace the string with other string based on starting index and ending index,delete-to delete the sub string or string from the existing string based on starting index and ending index.)



StringBuilder

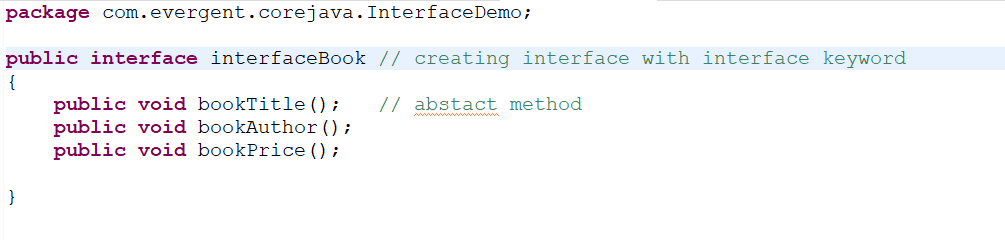
* String Builder is a final class.
* StringBuilder are mutable.
* StringBuilder class consists of methods.
* Methods in String class are non-synchronized.

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.)

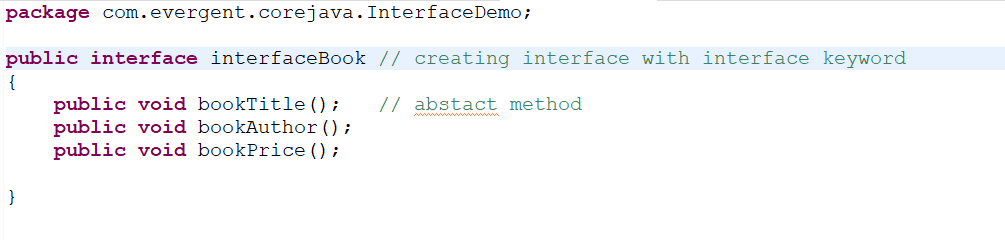


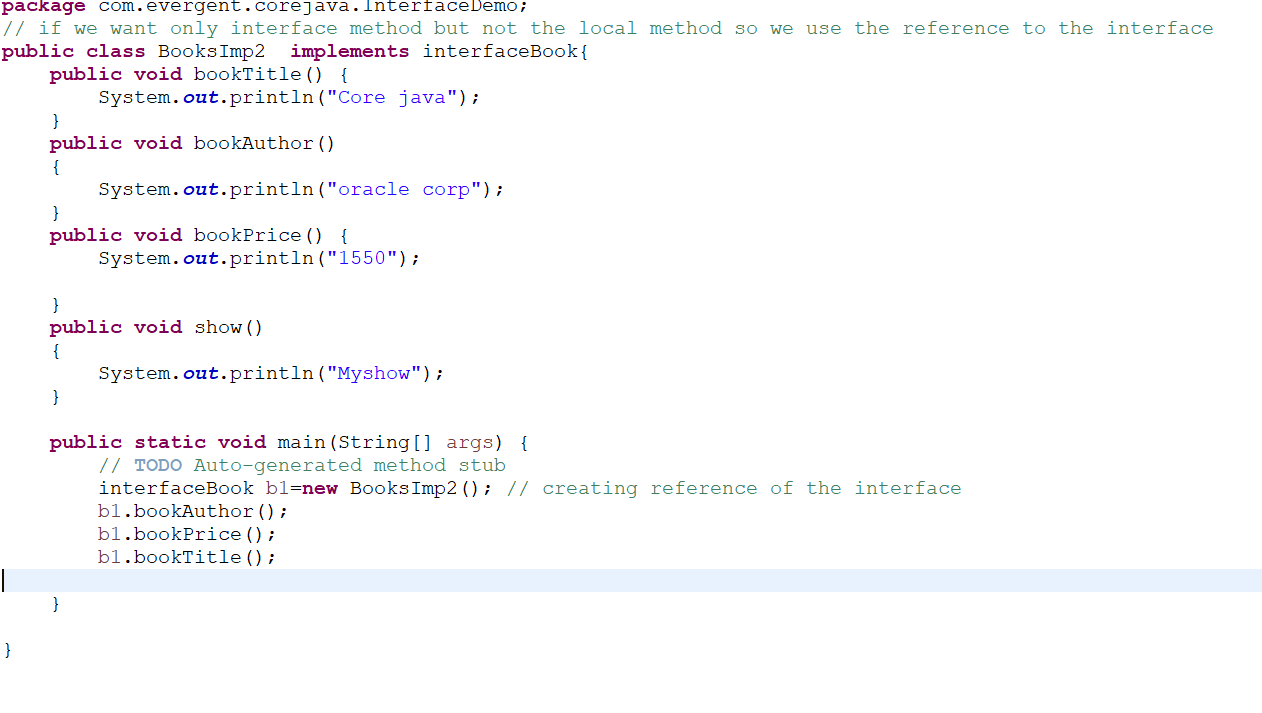
**Interface**

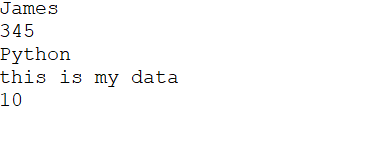
1. interface is keyword



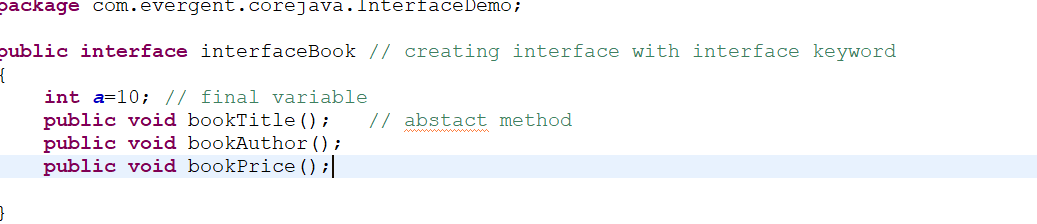
1. We can declare method interface are abstract
2. By default method signature is abstract
3. Any class implement interface that class should be overriden all interface method otherwise the class shown compile time error

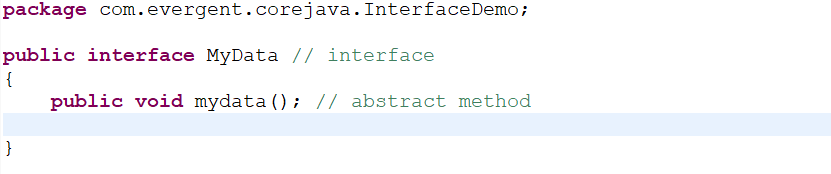


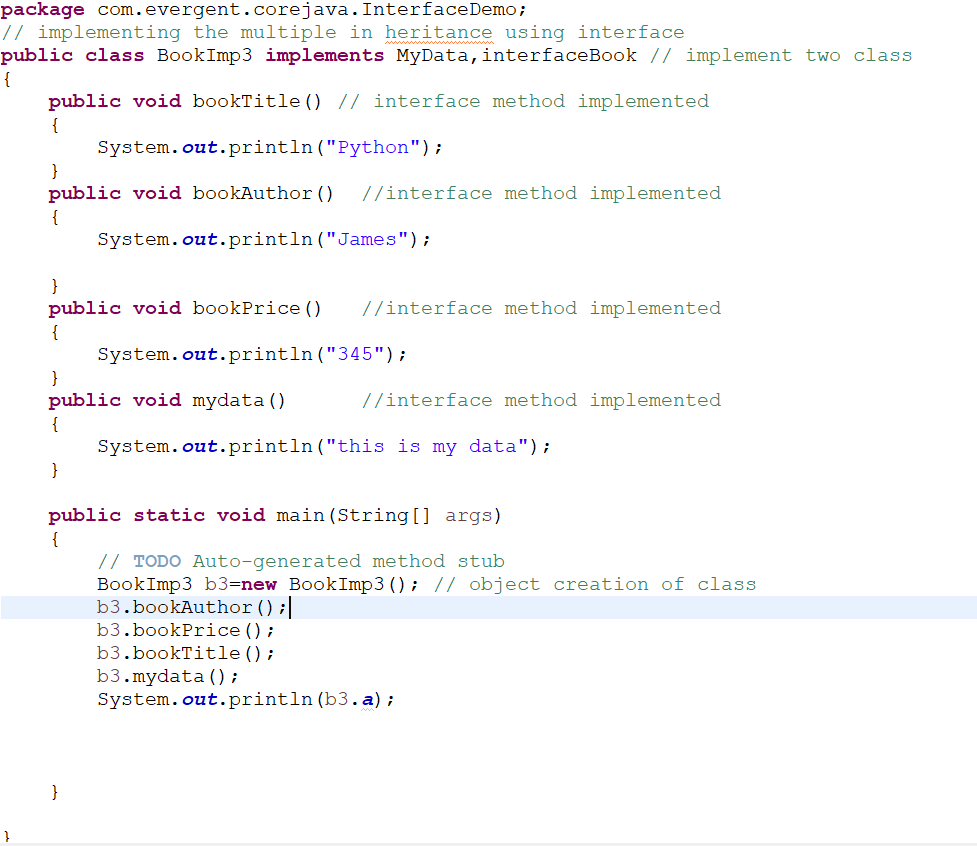


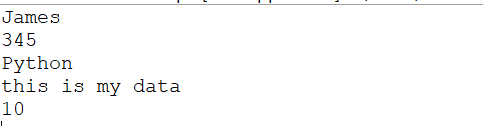


1. We cannot create object to interface but we can create reference to interface
2. We can declare variable inside interface all interface variable by default public static final
3. Java will support multiple inheritance through interface



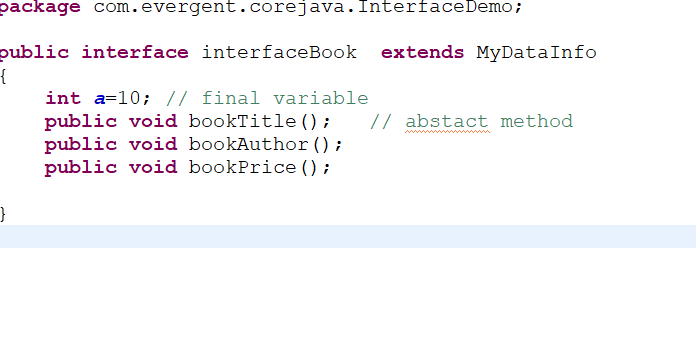


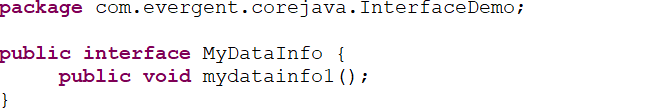




Viii)once class can implement interface

Ix)one interface can extends another interface



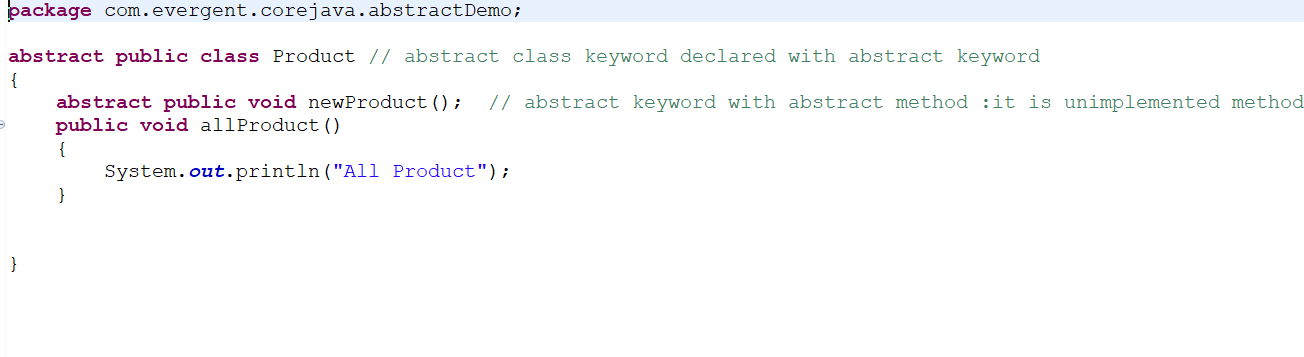
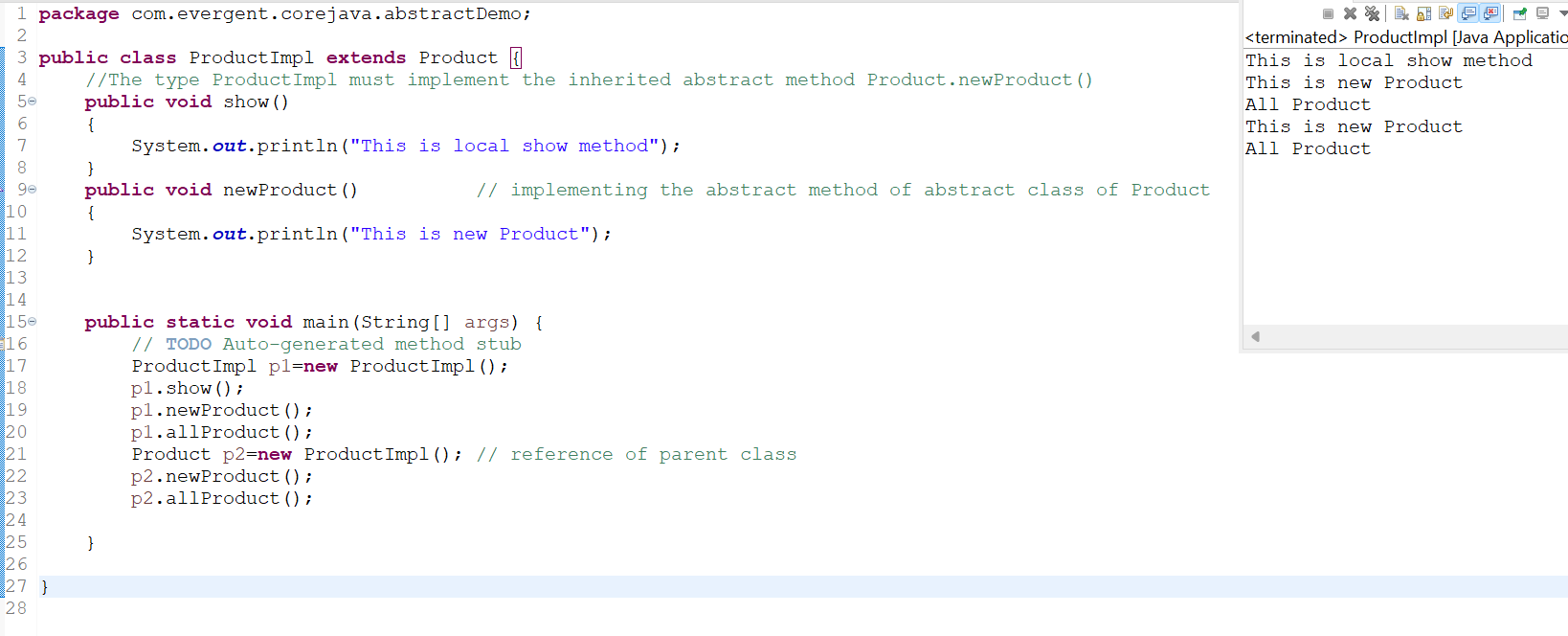
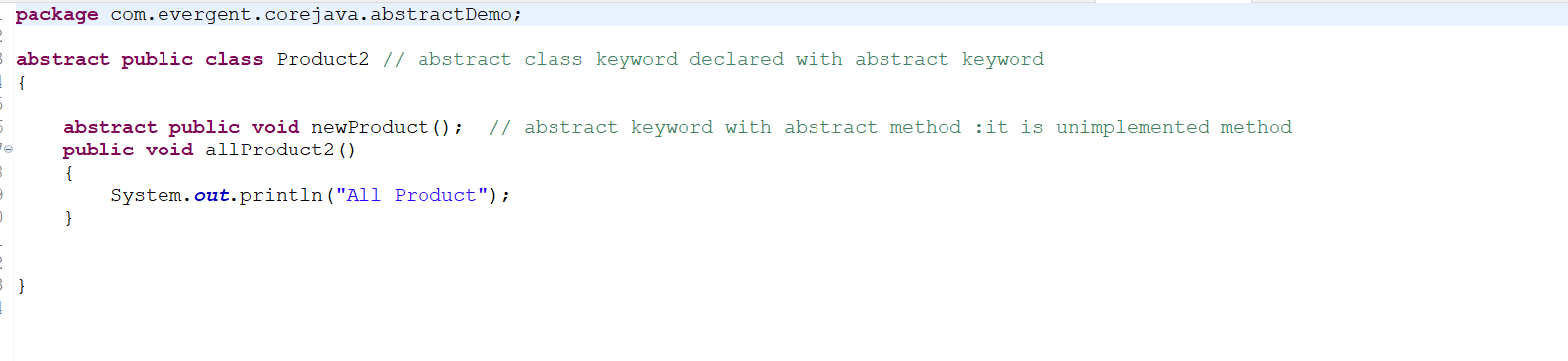
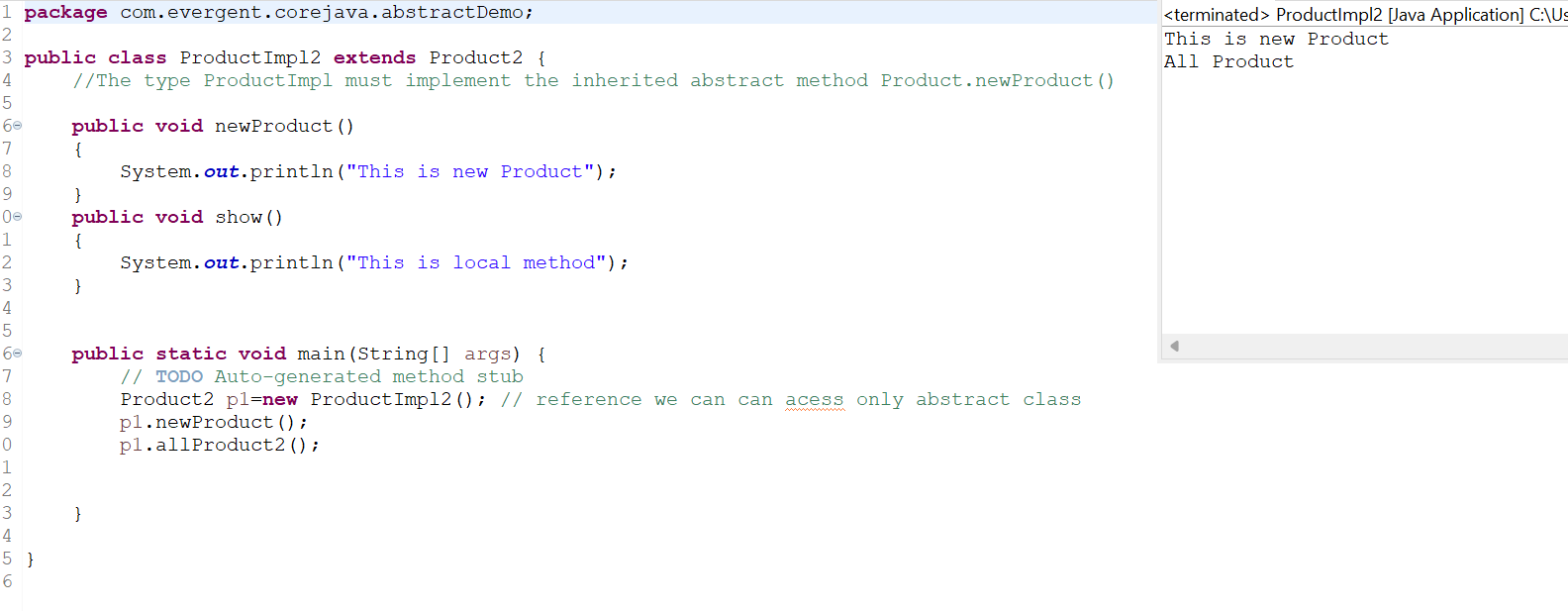
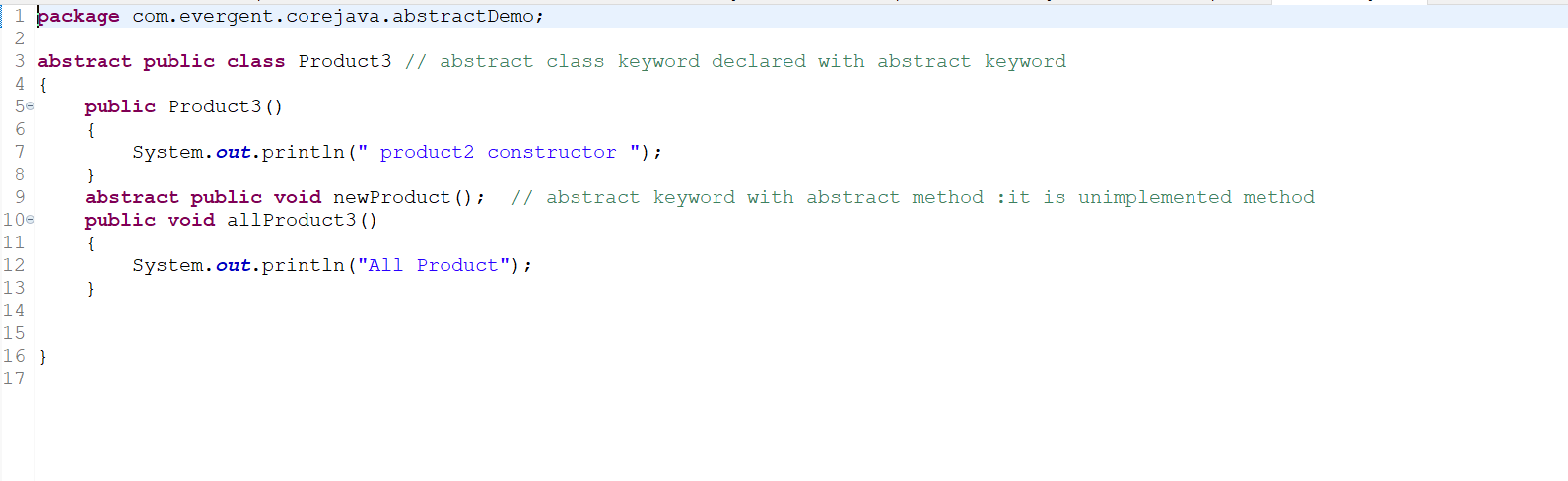
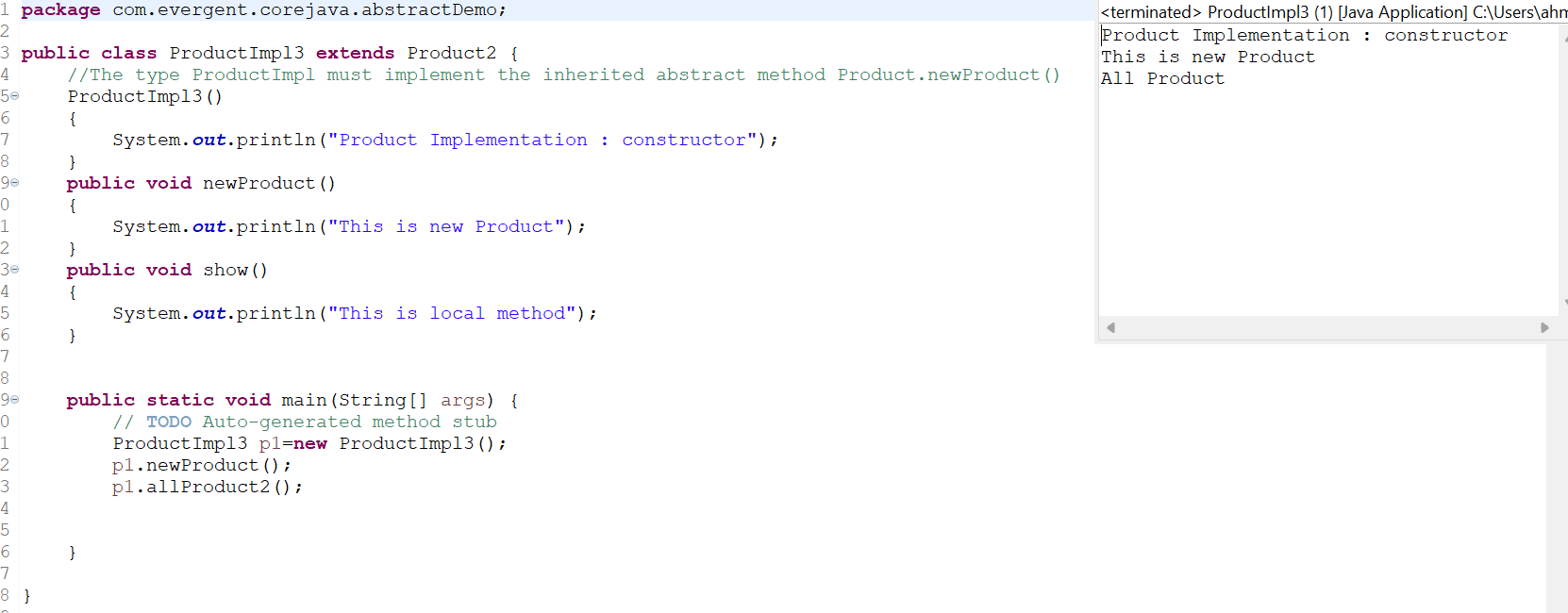


X)if we want onlu interface method but not local method.so we use reference to the interface

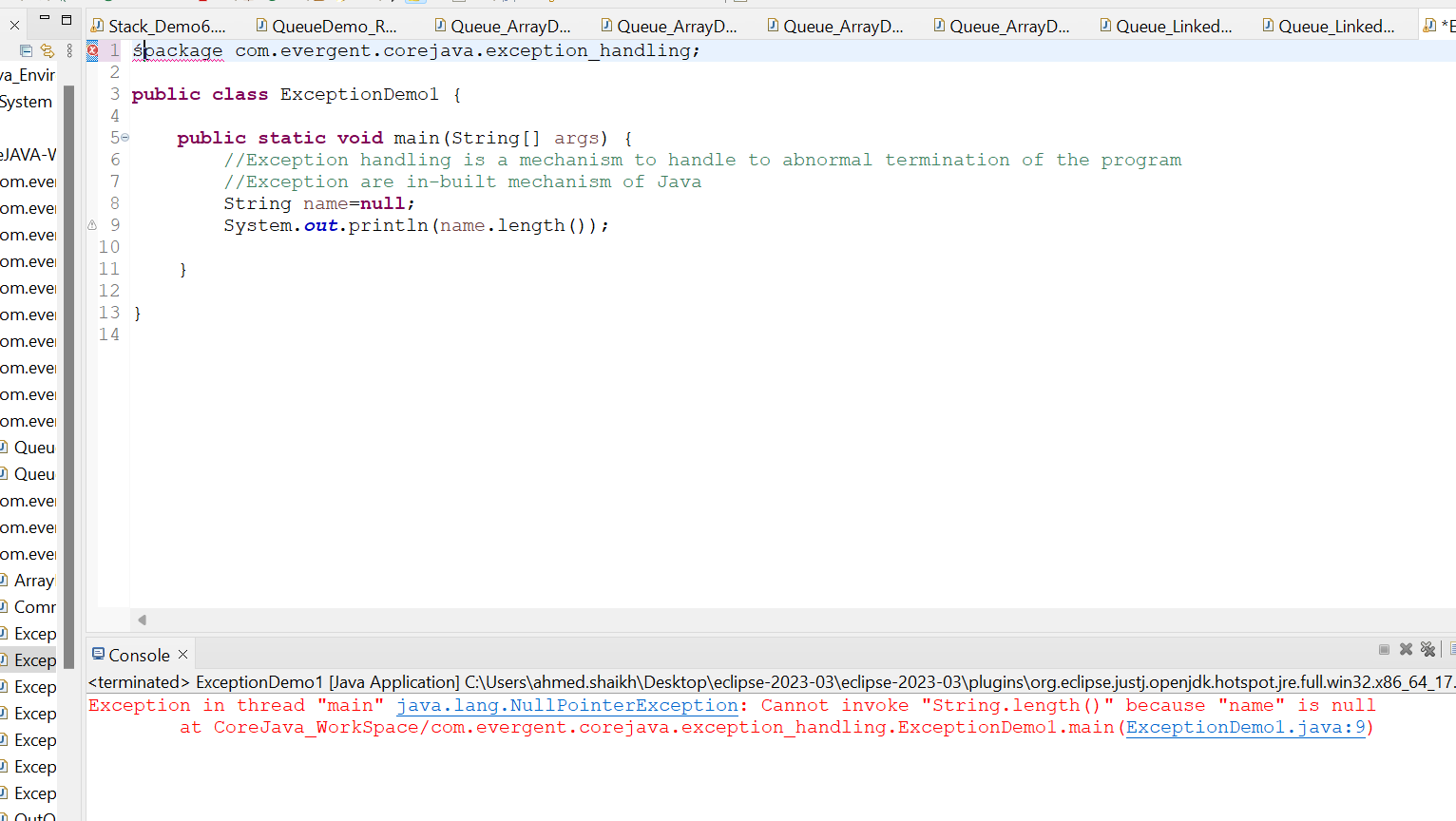
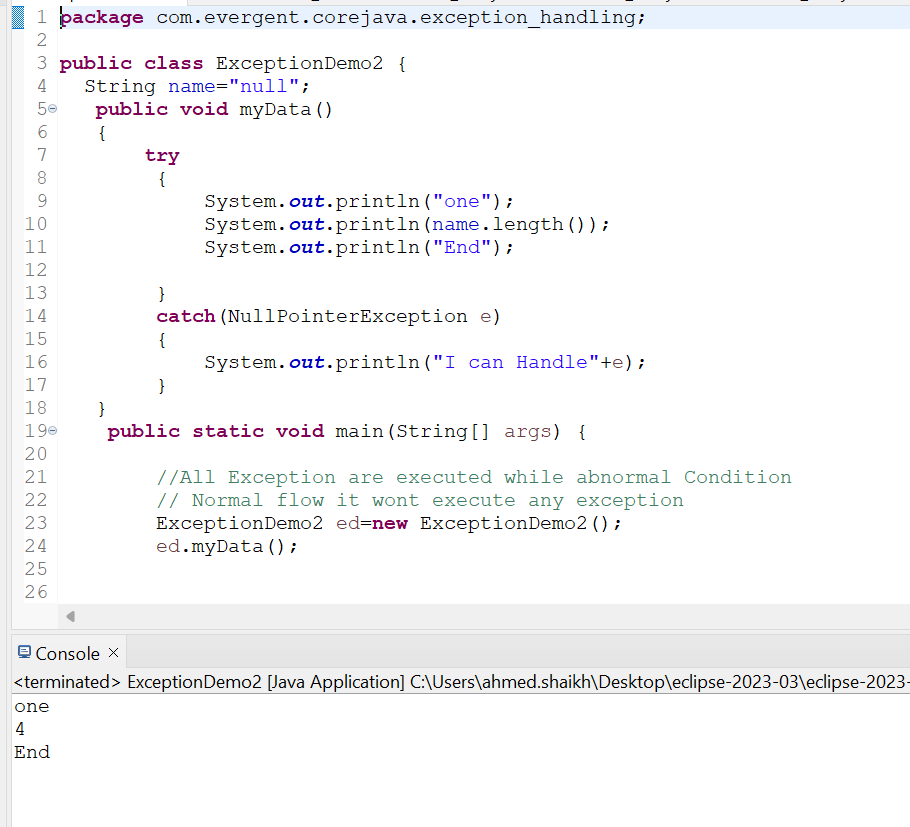
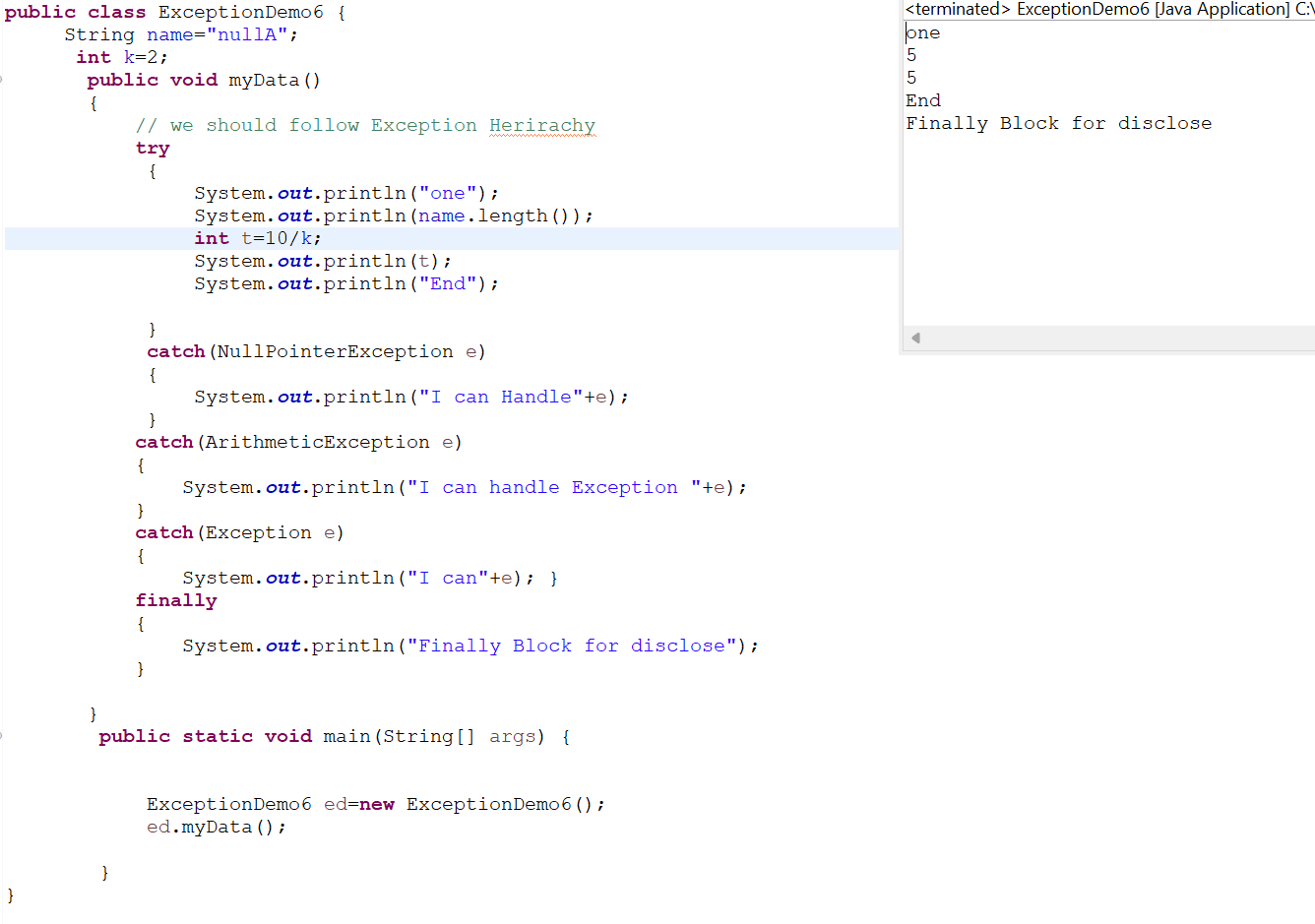
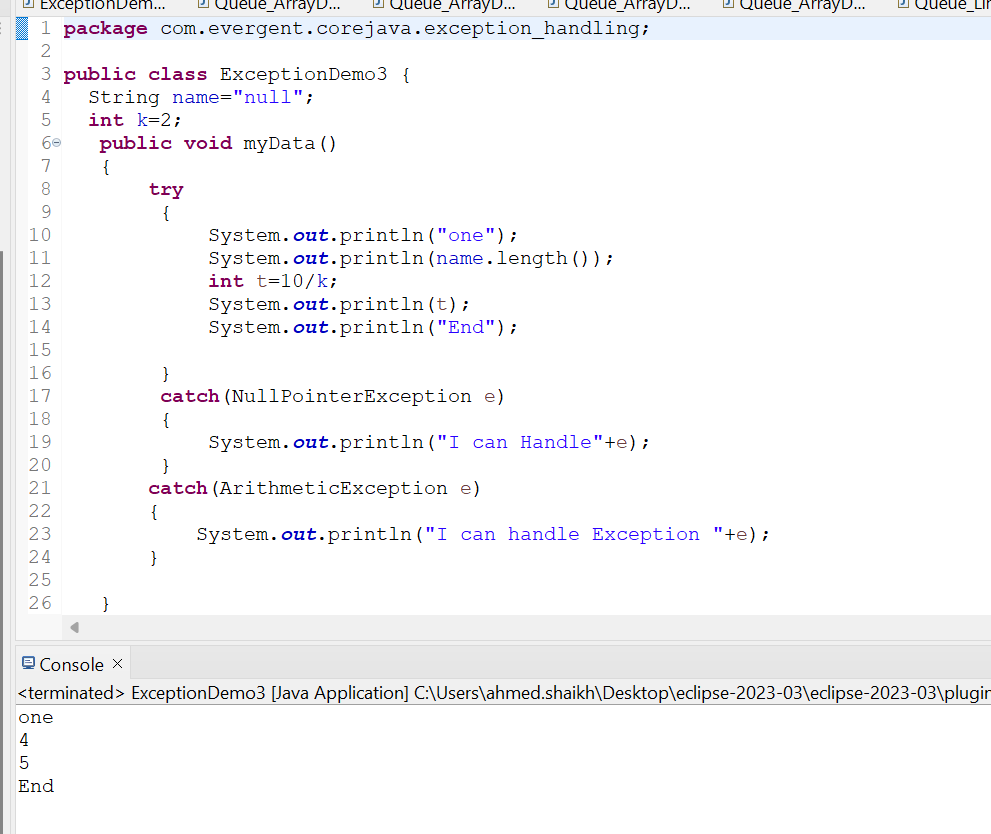
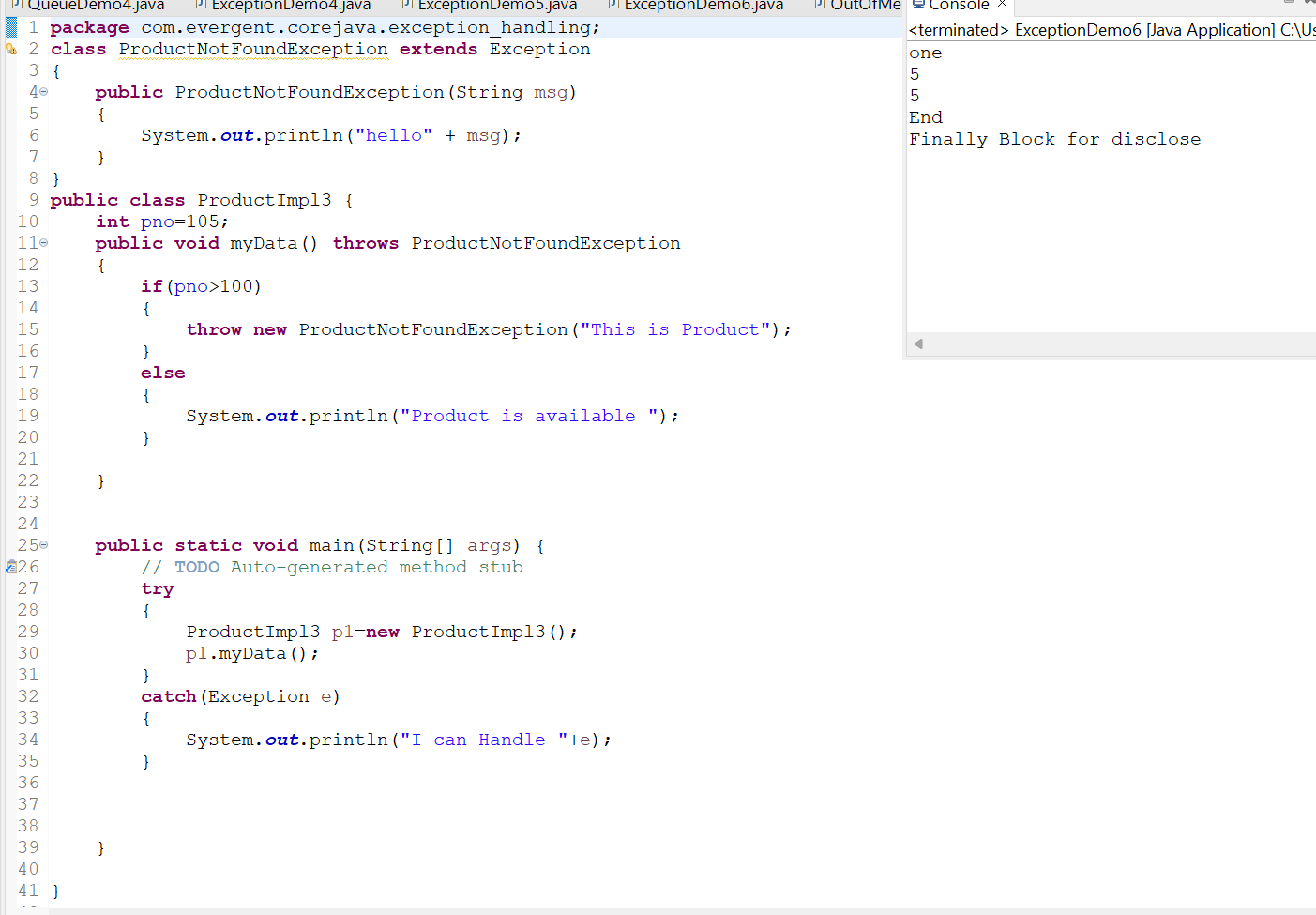
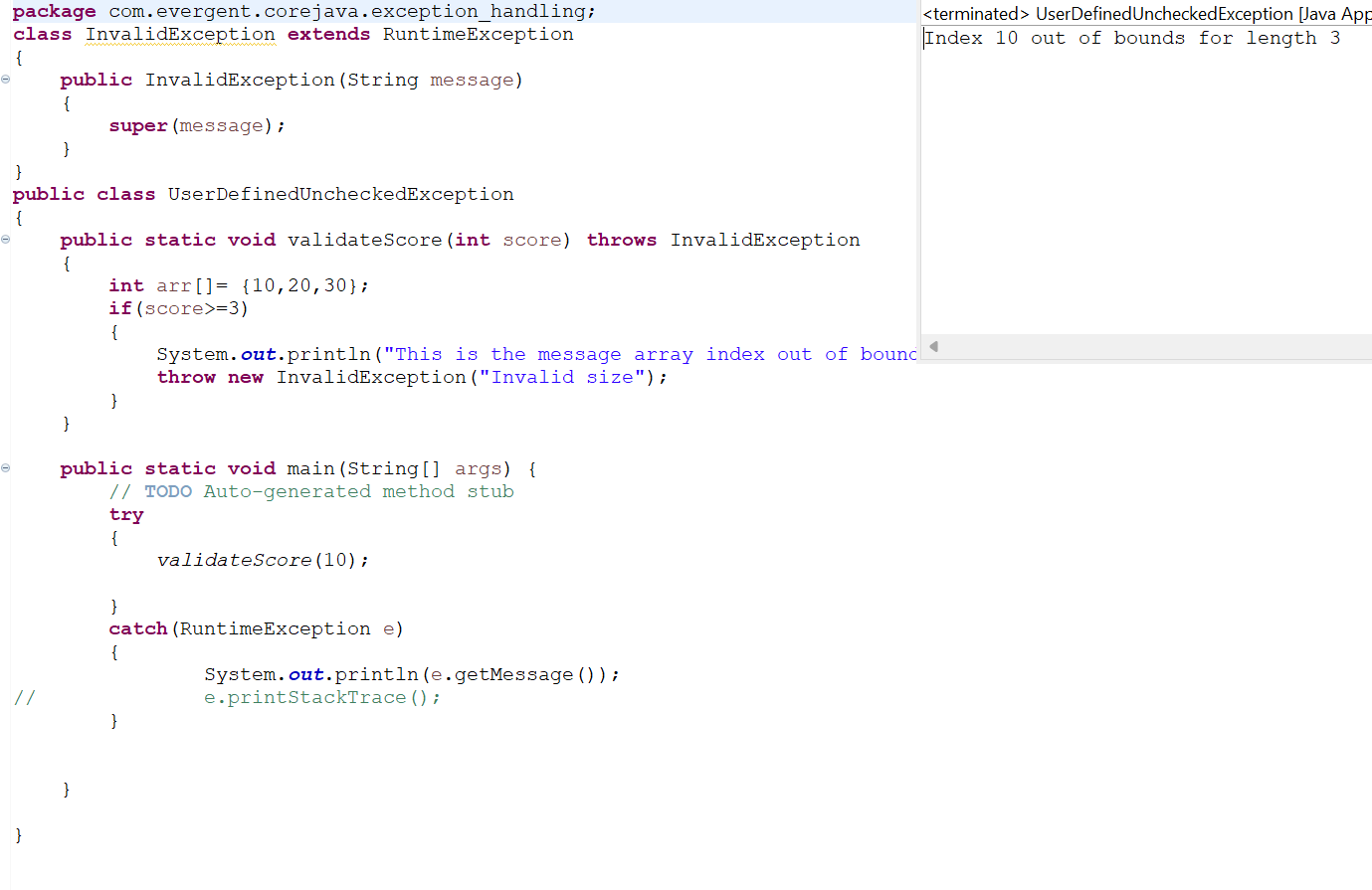
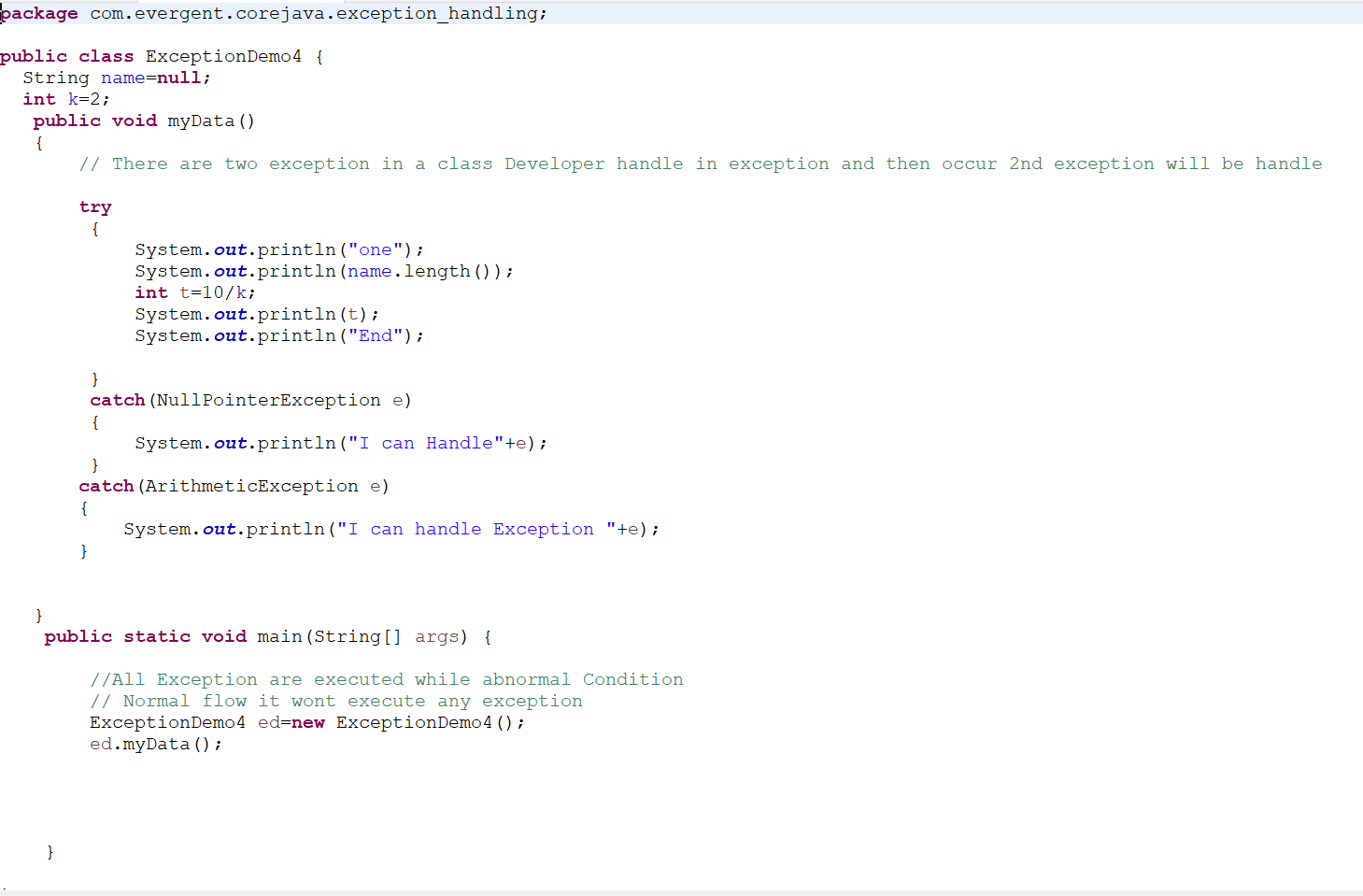
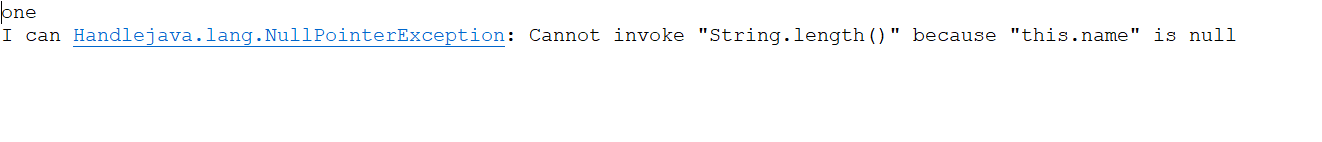
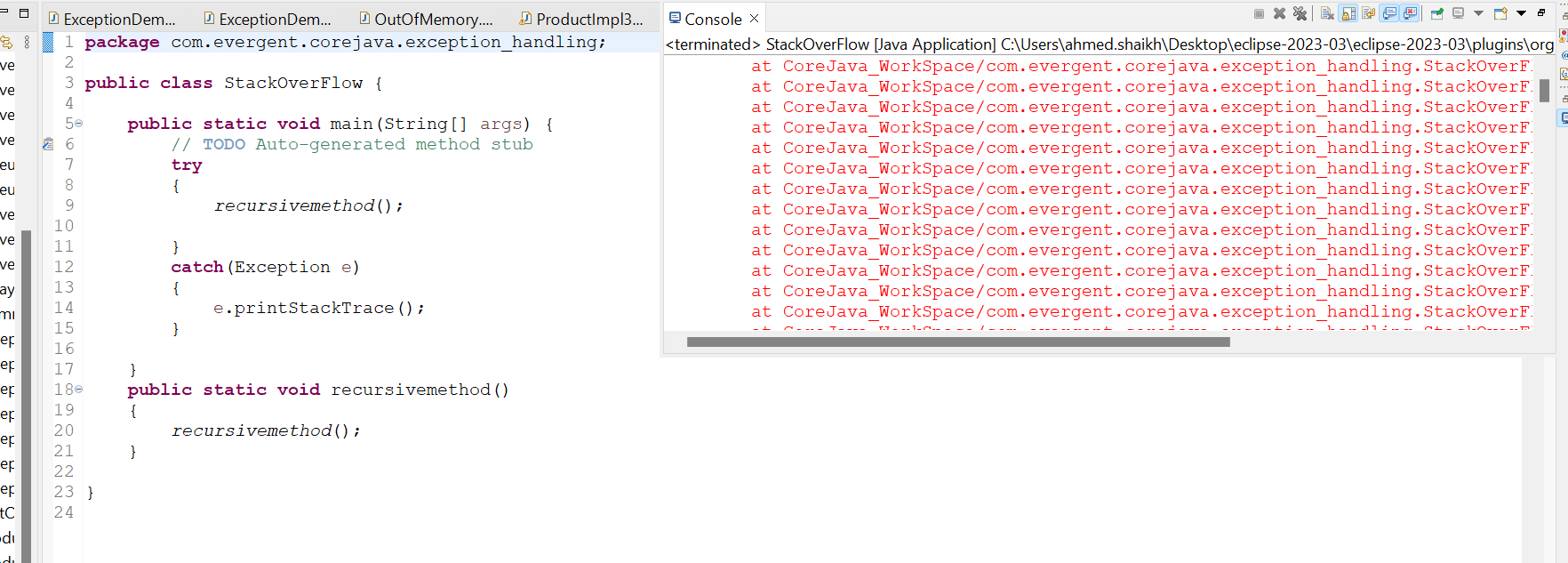
Xi)we can declare interface with zero method is called marker interface

Xii)clonable,seriazable interface.

**Abstract class**

1. abstract is Keyword
2. Abstract class having abstract method and concrete methods
3. 
4. If any class having one abstract method that class should be declared as abstract keyword other wise it show compile time error
5. 
6. If any class extends abstract that class should be override all abstract method otherwise that class will be shown compile time
7. We cannot create object to abstact class but we can create reference to it.
8. 
9. 
10. We Cannot create Constructor in abstract class but we can access SuperClass Object Present or not if it execute Constructor
11. We can Access Abstract class Constructor through subclass object
12. 
13. 

Exception Handling

1. Exception Handling is mechanism
2. Exception are in-built mechanismof java
3. All Exception are Excecuted While abnormal Condition only
4. 
5. Normal Flow it won’t execute Any Exception
6. Once any Exception are occuring in java Code the remaining line of code is reachable
7. 
8. They are two types of Exception in Java
9. A)Checked Exception B)Unchecked Exception
10. Checked Exception are declared are Compile Time Exception
11. All Unchecked are Runtime Exception
12. Java.lang.Throwable() is super class For Exception and error
13. They are 5 KeyWords in Exception Handling
14. A)Try
15. B)Catch
16. C)Finally
17. D)Throws
18. E)Throw
19. Try for bussiness Logic and Statement
20. Catch for handling Exception
21. 
22. Finally is block exceptionis occuror not finally will be executed
23. 
24. Throws is an exception will be executed method by method
25. Throw is for runtime Exception will call Predefined Exception
26. 
27. Try followed either catch block or finally block
28. We should follow Exception herarichical
29. 
30. We can create our own user defined Exception
31. 
32. We can create our own exception extends Exception or Runtime Exception
33. 
34. All Exception are in into Java.lang Package
35. A class have two Exception one Exception remaining Code is unreachable it handled
36. 
37. 
38. Error are not Handled in our Developer Control
39. 
40. Handling Multiple Exception with throws
41. Handling Multiple Exception using Muti-Catch
42. We can use try with in the try it is nested try