**JAVA TRAINING INDEX**

**Date:5/08/2024**

**Day-1**

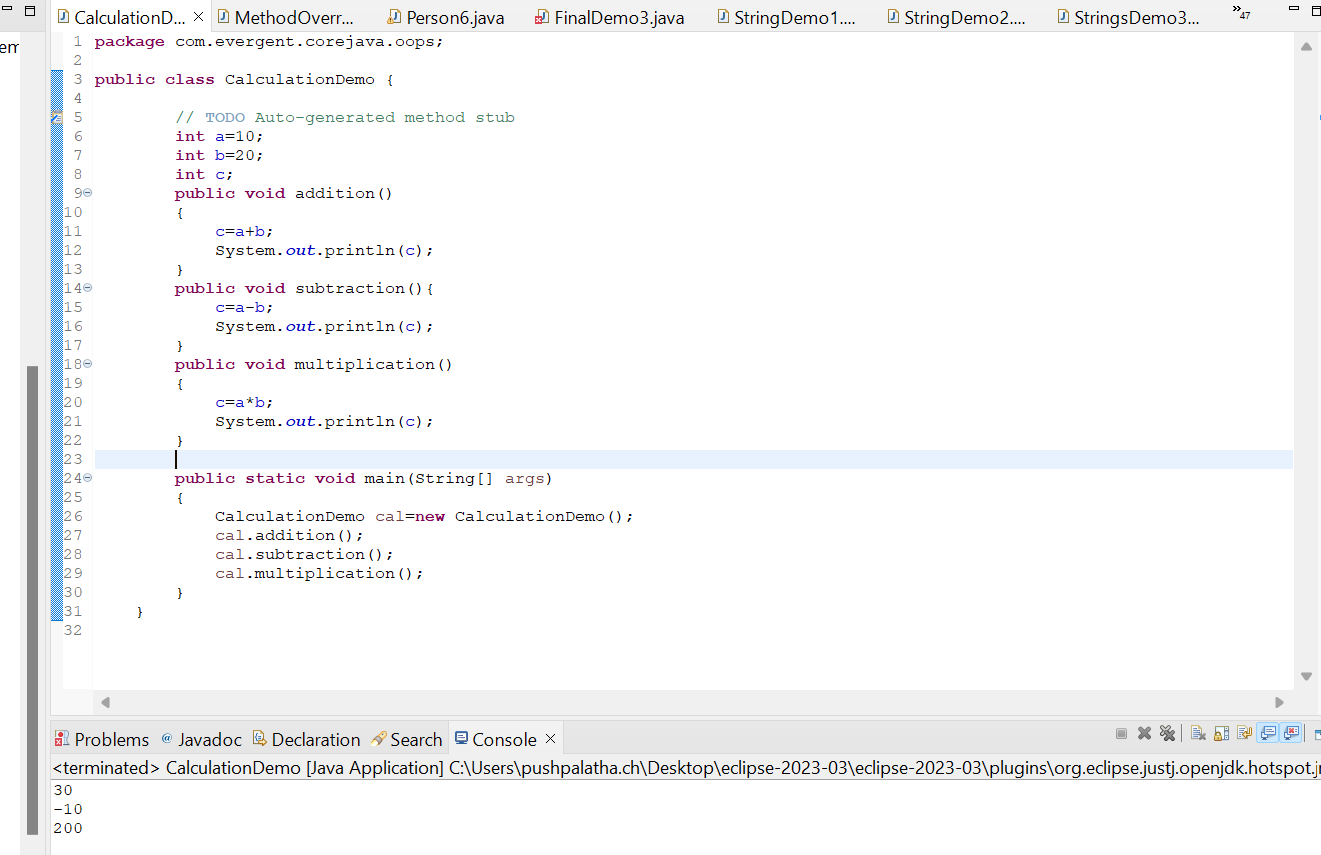
1.languages and applications

2.Java Features

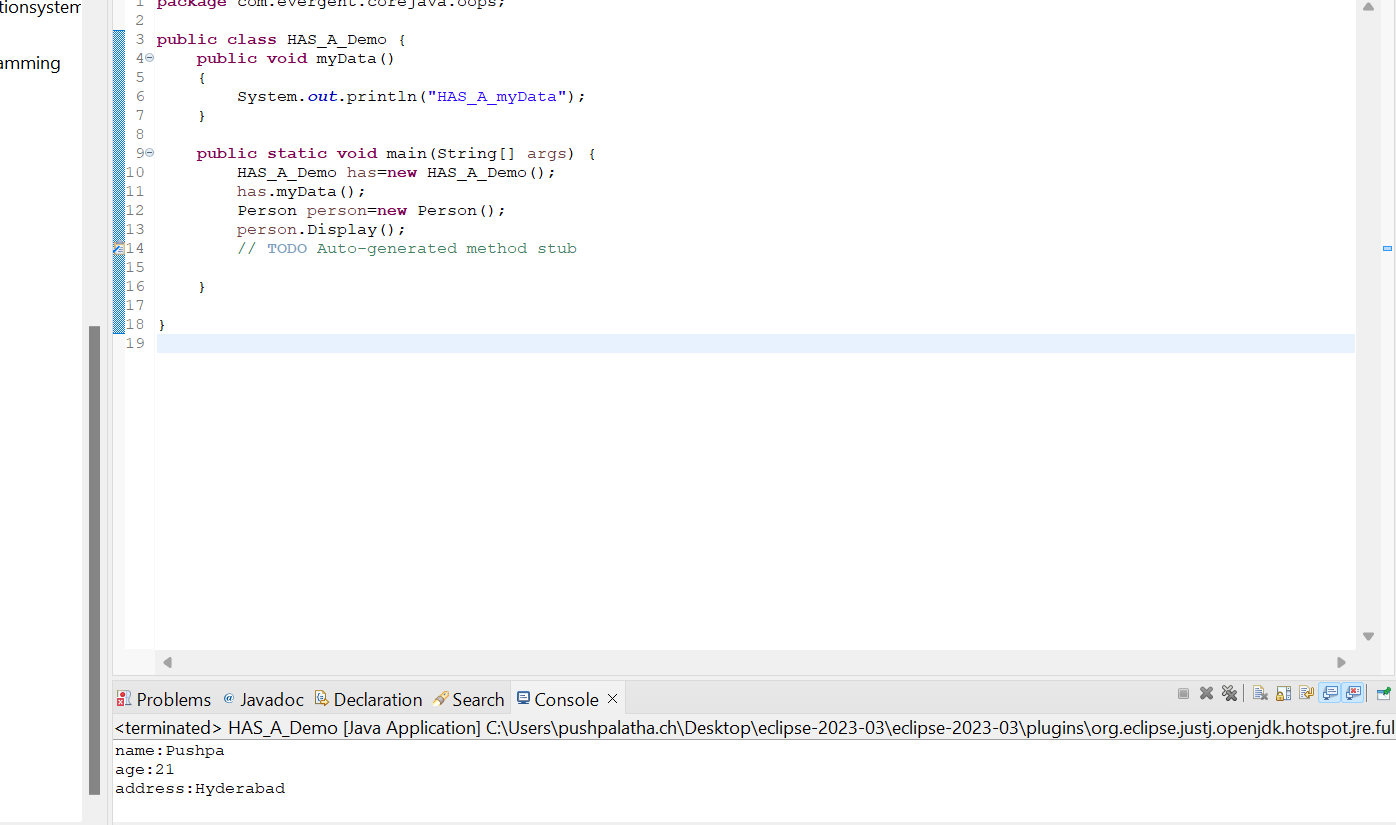
a.Why Java is platform Independent

b.oops

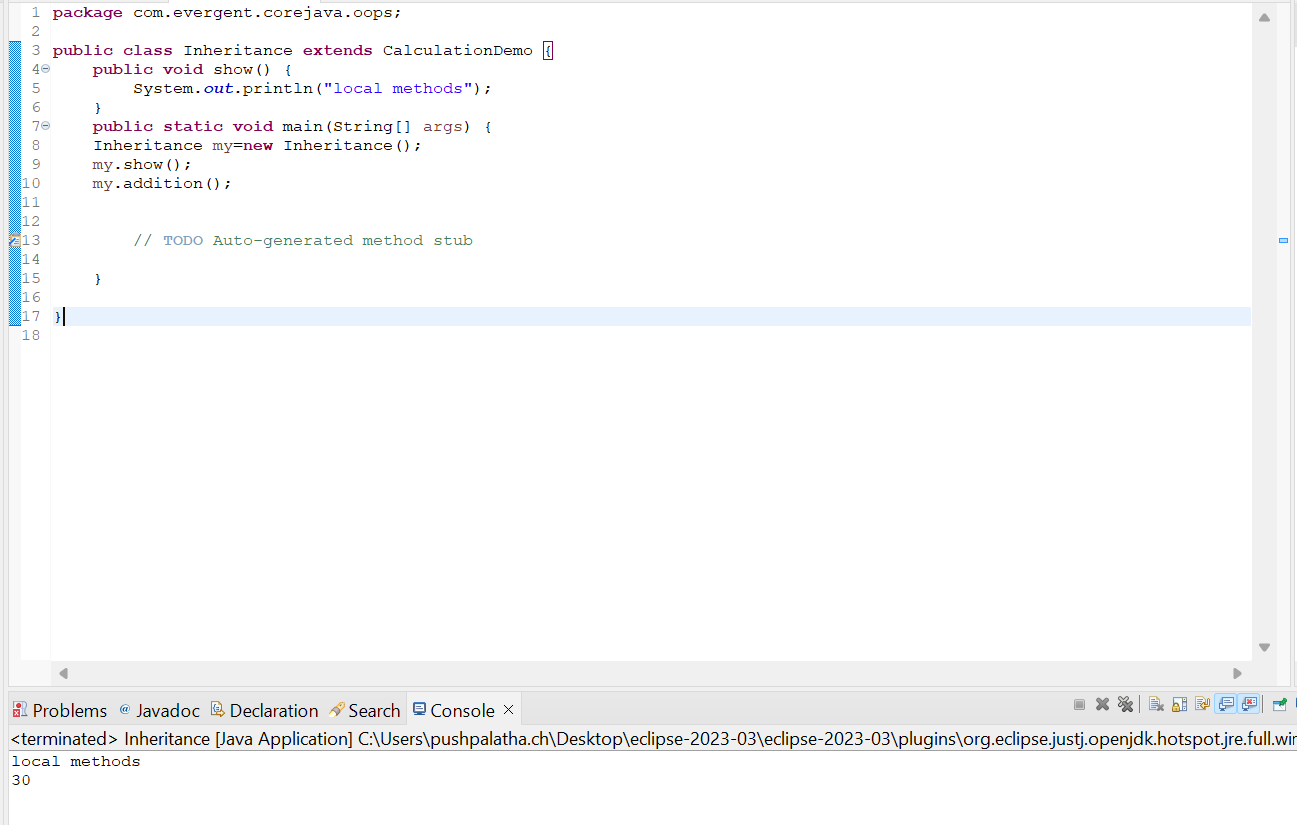
1)Performing arithmetic calculations.



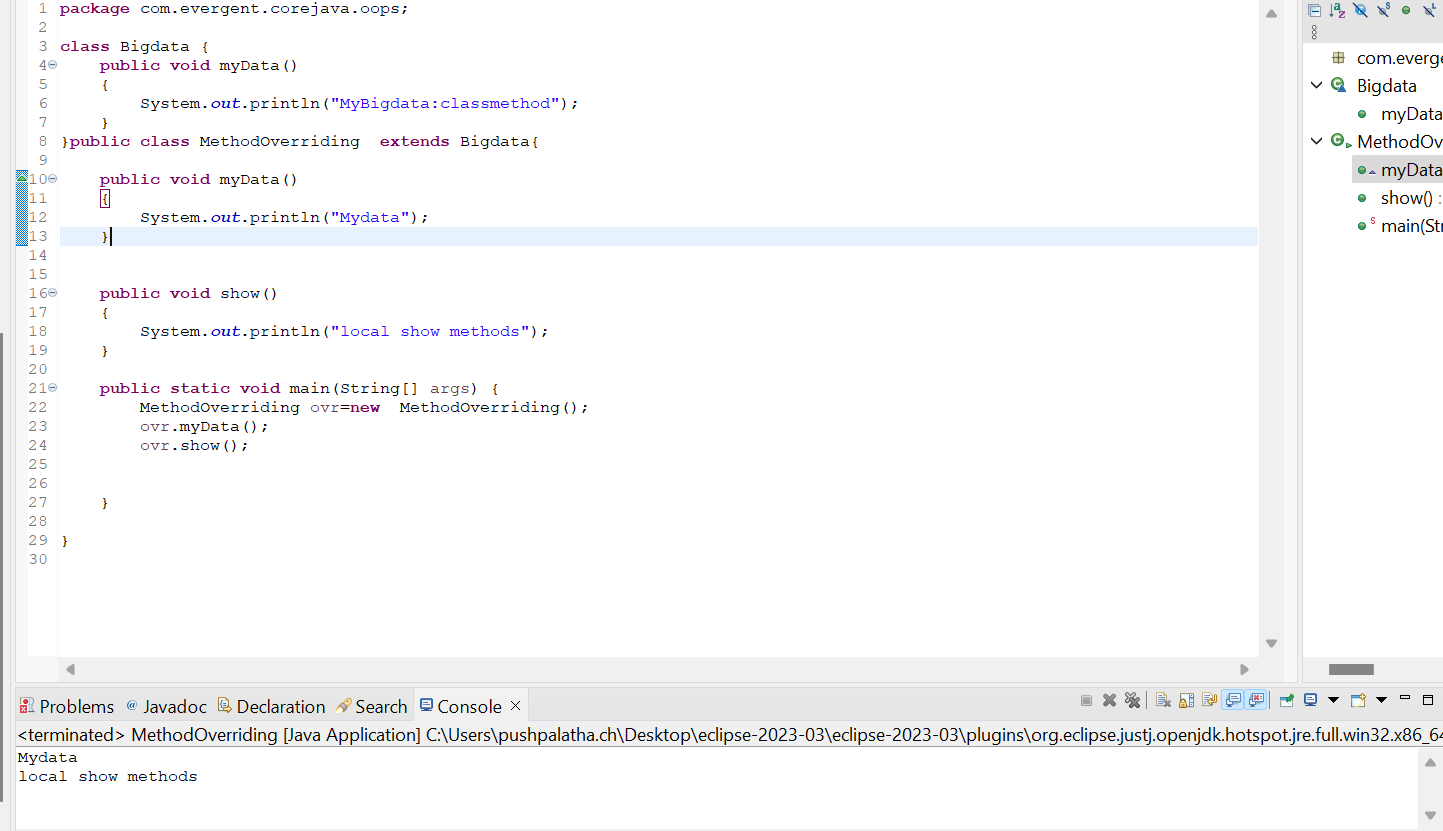
1. HAS\_A(based on object creation we can call the method from the class)



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

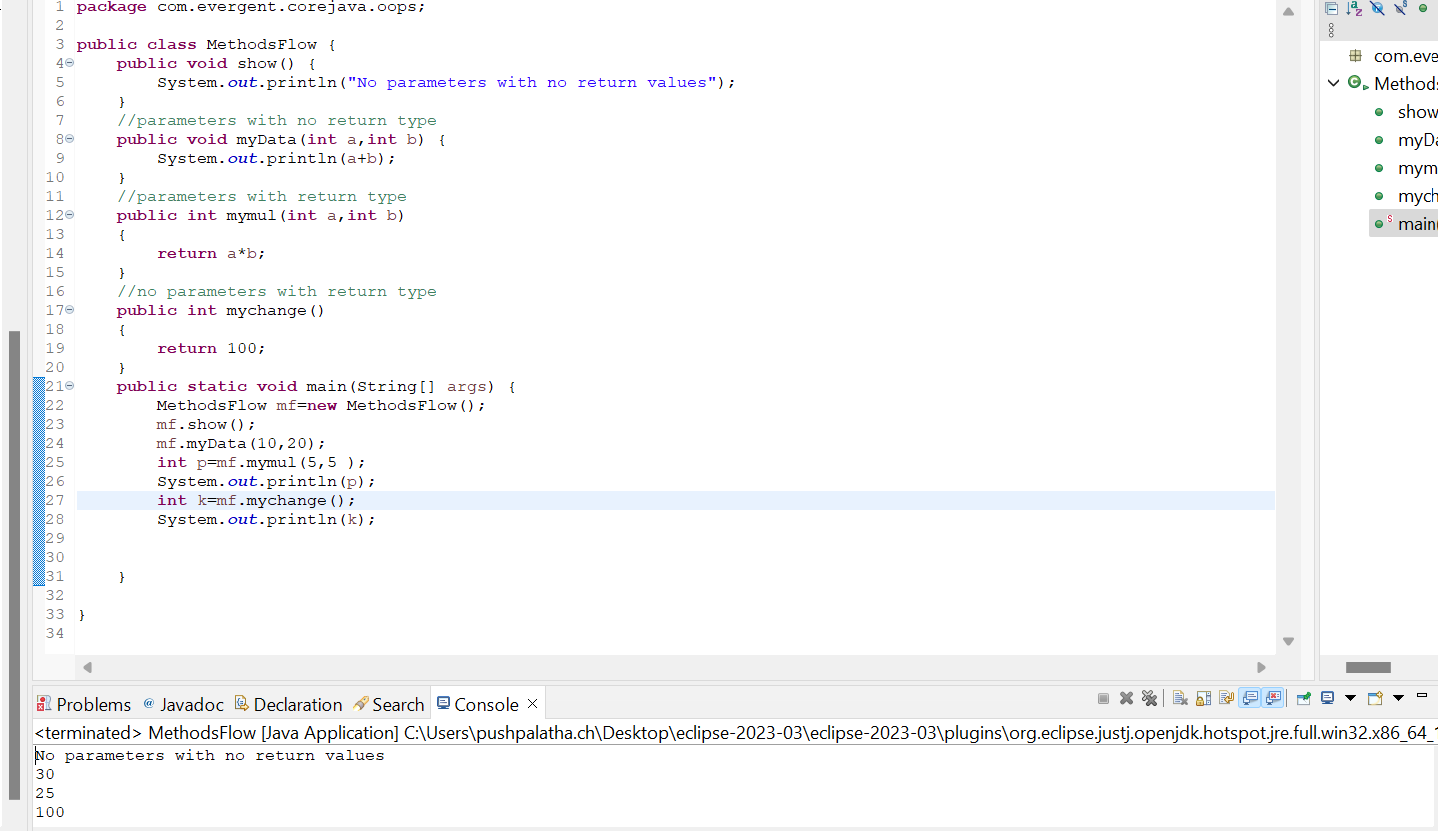
Basically we can call methods in four ways:

1)Methods with parameters without return type.

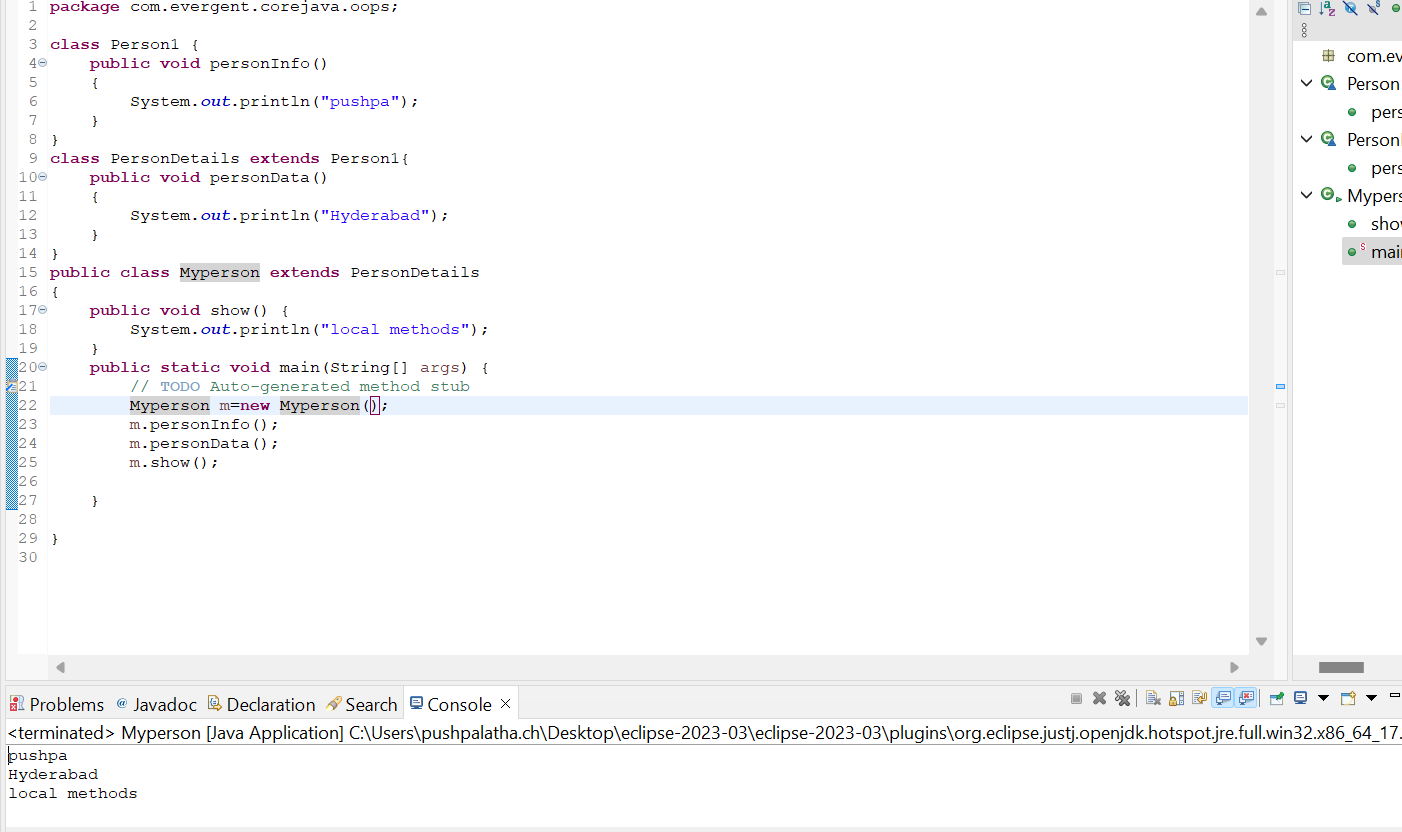
2)Methods with parameters with return type.

3)Methods without parameters without return type.

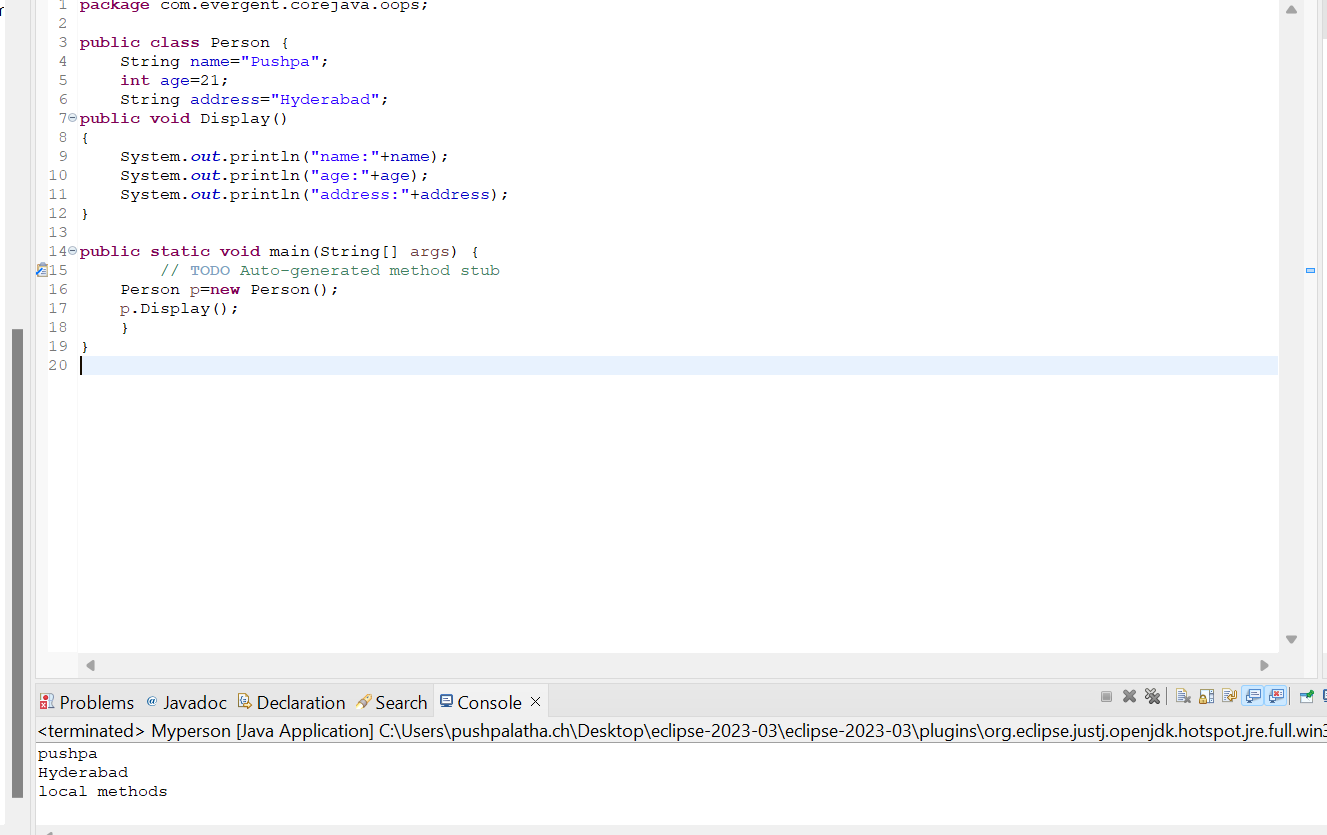
4)Methods without parameters with return type.



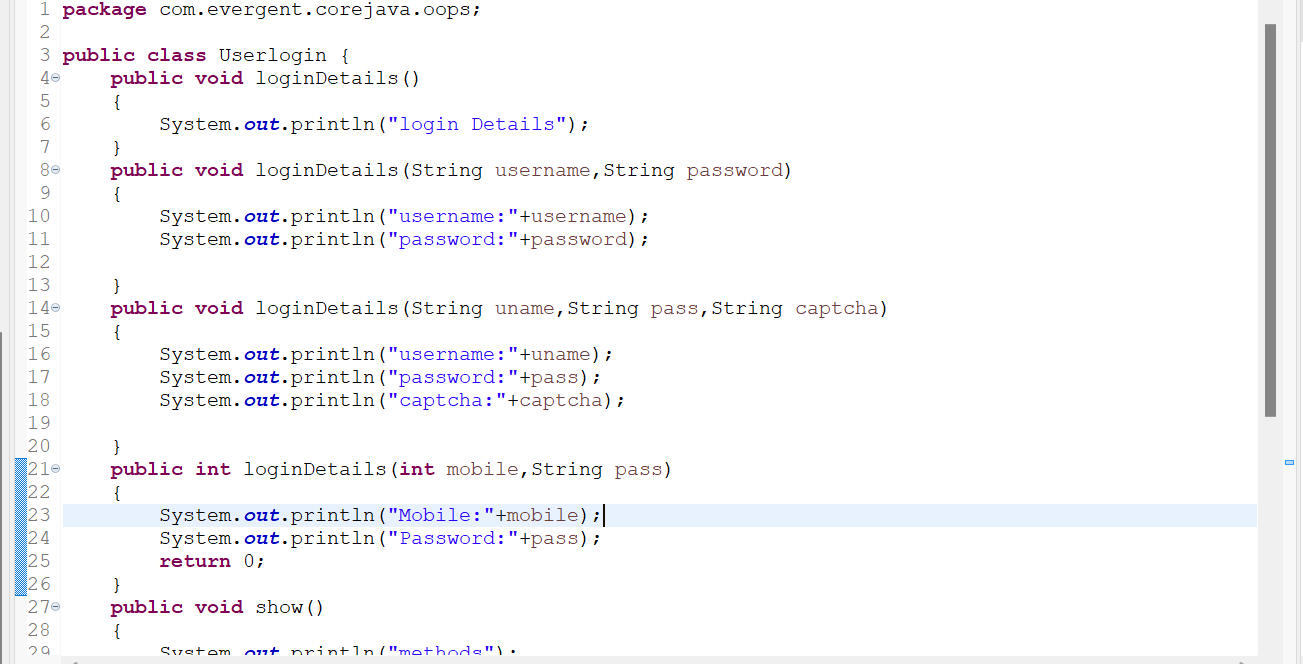
6.Multi Level Inheritance

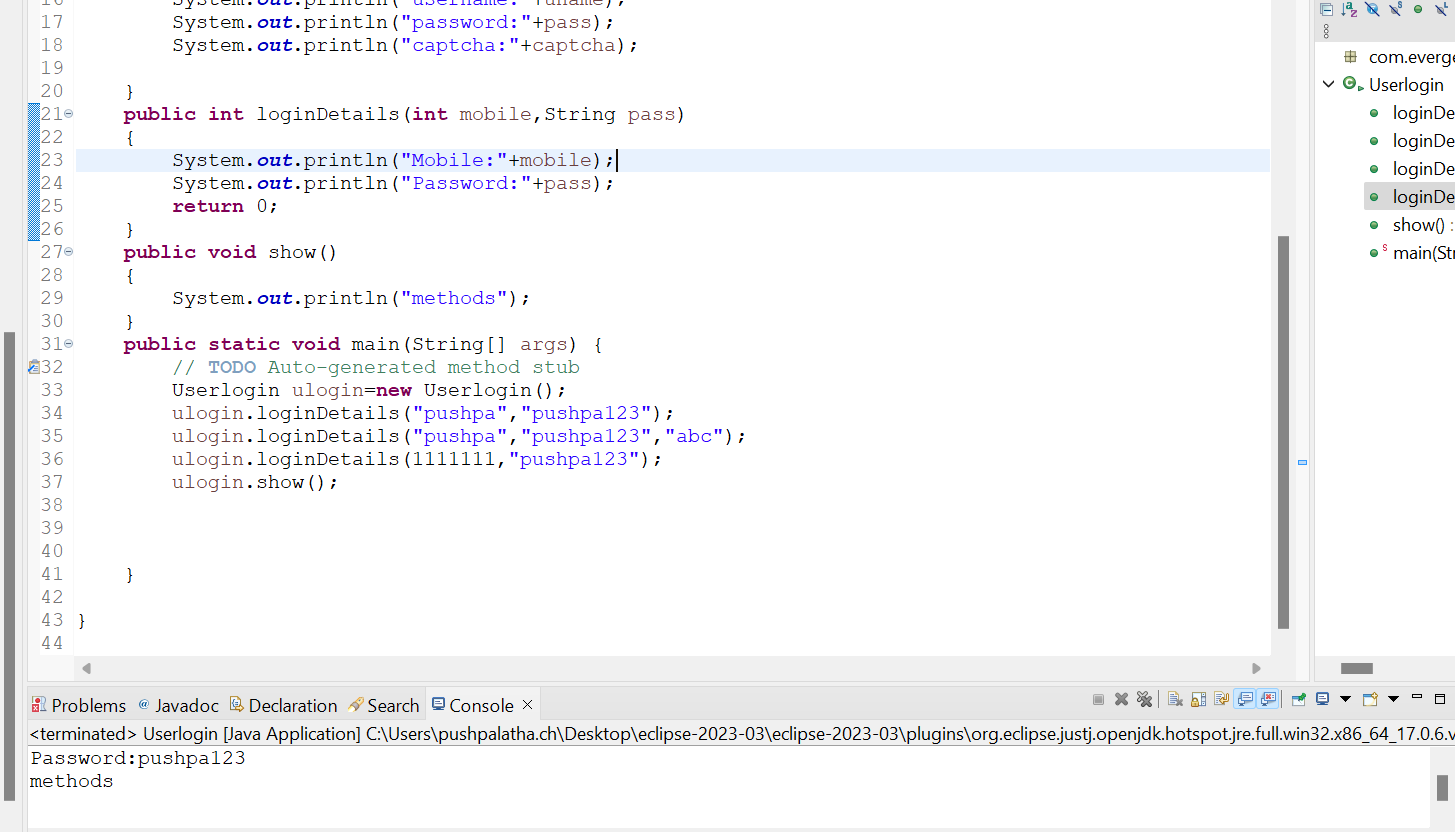


1. OOPS concept Example



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





c.Exception handling

d.Multithreading

e.web application

f.open source

g.security

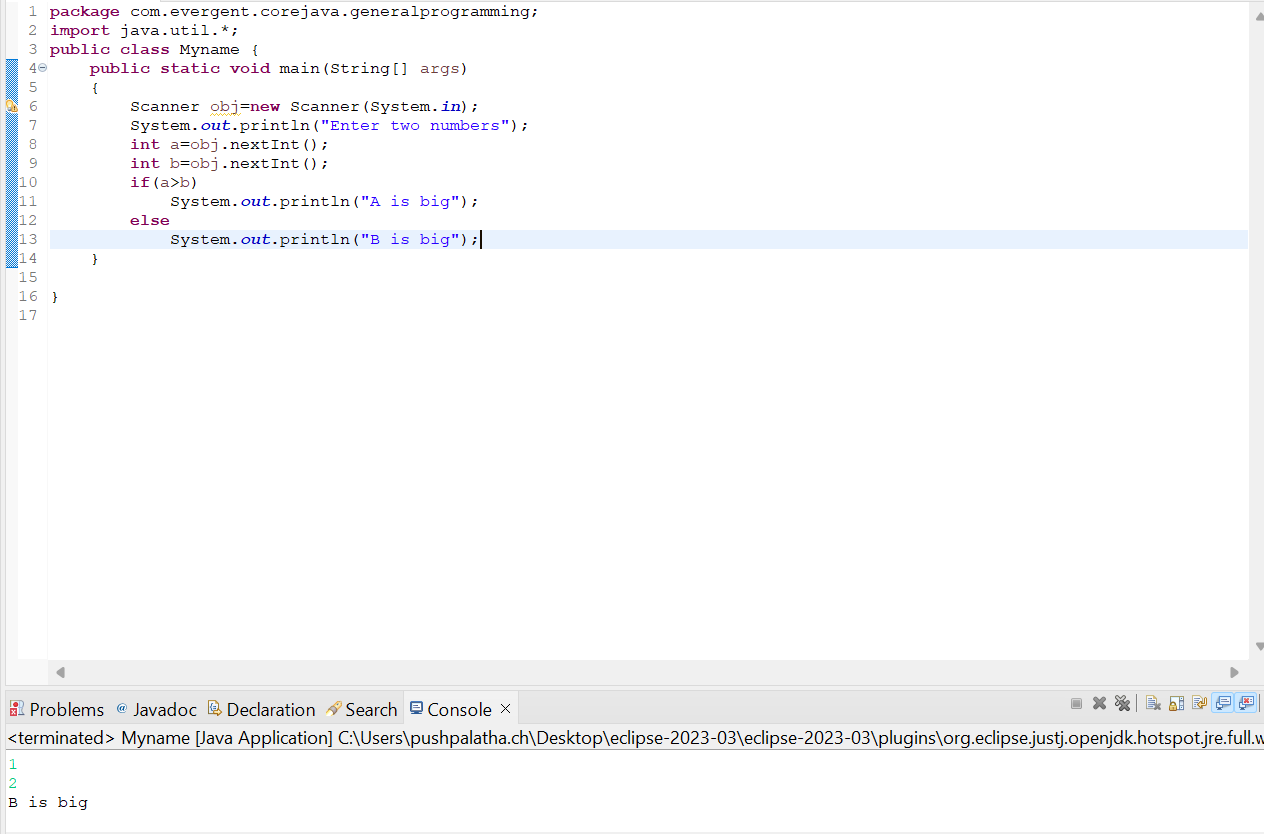
h.supports networking

i.memory management

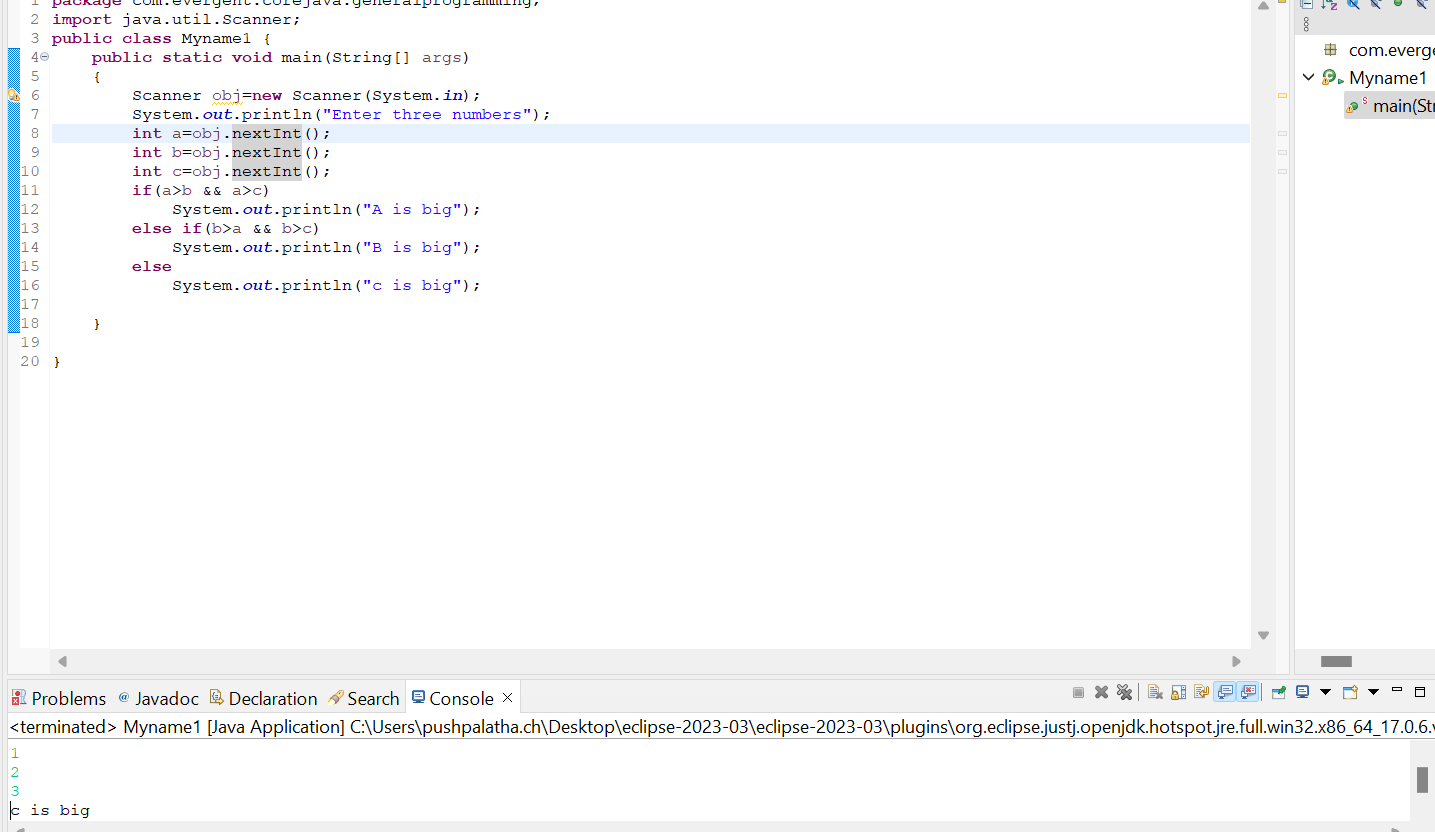
3. JDK,JRE,JVM

4.Basic programming

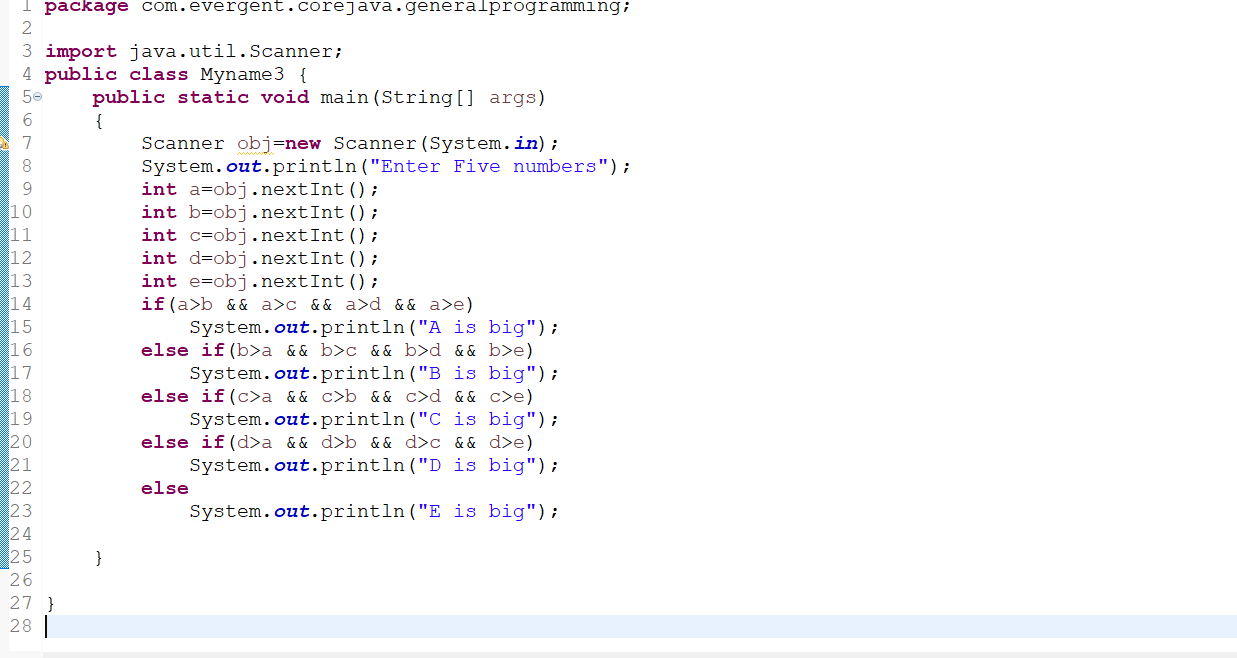
Biggest of 2 numbers

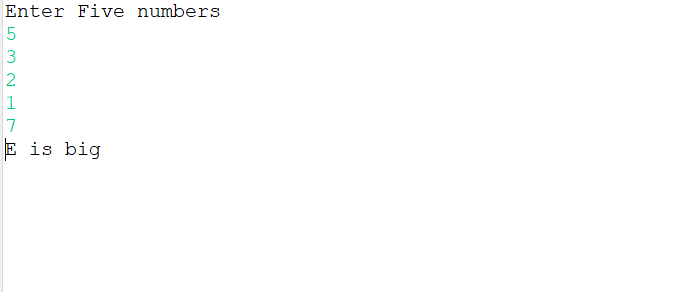


Biggest of 3 numbers

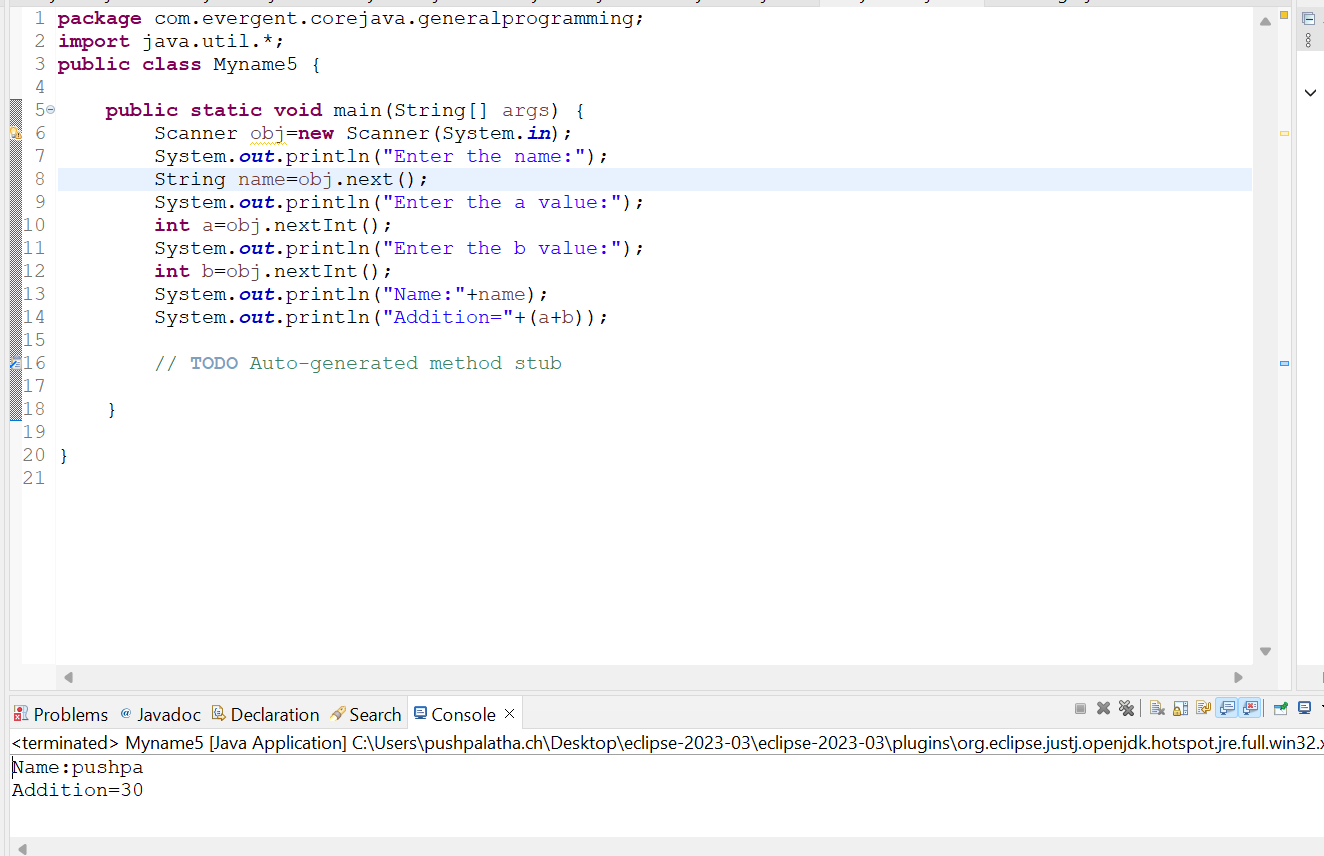


Biggest of 5 numbers





Print Addition and name



Swapping of 2 numbers without and with temp



5.Packages.

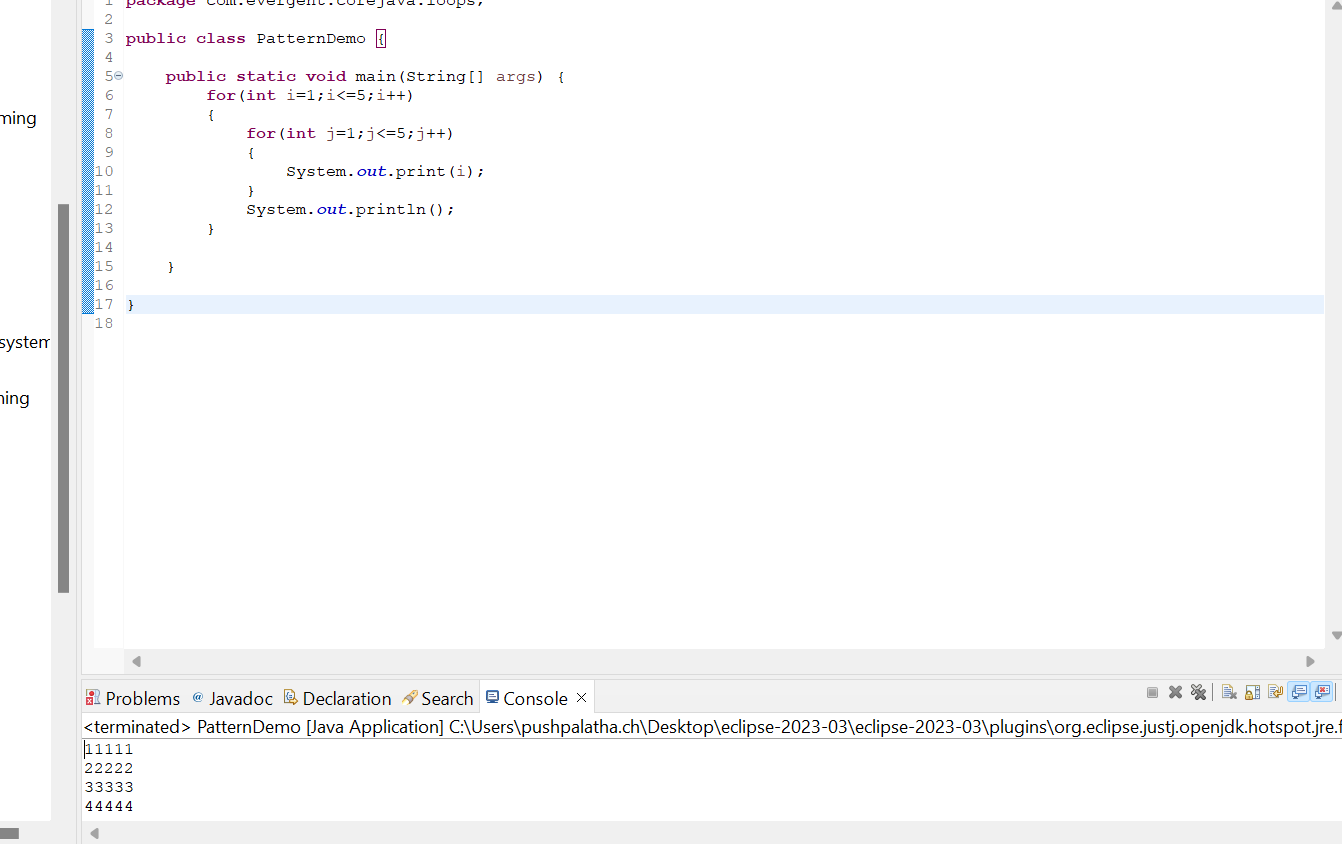
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Date:06/08/2024**

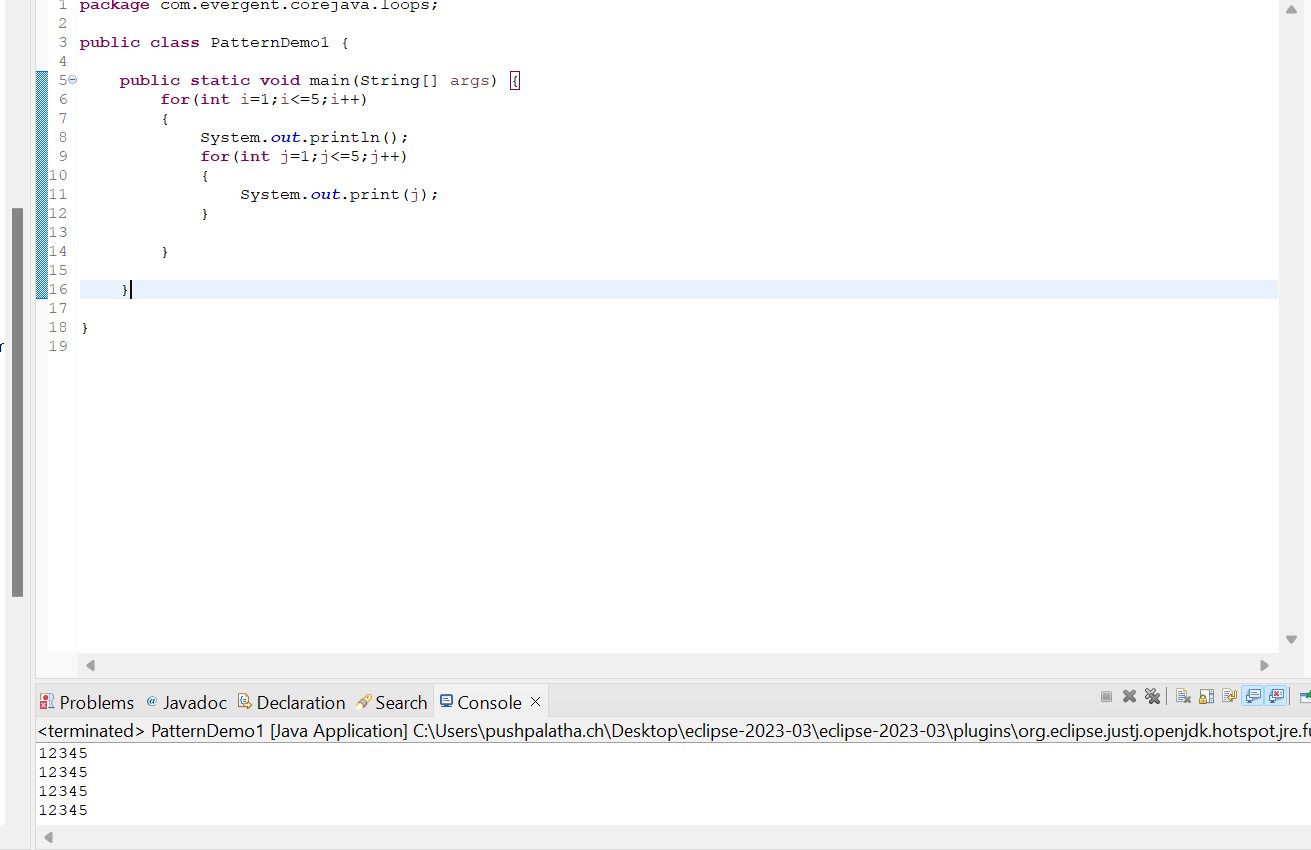
**Day-2**

1. Nested loops

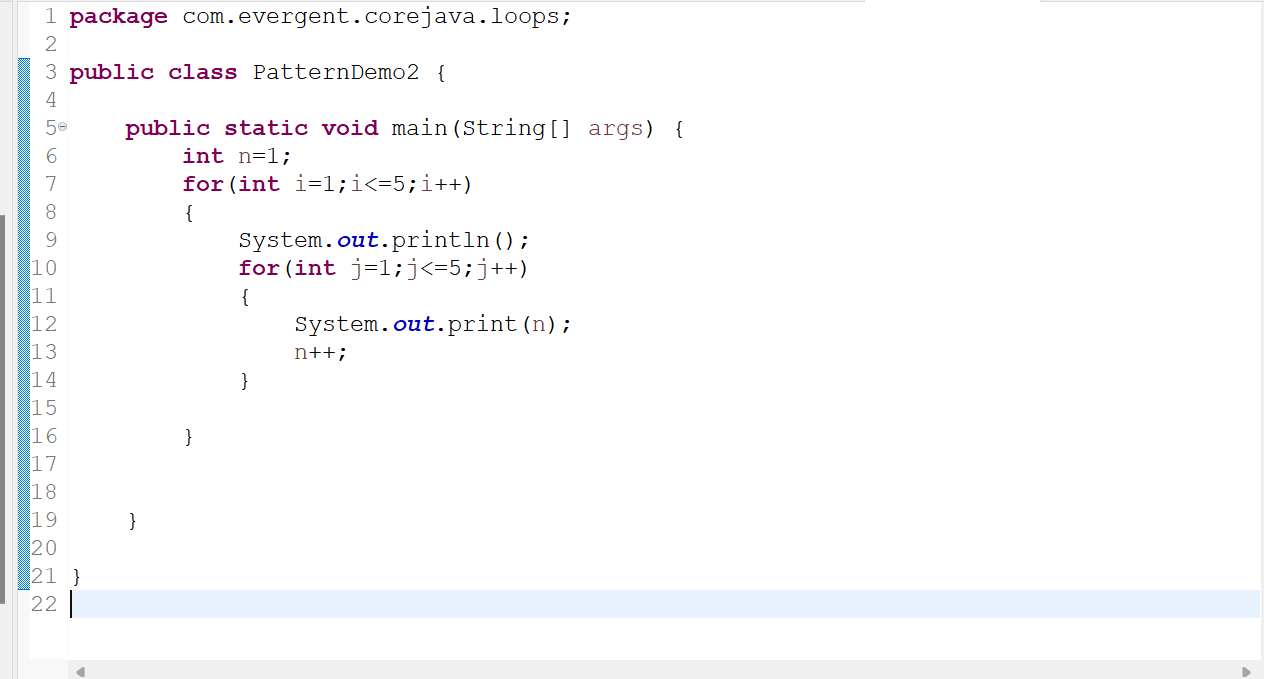
1.Pattern1

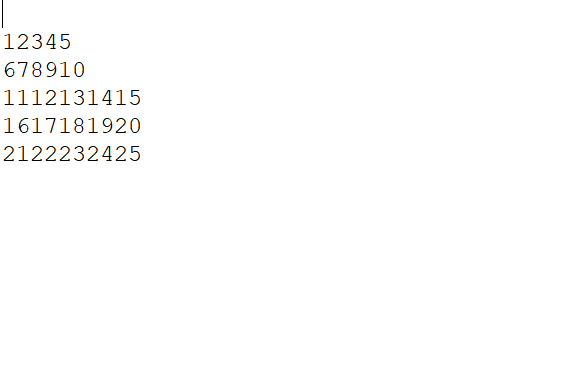


Pattern 2



Pattern 3

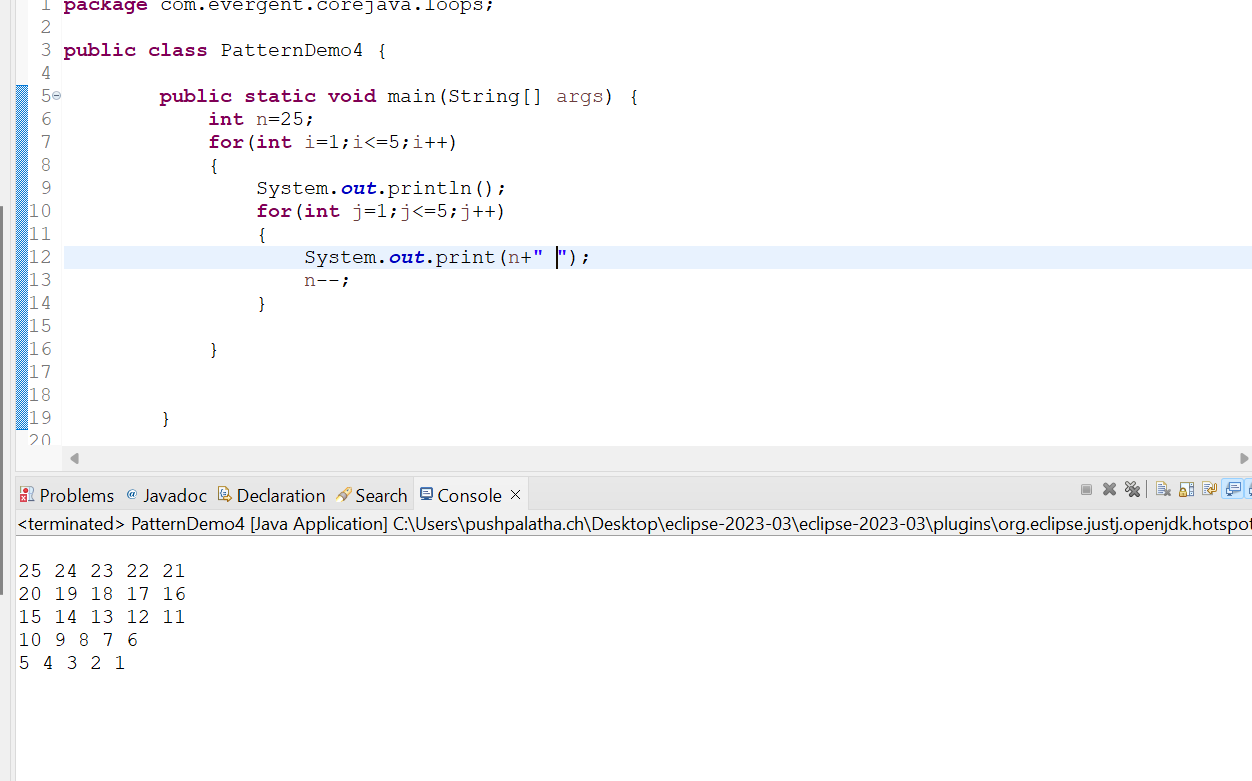




Pattern 4



Pattern 5

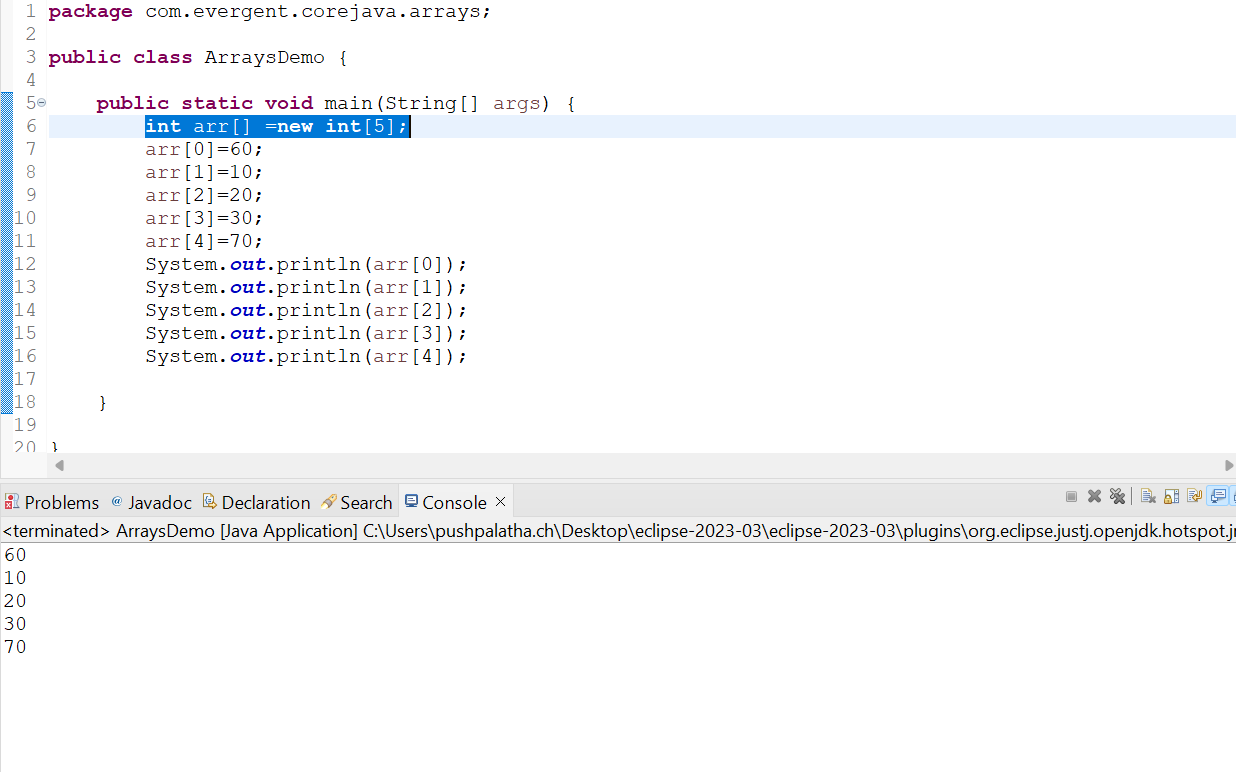


2.Arrays(collection of Homogenous data types stored in contiguous memory allocation with fixed size.)

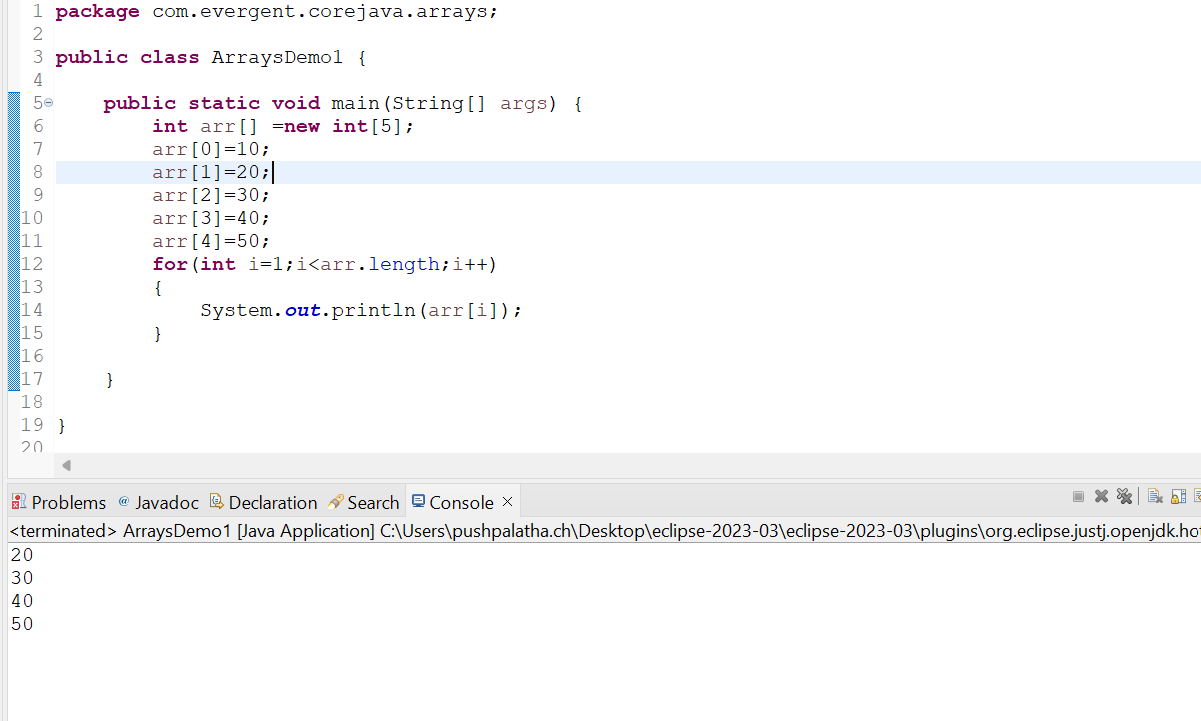
a.single dimensional array

b.double dimensional array

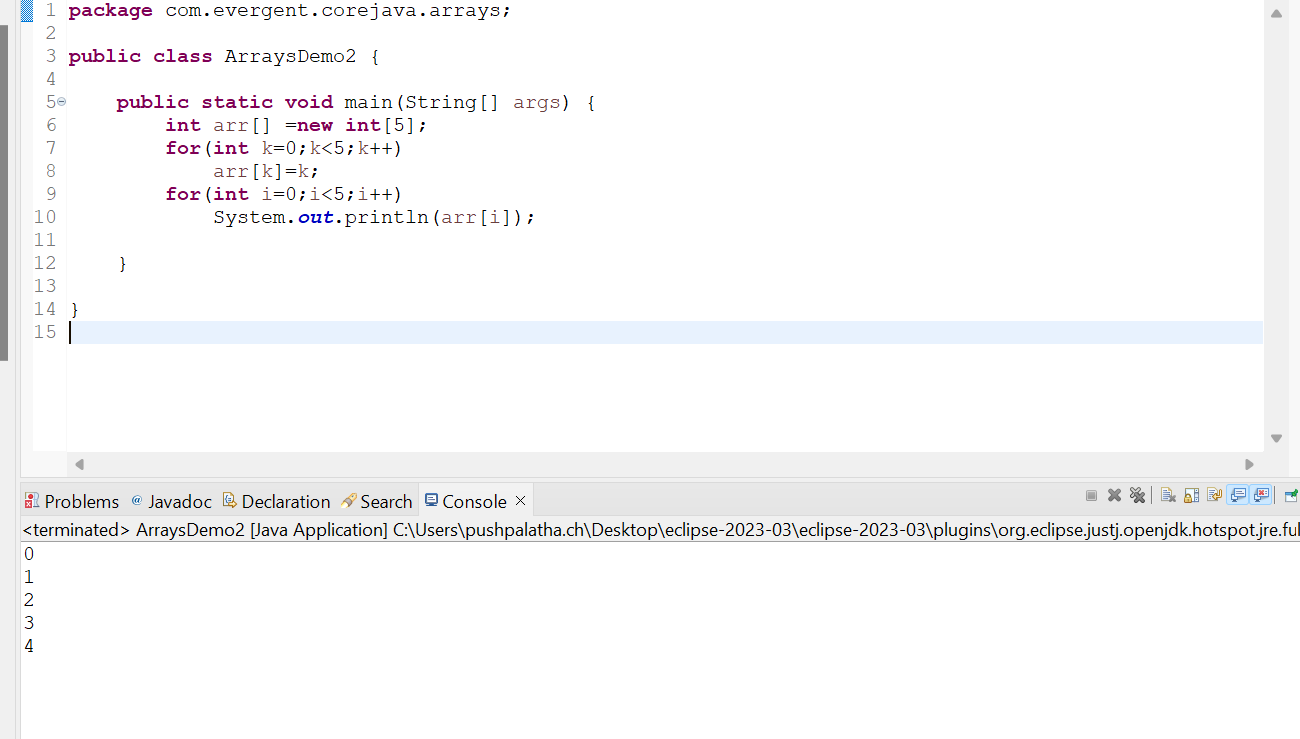
Array1(initializing,retriving of data in single dimensional array)



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



Array3



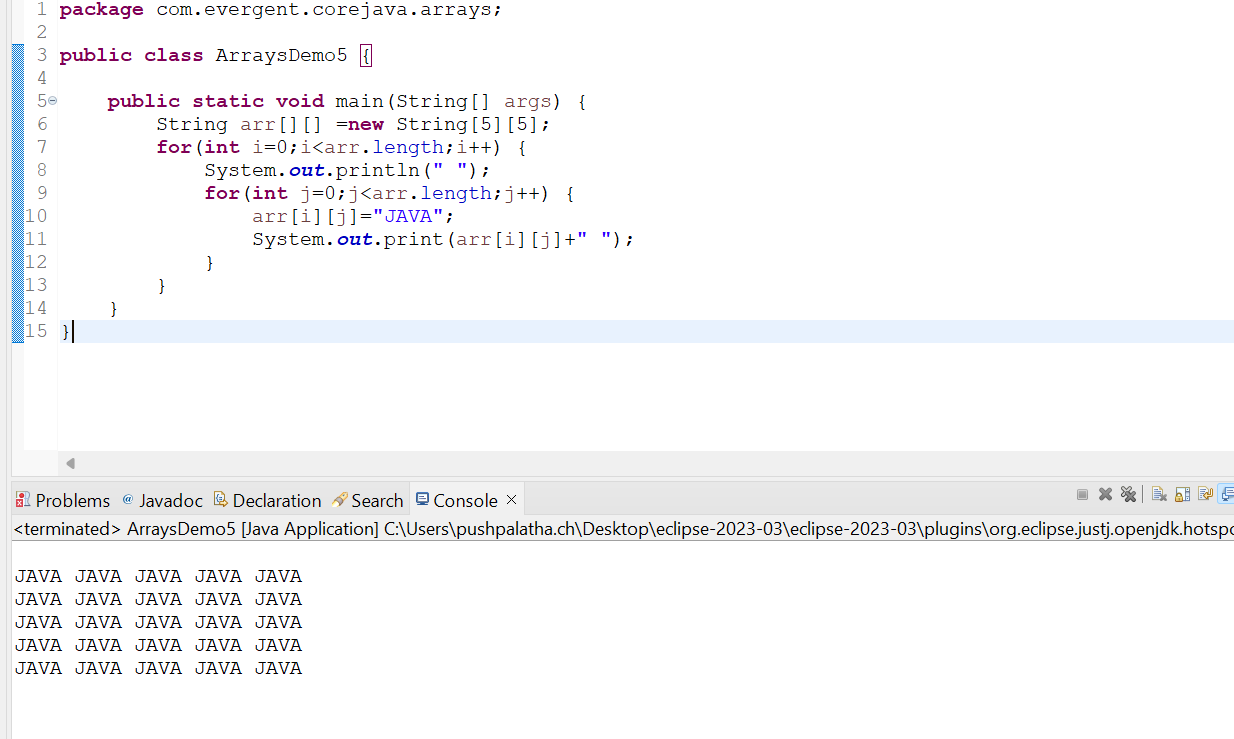
Array4



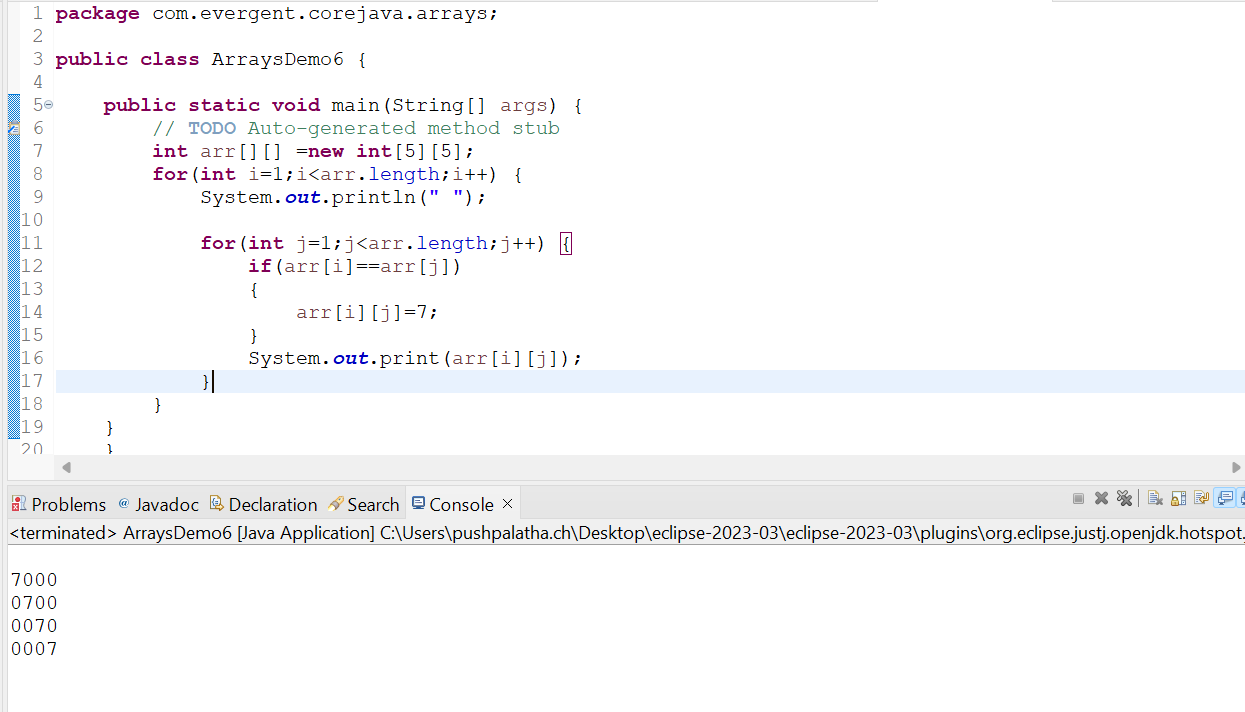
Array5



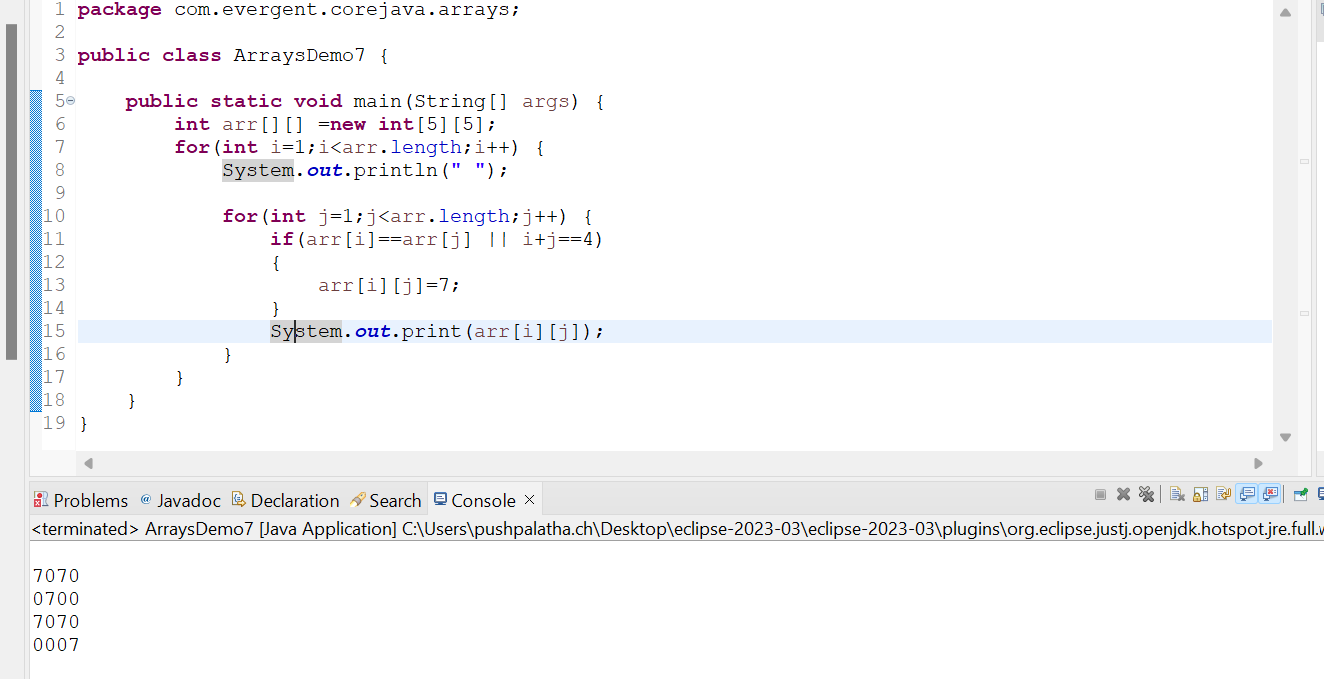
Array 6(Multidimensional Array)



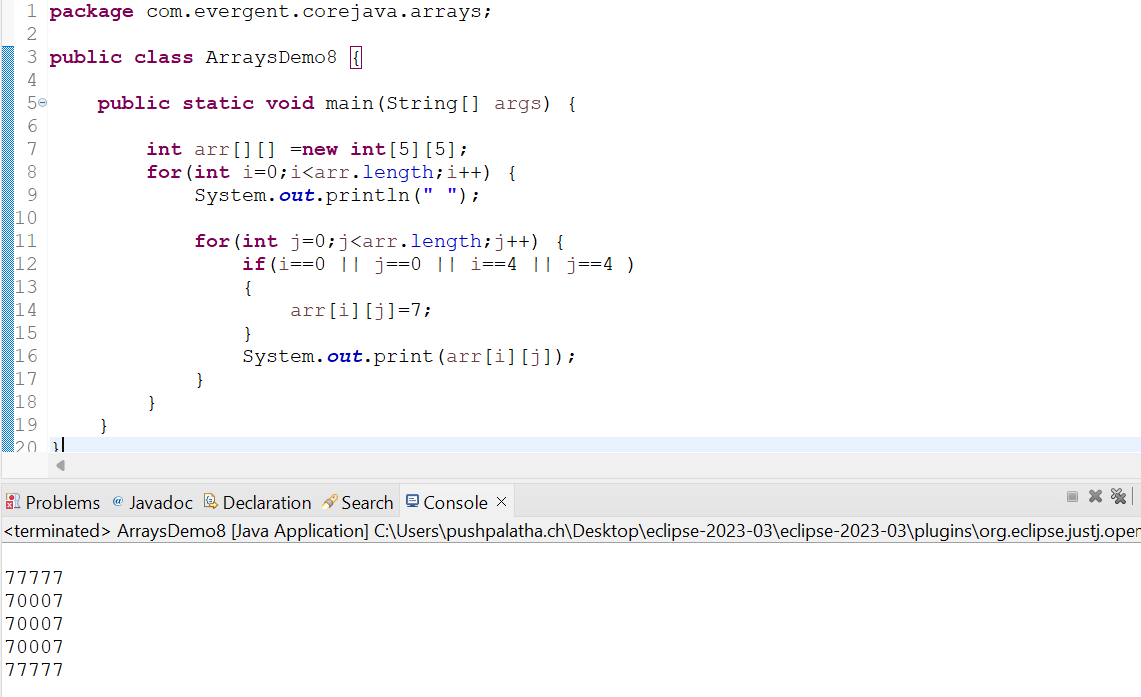
Array 7(Multi dimensional array)



Array 8(Multi dimensional Array)



Array 9(Multi dimensional Array)

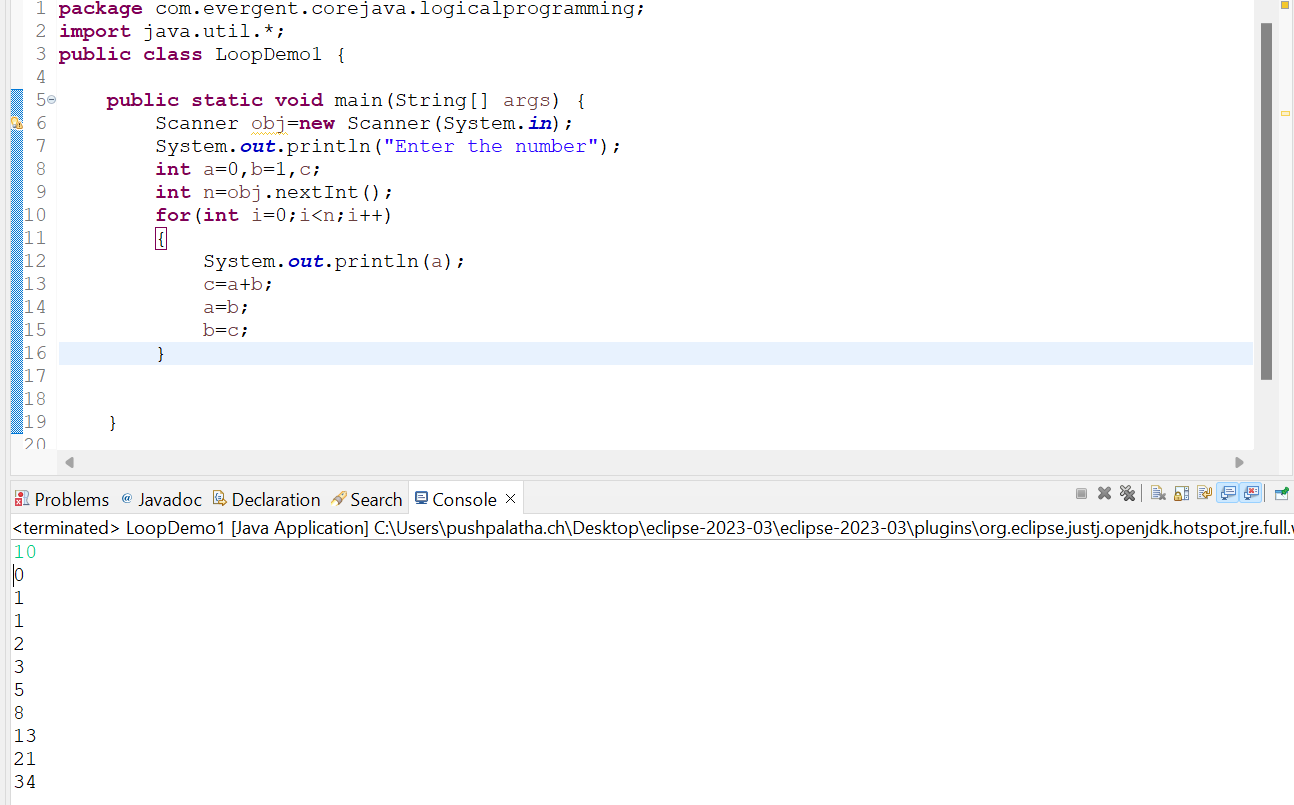


1. Logical programming

Factorial program



Fibonacci series program

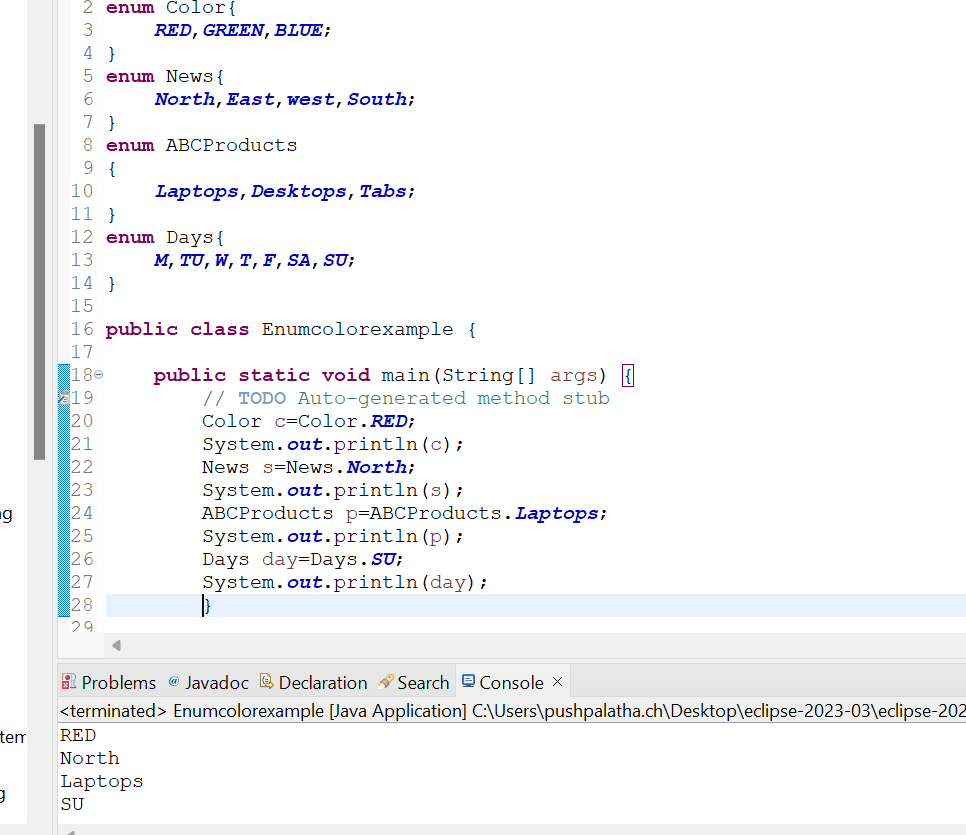


1. Switch case

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.

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

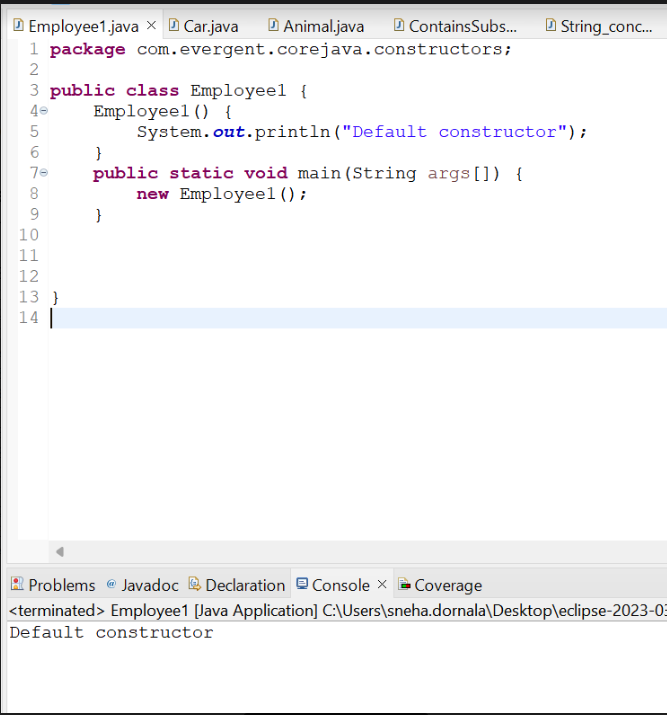
**Date 08/08/2024**

**Day-4**

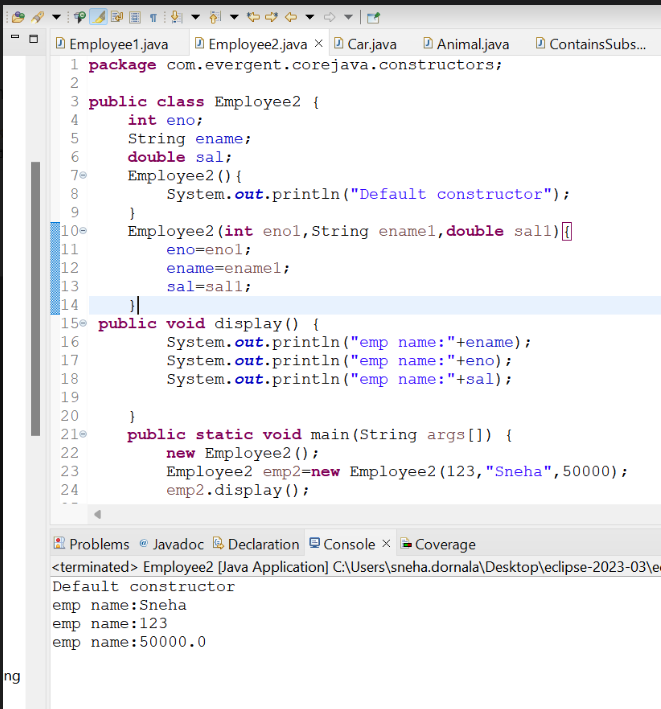
**Constructors**

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

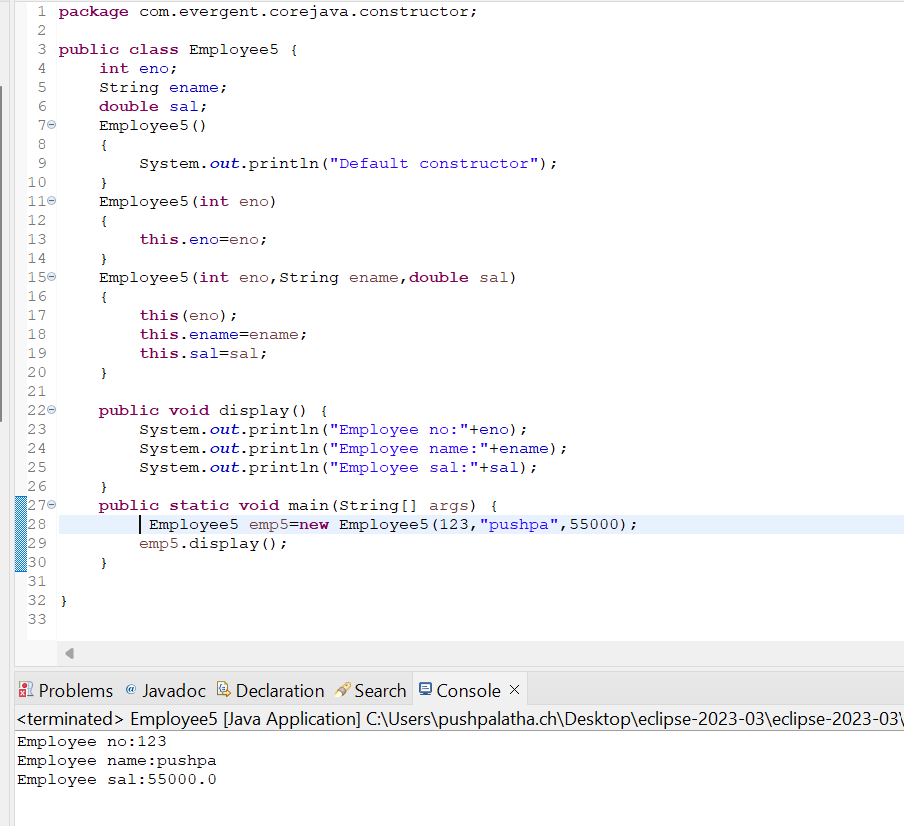
Default constructor



Parametrized constructor

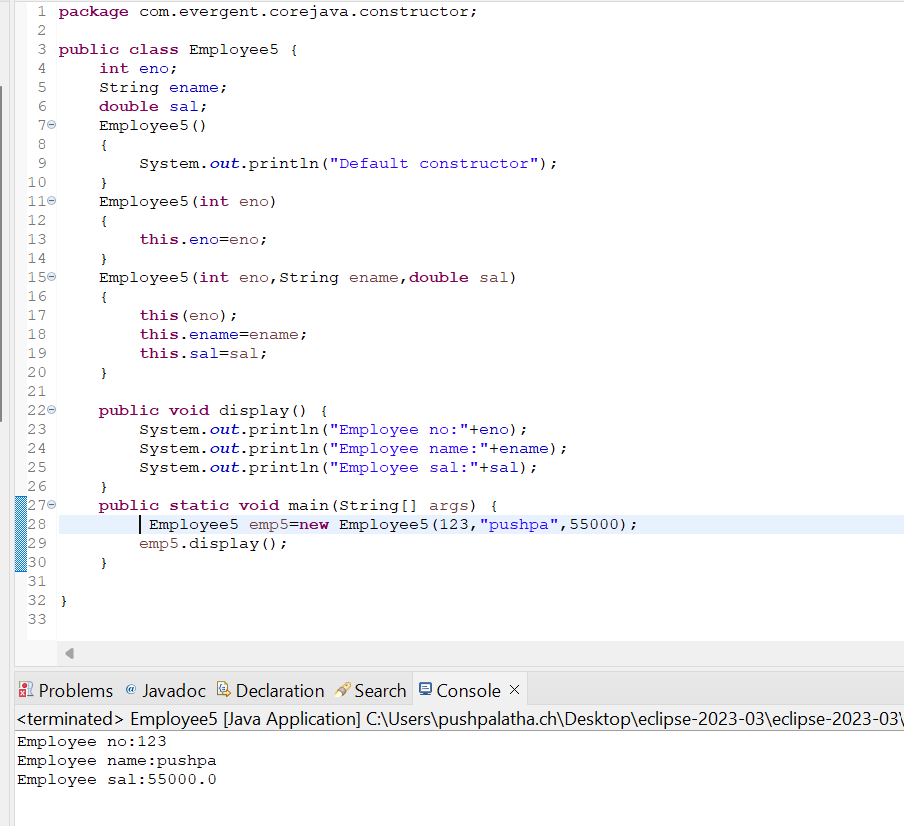


1. We can access constructors while creation of objects.
2. Constructors are mainly for intializing the objects.
3. Constructors does not any return type not even void also .If we declare as a void it will consider as a method not as constructor.

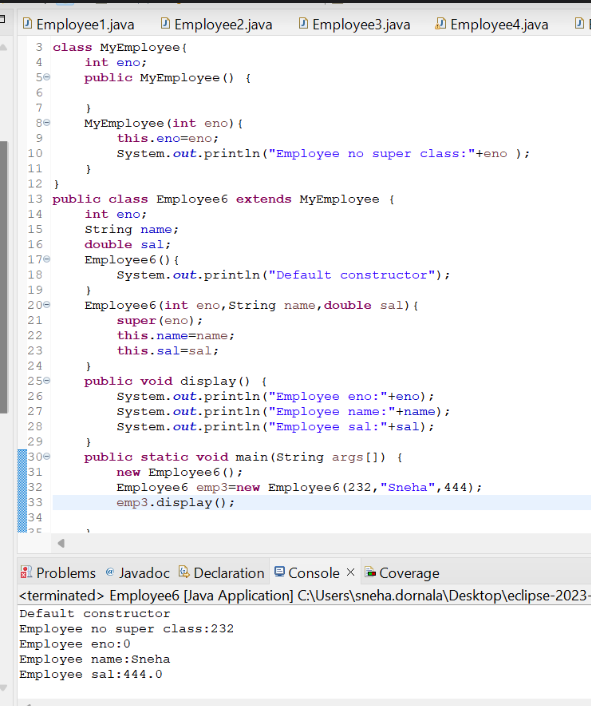


1. Every Class Contains atleast one default constructor Either we will create or JVM will generate at the time of compilation.
2. this and super Keywords.

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)

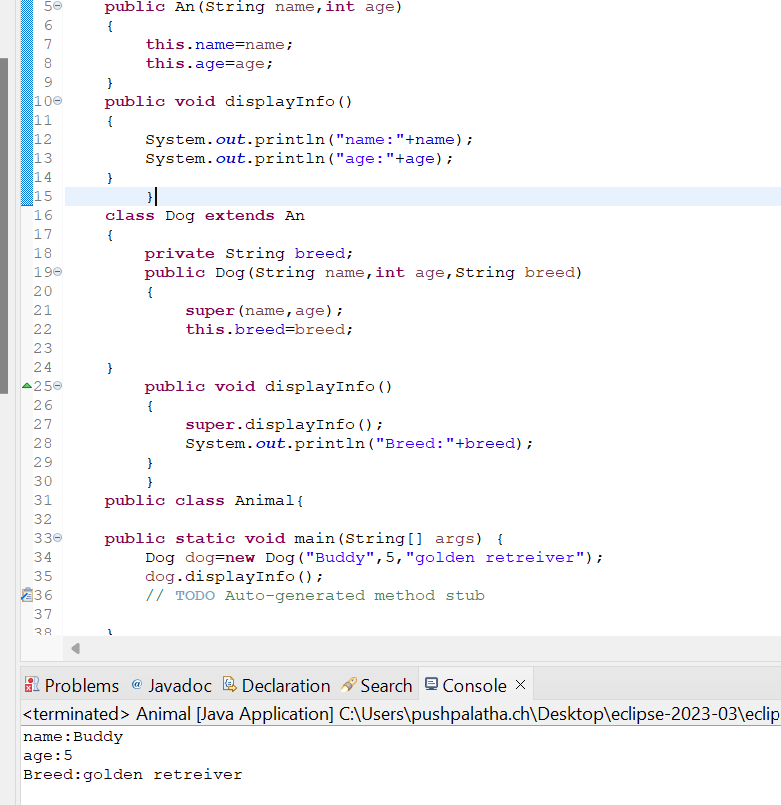


1. Object copy constructor.



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.

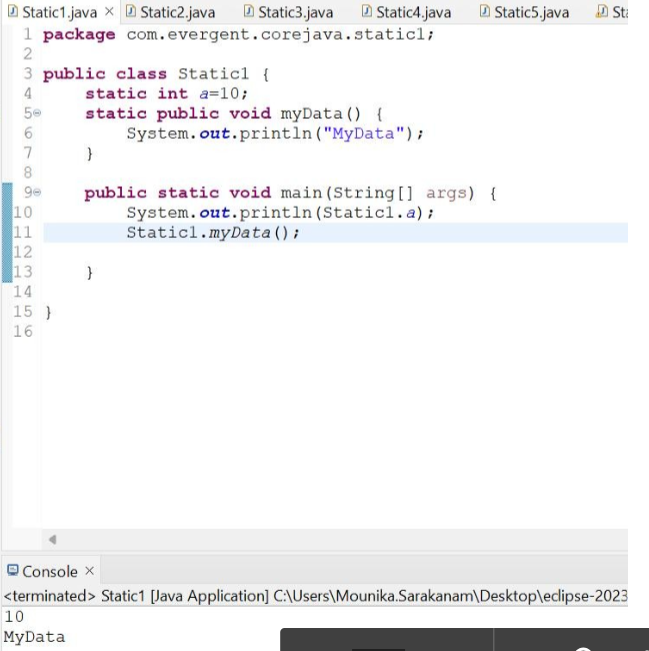


ConstructorOverloaded

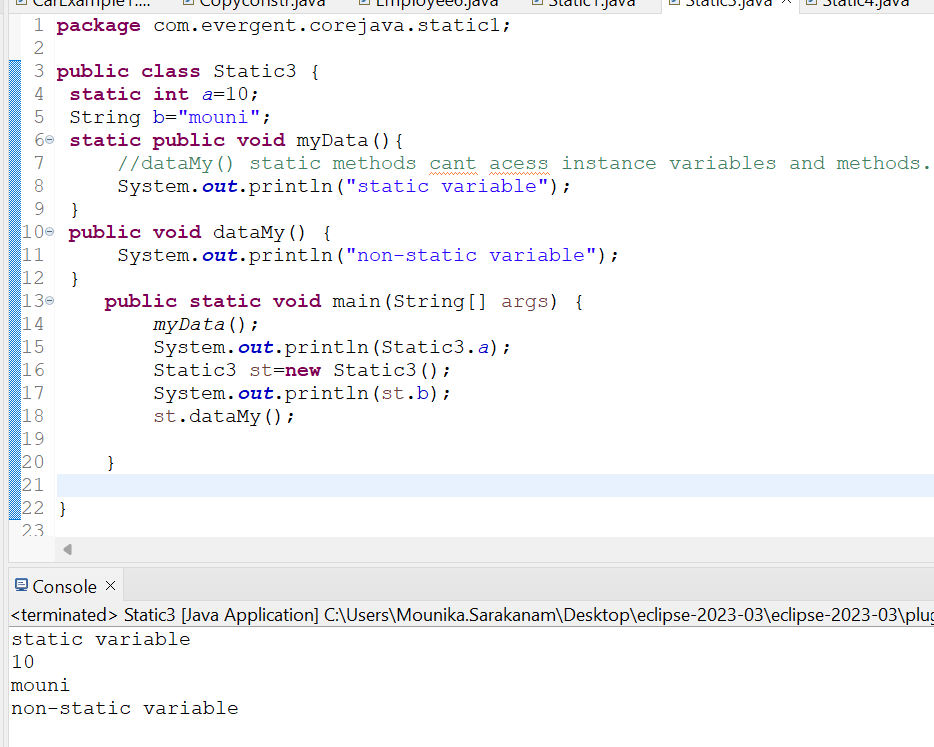
**Static keyword**

Static is a variable

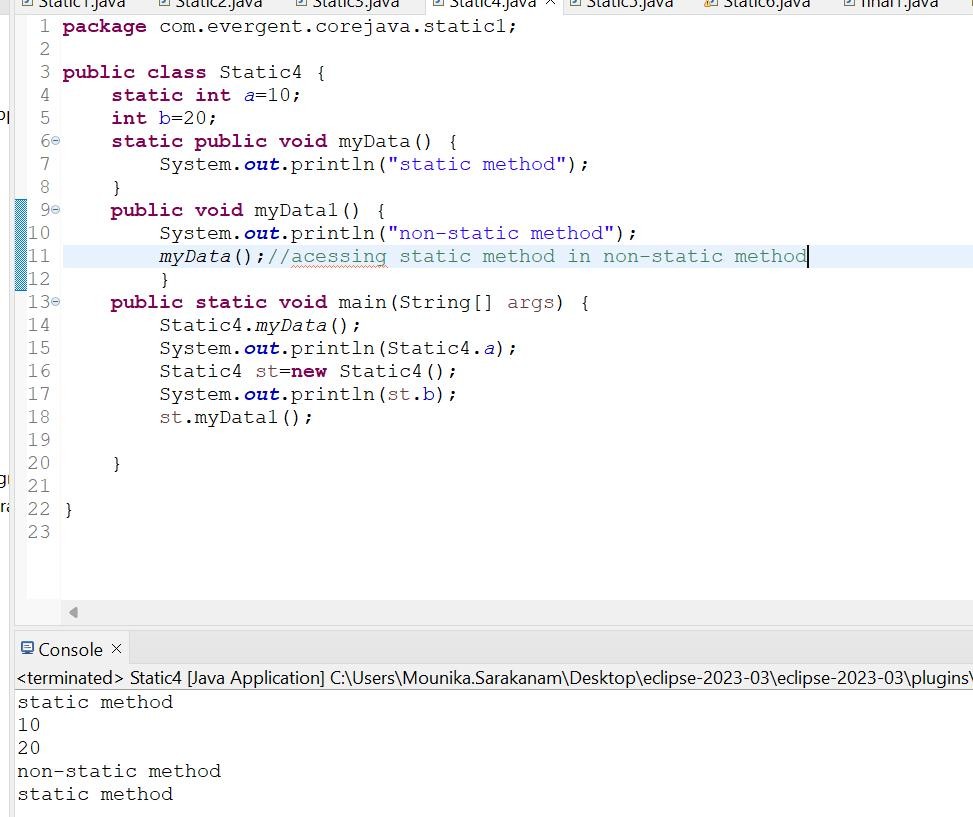
We can declare static as variables,methods



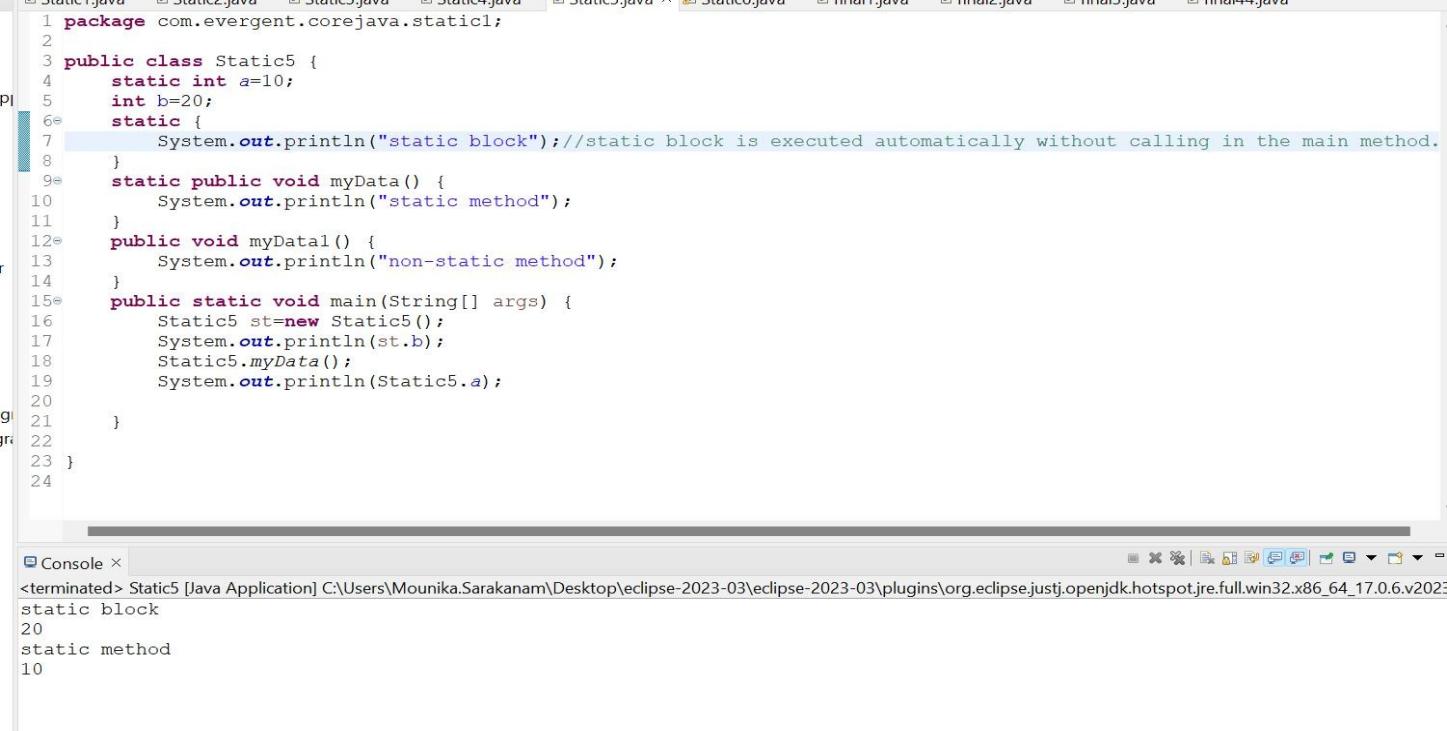
Inside static method we cannot access non static variables or methods.



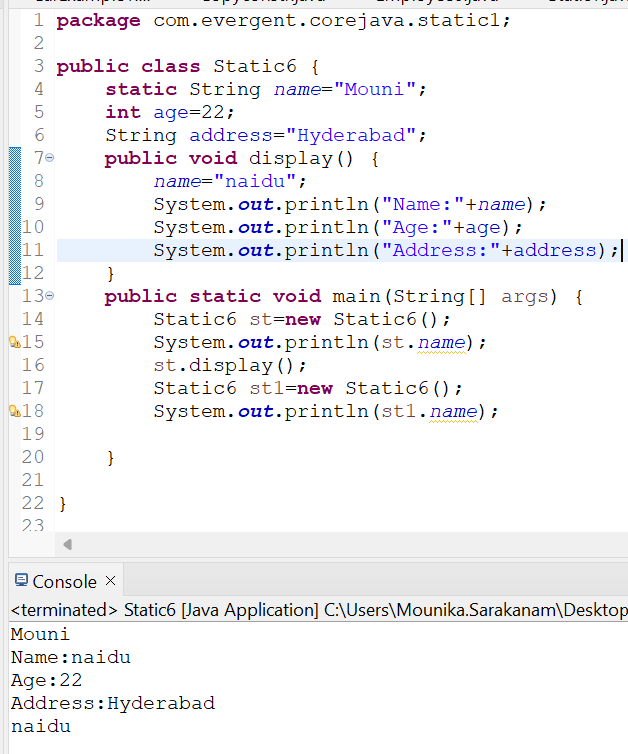
Inside non static method we can use static methods or variables.



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

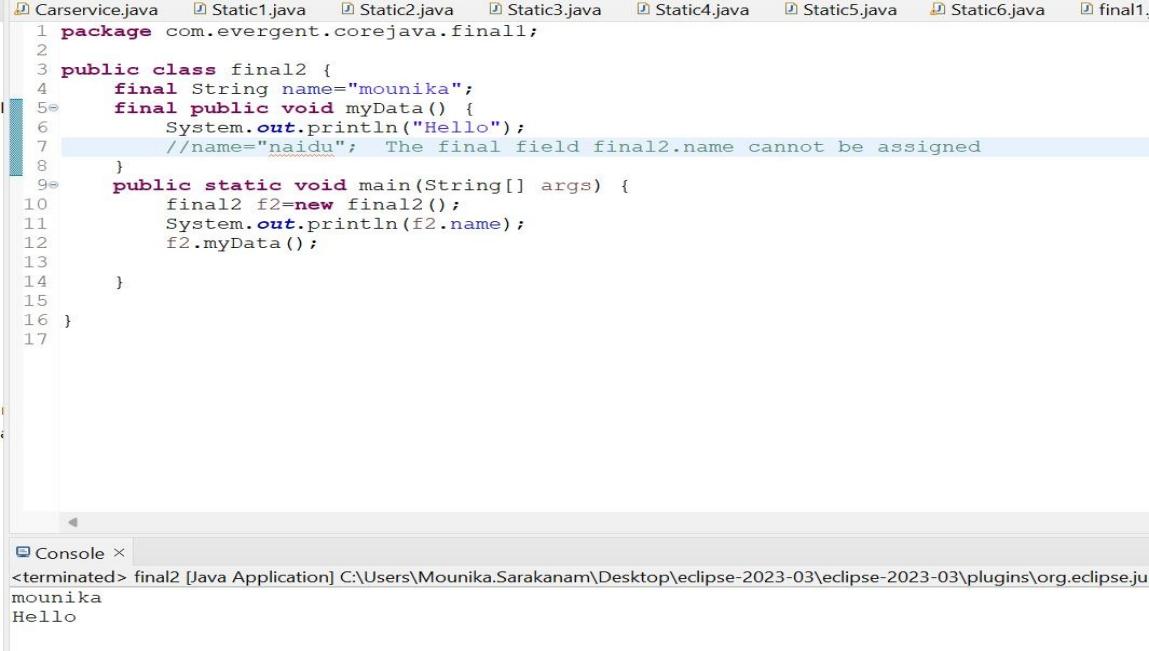


# final

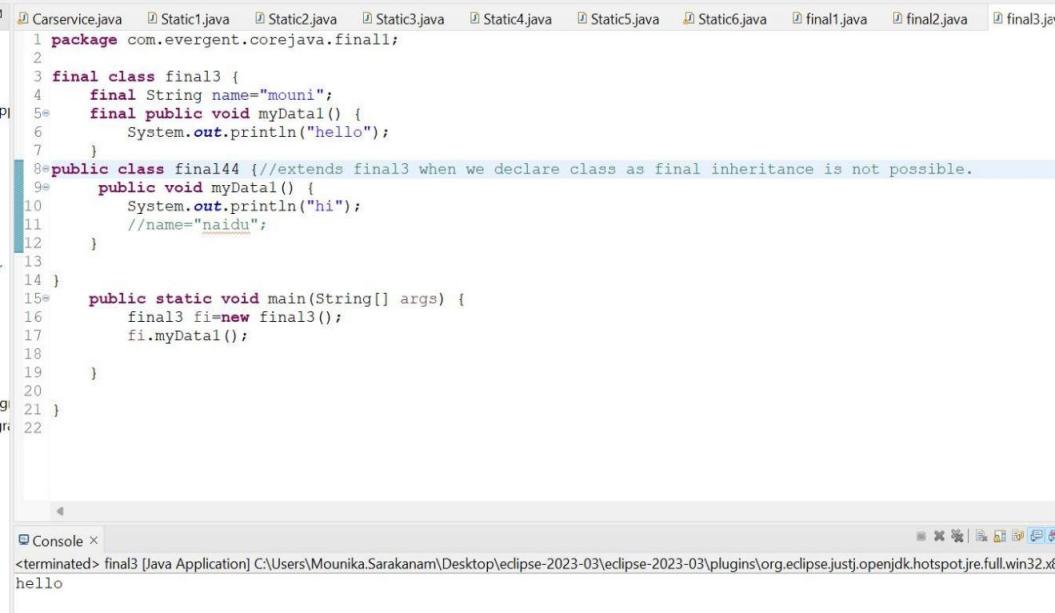
final is a keyword

final can be declared as variable,method,class

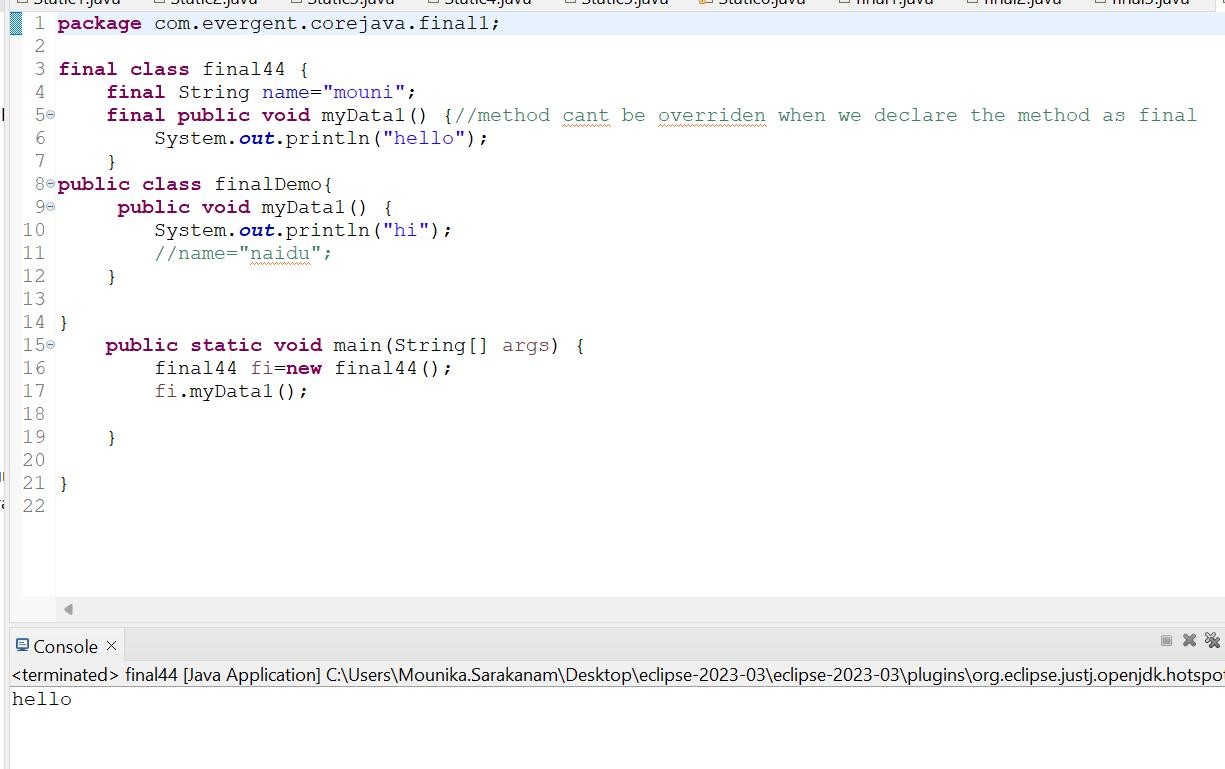
If variables are declared as final those are treated as constants cannot be modified after.



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.

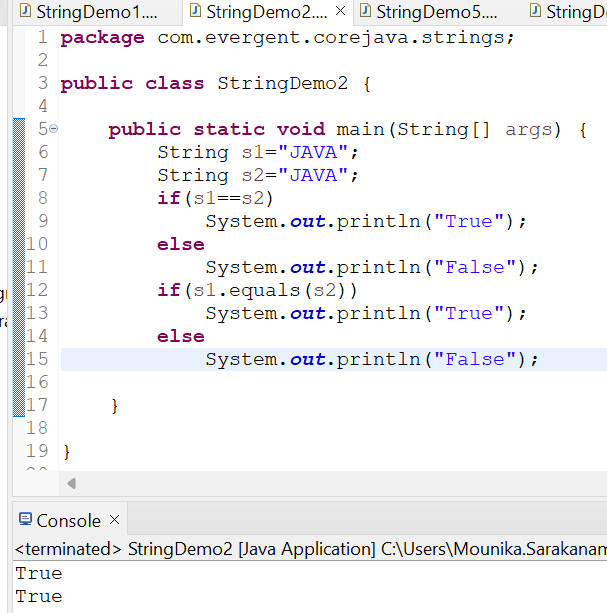
== method in String is used to check the Memory location.



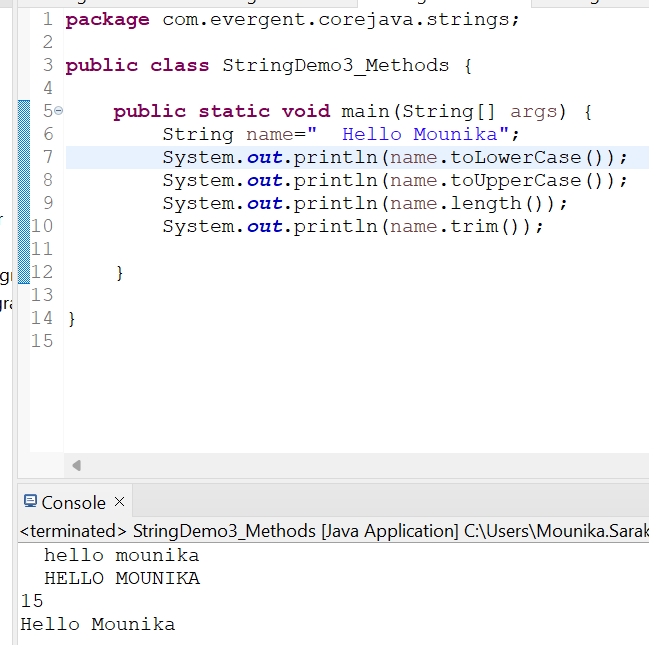
2)Creating String using literals

equals() in String used to check the content.

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



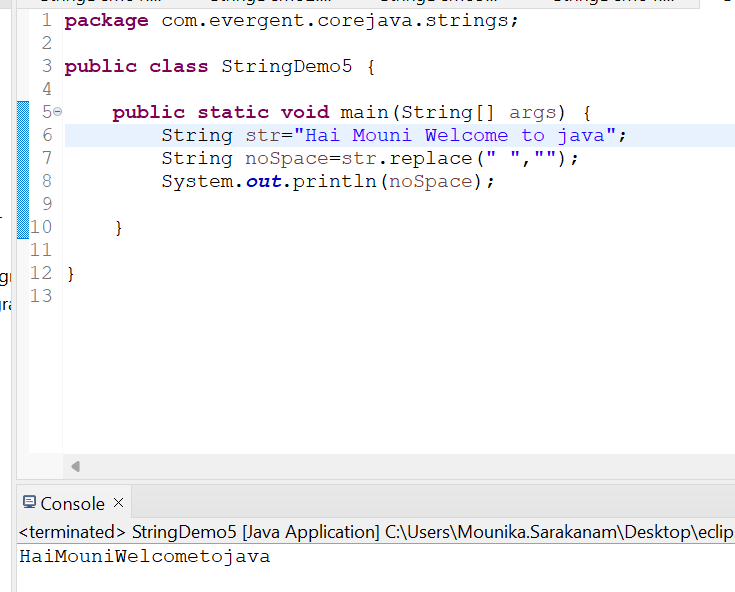
1. String methods (toLowerCase()-Changes the complete String into lowercase,
2. toUpperCase()-Converts the complete String into uppercase,
3. length()-To know the length of the String,
4. trim()-To remove the spaces before the String)



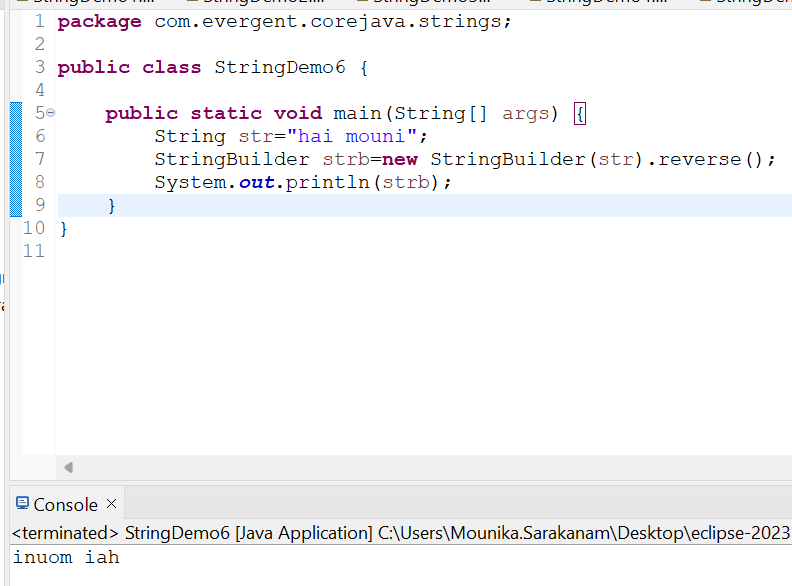
4)String method(contains()-to check whether the substring is present in the string)



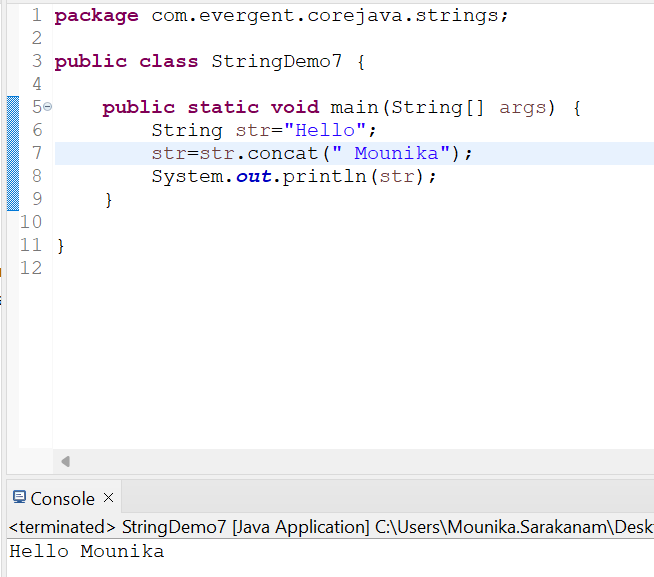
5)String method(replace()-to replace the values with other values)



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



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

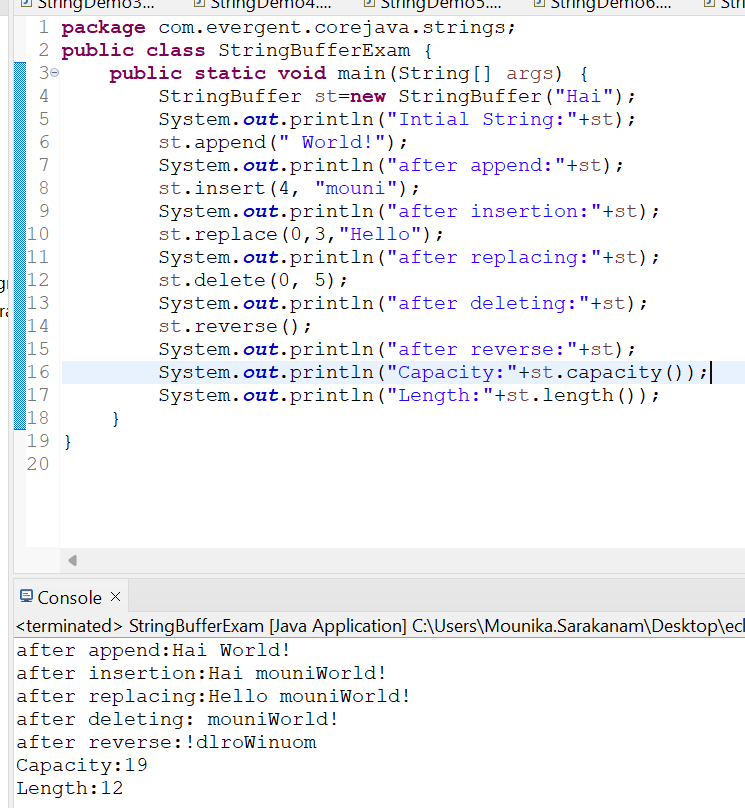


StringBuffer

* 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,

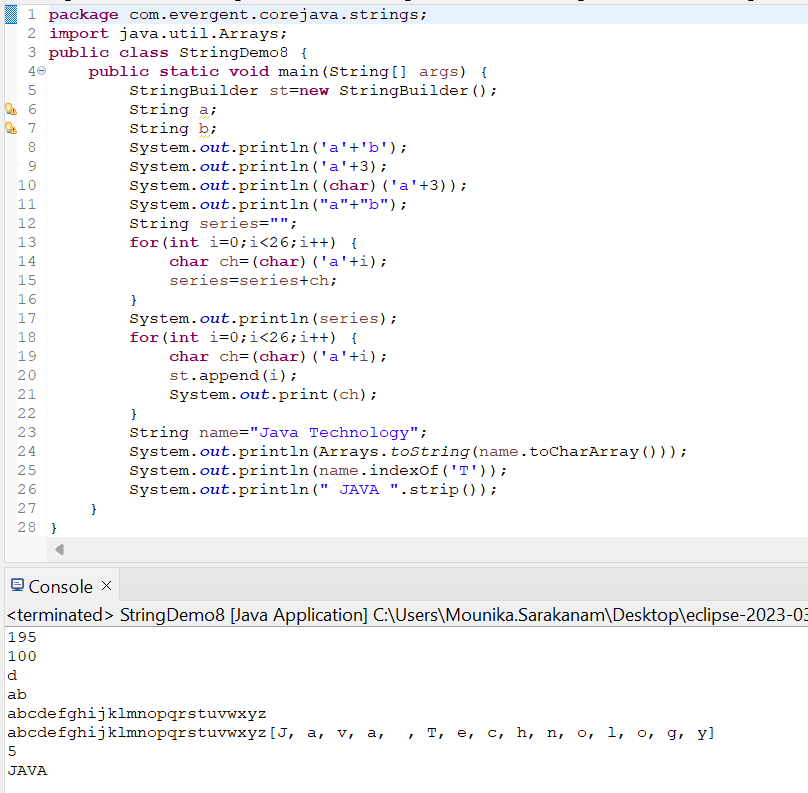
Insert-to insert the string into the present string based on the index value,reverse-to reverse the string ,replace-replace the string with other string based on starting index and ending index,delete-to delete the sub string or string from the existing string based on starting index and ending index.)



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.)

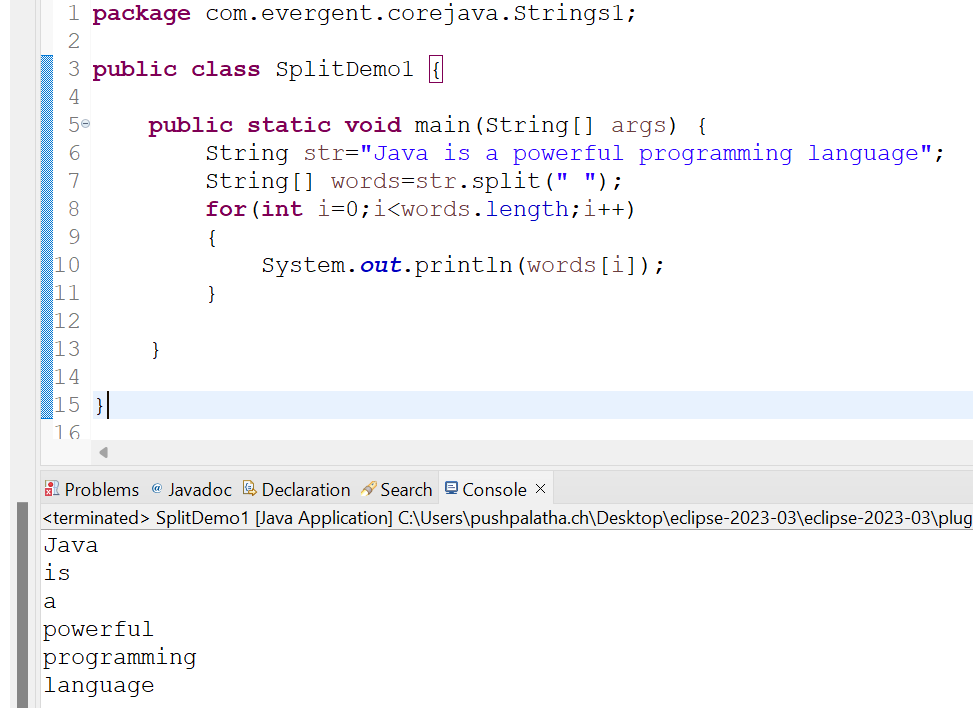


Date:13/08/24

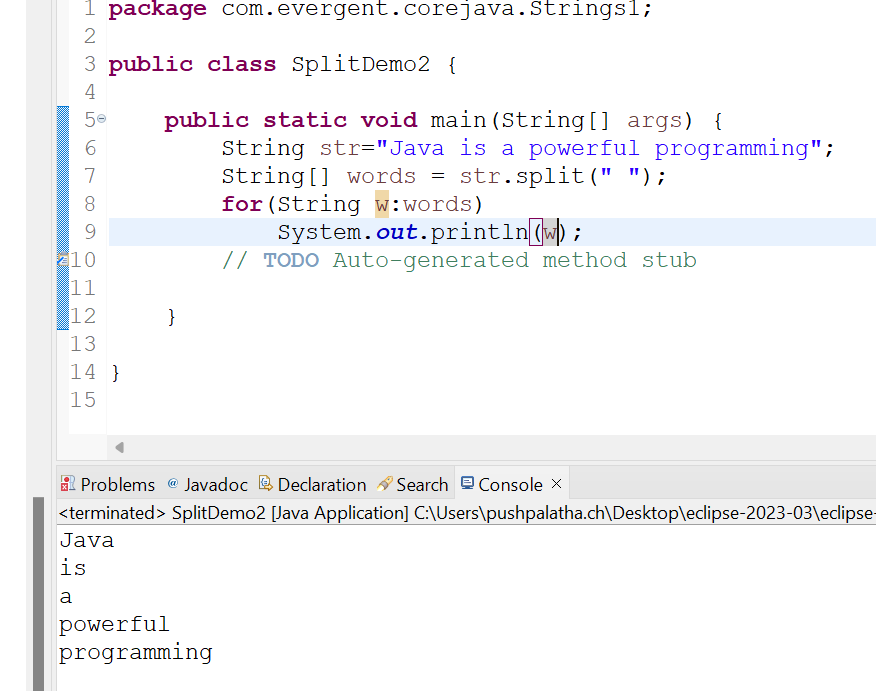
Day-07

INDEX

Split



**Using For Each loop**



Immutable class:

1. We can decalre class as final
2. The class is decalred as final so that it cannot be subclassed.

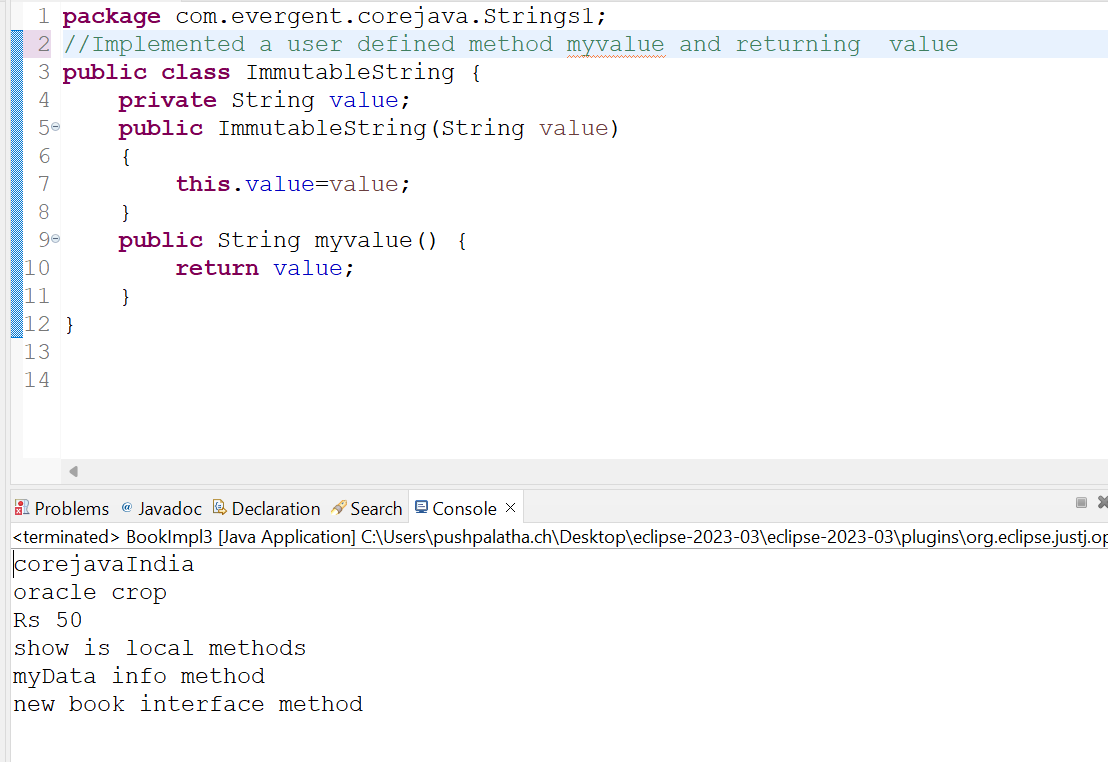
Private Final Field:

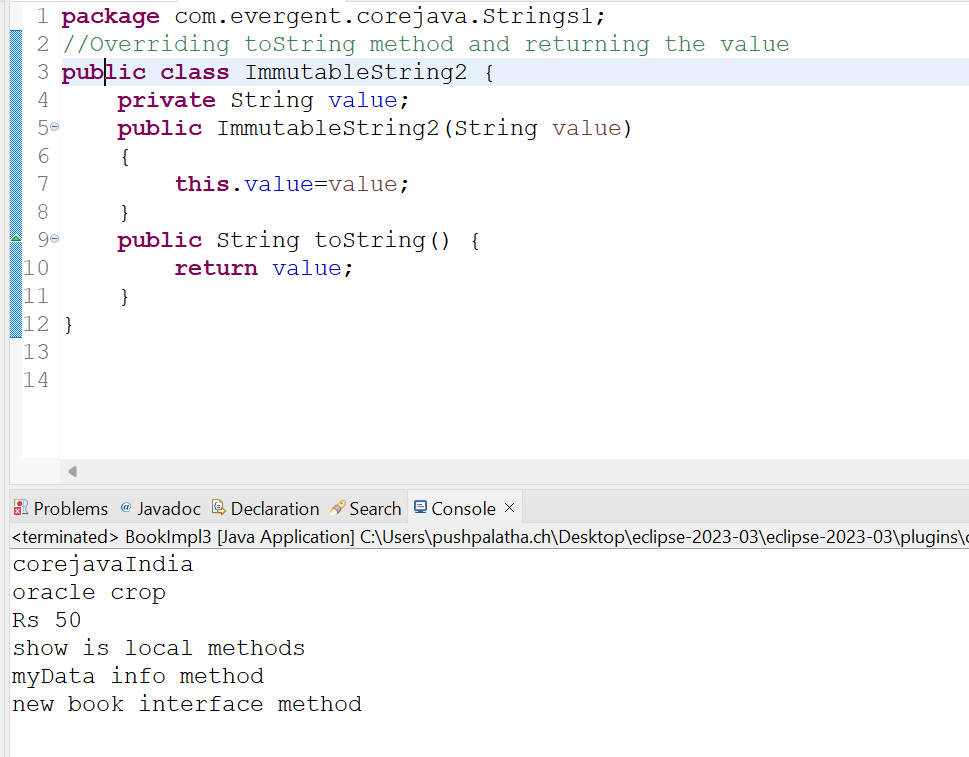
1. The fields name and age are private and final,constructor.
2. The constructor intilizes the final fields when a person object is created.

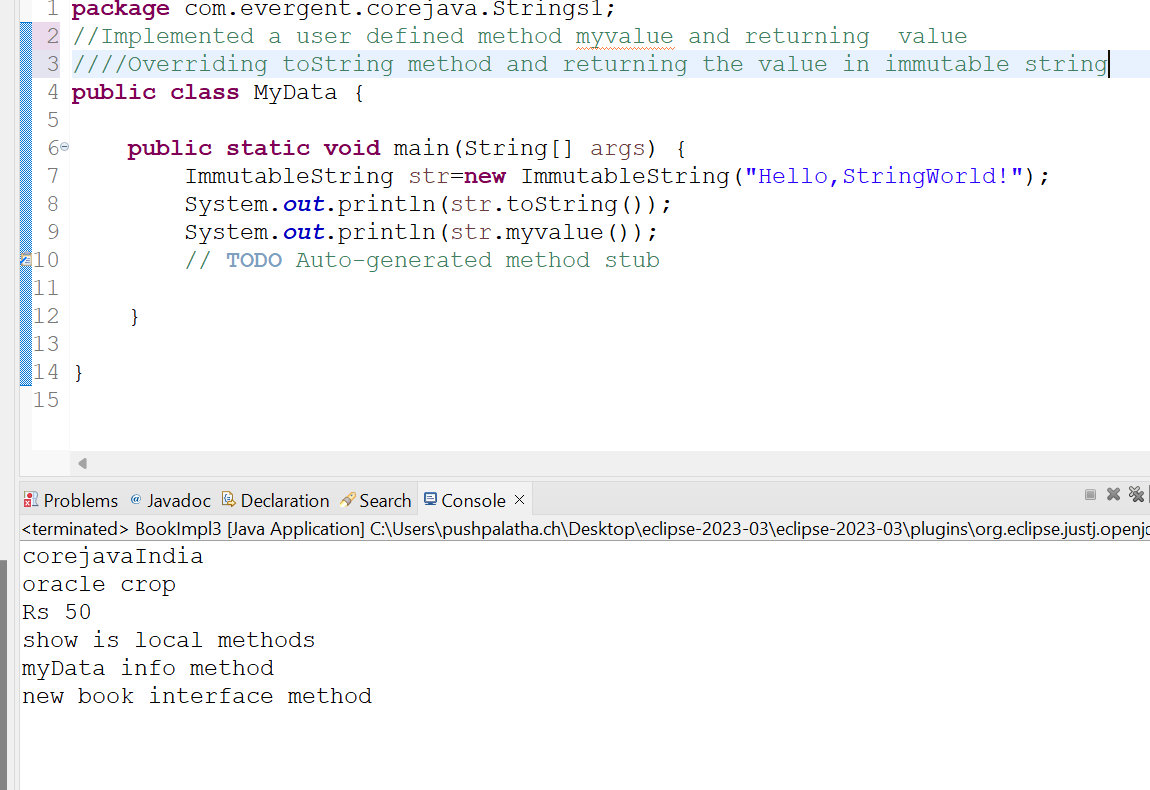






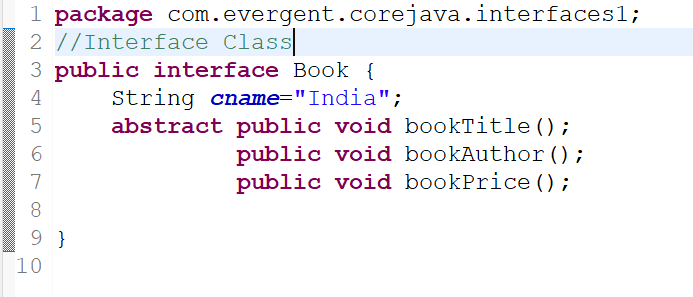




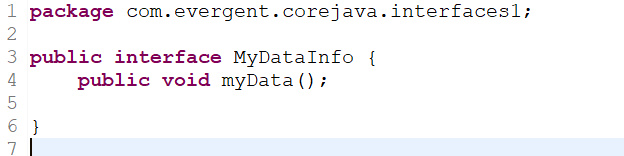


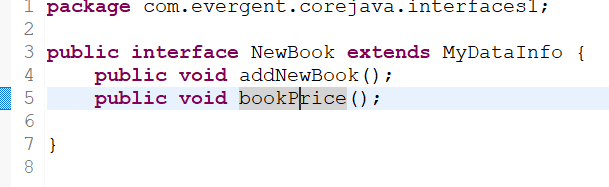
Interfaces:

1. Interface is 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 override all interface methods otherwise that class will be showing compiletime Error.
5. We cannot Create Object to interface but we can create reference to interface.
6. We can declare variables inside interface .All interface variables are by default public,static,final.
7. Java will support multiple Inheritance through Interface.
8. One class can implements interfaces.
9. One interface Extends other interfaces.

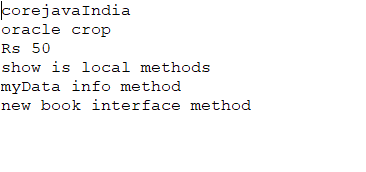


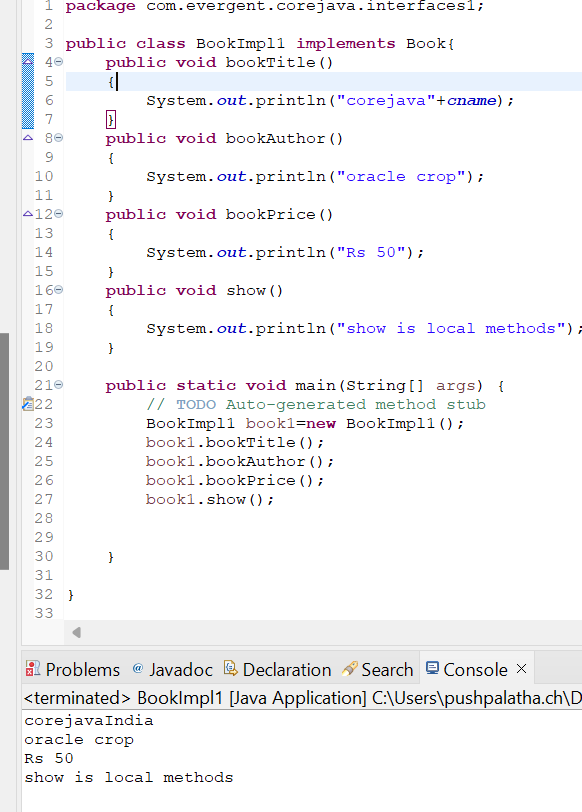
1. We cannot Create Object to interface but we can create reference to interface.
2. We can declare variables inside interface .All interface variables are by default public,static,final.
3. Java will support multiple inheritance through interface.
4. One class can implements interfaces.
5. One interface Extends other interfaces.

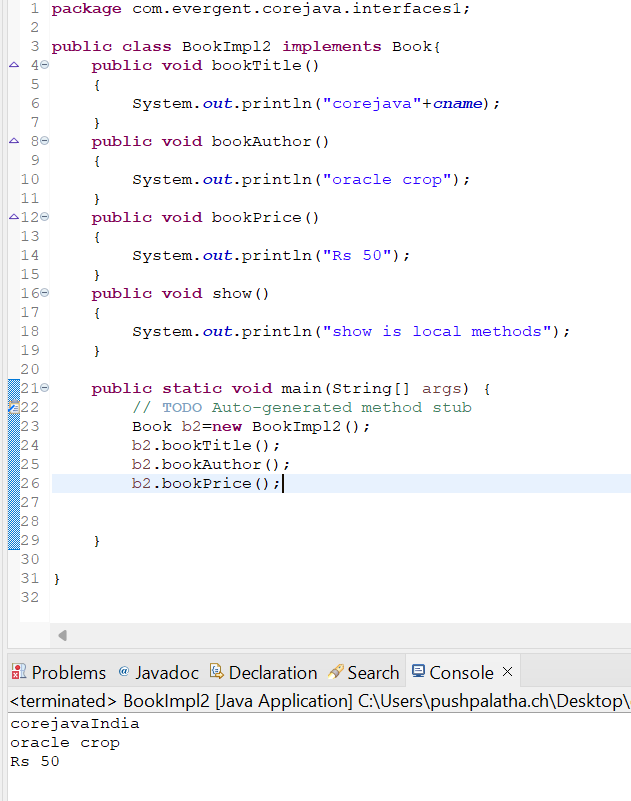












Day 8

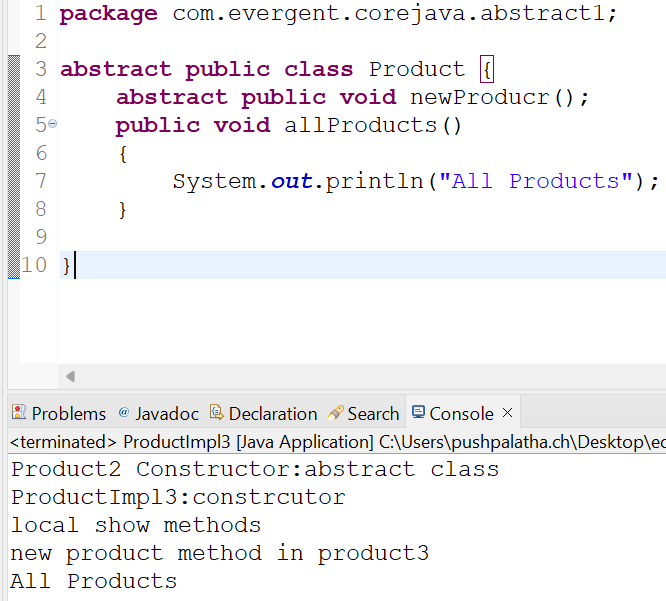
Abstract class:

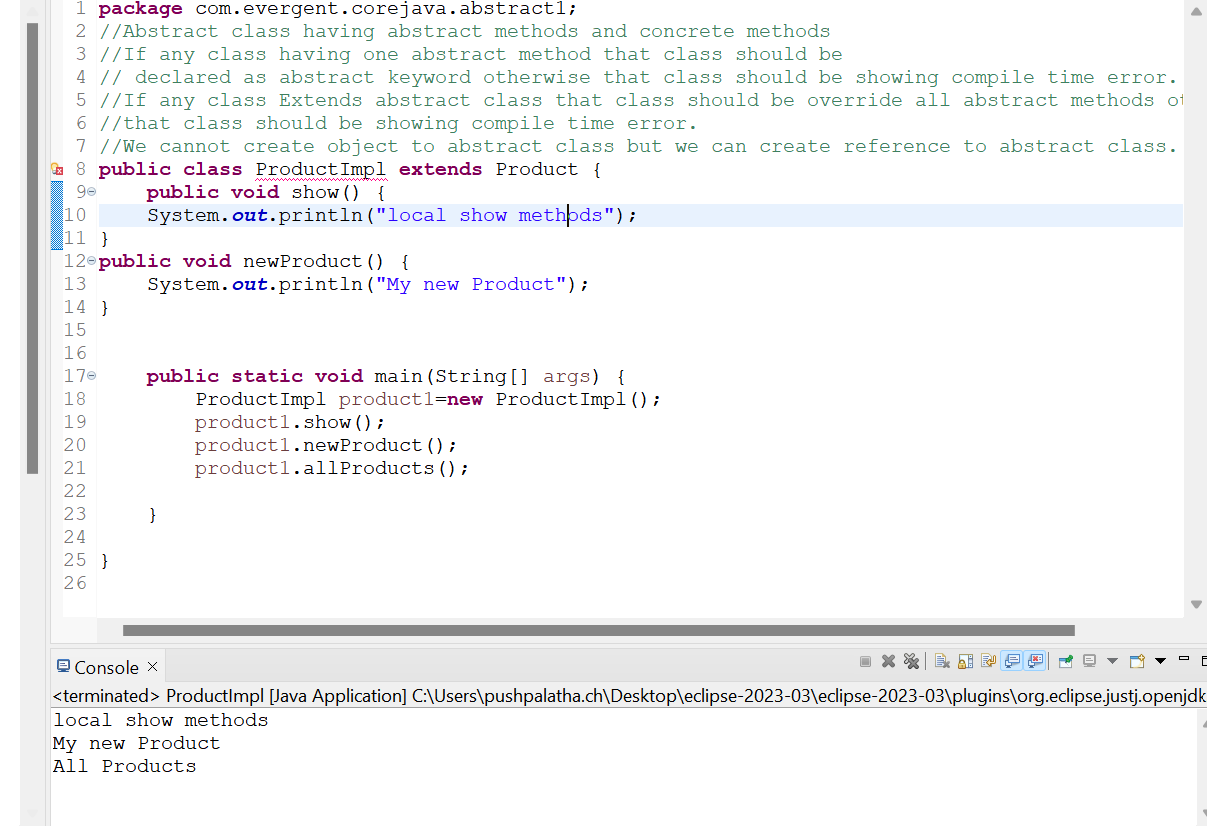
1. Abstract is a keyword.
2. Abstract class having abstract methods and concrete methods(Implementation methods).
3. If any class having one abstract method that class should be declared as a abstract keyword otherwise that class will be showing compile time error.
4. If any class Extends abstract class that class should be override all abstract methods.otherwise that class will be showing compile time error.
5. We cannot create object to abstract class but we can create reference to abstract class.

6.We can create constructor to abstract class.

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

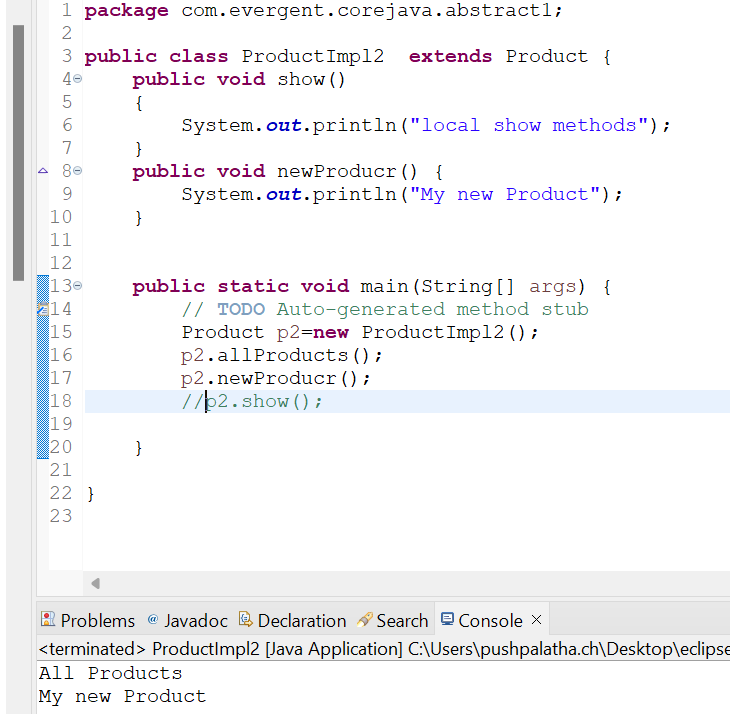
**Program 1**





**Program2**

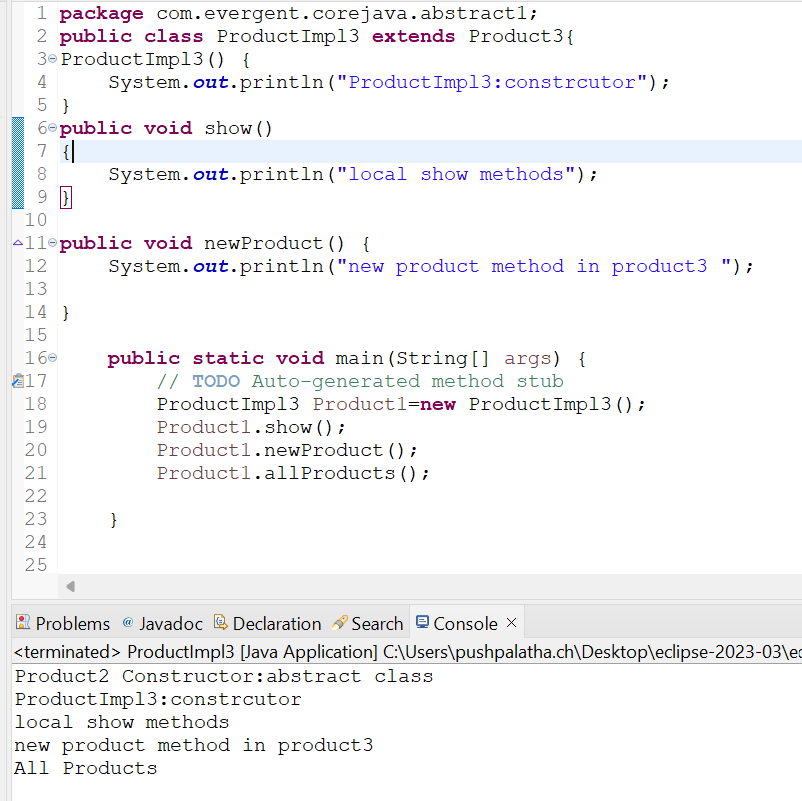
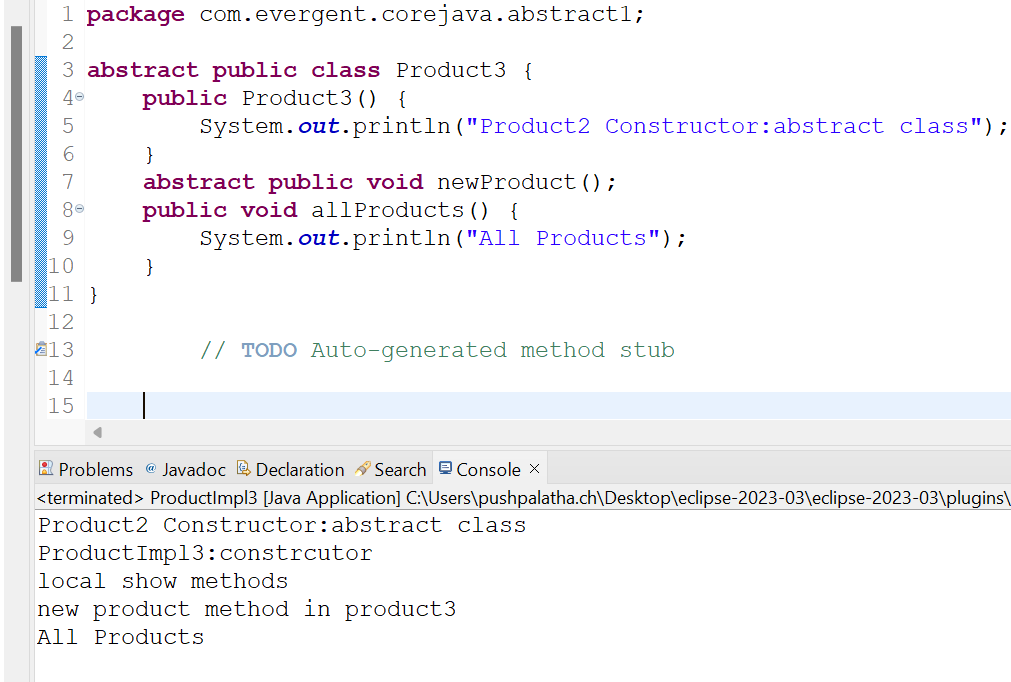
**Either we can declare as abstract while extend or override but if we take abstract no object is created.**



**Program 3**

**We can create constructor to abstract class.**

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

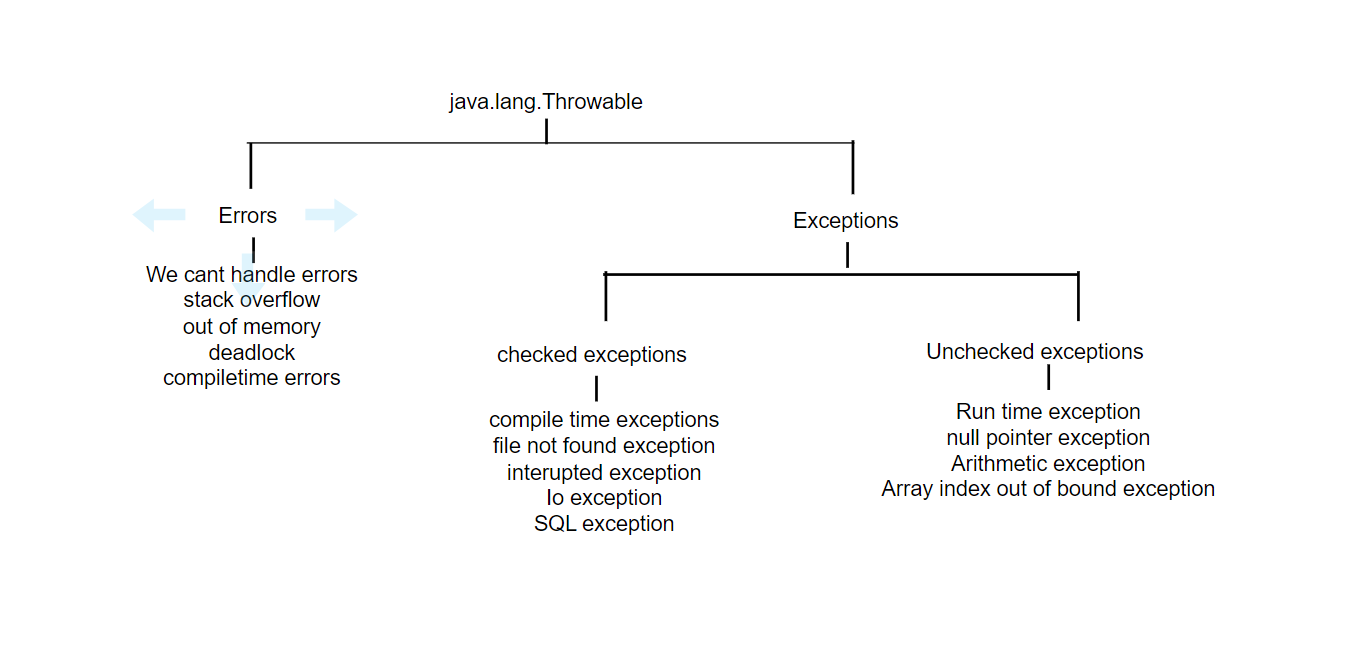


Day 10:

Exception Handling

1. Exception Handling is mechanism
2. Exceptions are inbuilt Mechanism of Java.
3. All Exceptions are executed while abnormal conditions only.
4. Normal flow it won’t execute any exceptions.
5. Once any exceptions are occurring in java then remain lines of code in unreachable.
6. Java.lang.Throwable is super class for execution and errors.
7. There are 2 types of exceptions in java are
8. Checked Exceptions
9. Unchecked Exceptions.
10. All checked Exceptions are compile time exceptions.
11. All unchecked Exceptions are Runtime Exceptions.
12. There are 5 keywords in exception handling
13. Try
14. Catch()
15. Finally
16. throws
17. throw
18. try is for business logic
19. Catch is for handling exceptions
20. Finally is a block if exceptions is occur or not finally block will be executed
21. Throws an exception will be executed method by method
22. Throw is for runtime exceptions and will call predefined exception classes
23. Try followed by either catch or finally block
24. We should follow exceptions hierarchical
25. We can create our own user defined exceptions
26. Our own extends exception or runtime exception
27. All exception classes are into java.lang package
28. There is two exceptions in class developers should be handled first exception then after second exception should be handled.

22.Developers cannot handle errors.



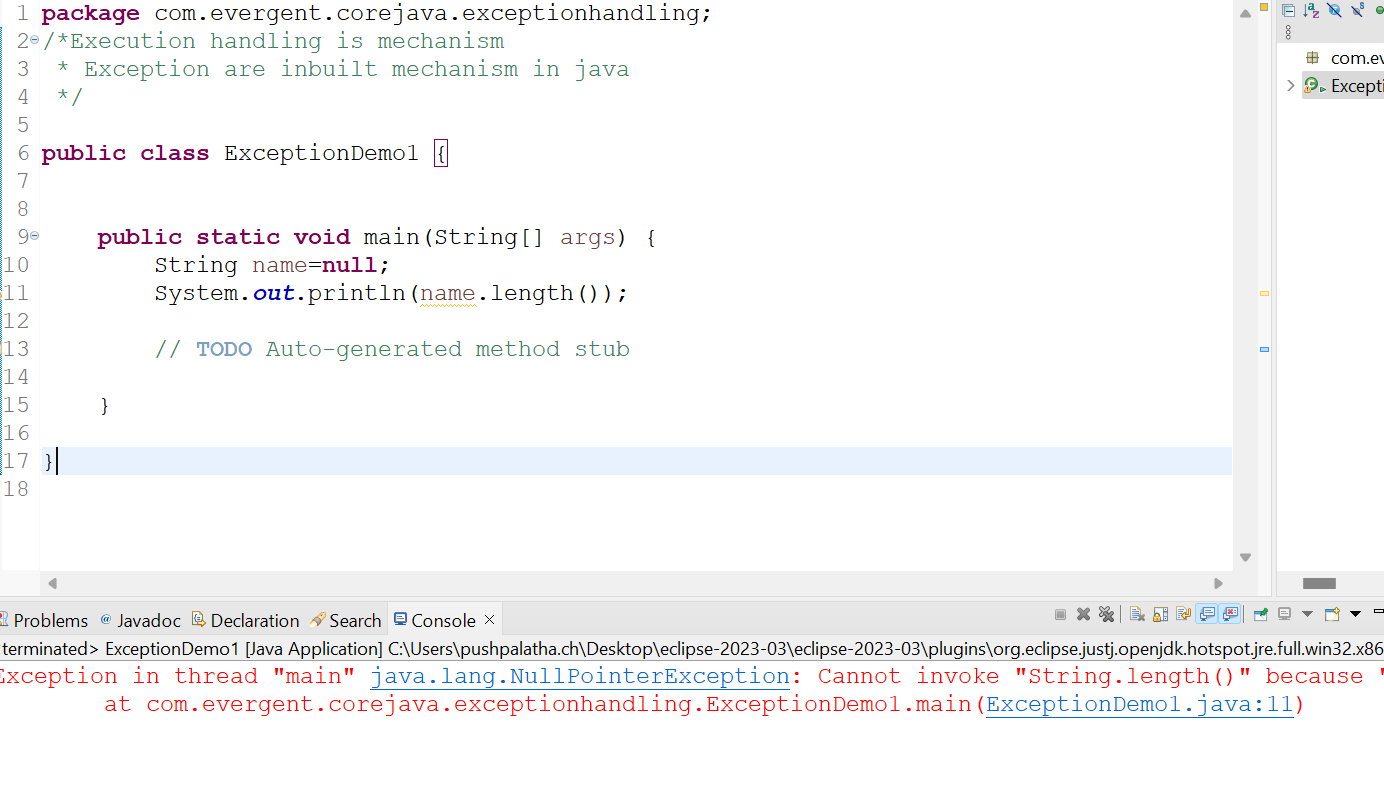
**Errors**

StackOverflowError:Error occurs when a method is called multiple times.

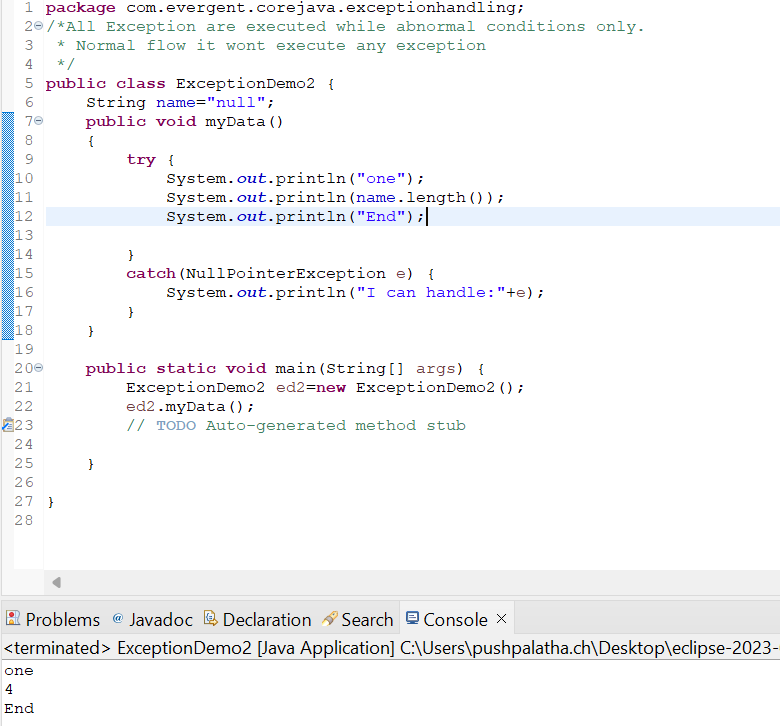
OutOfMemory:Error occurred when the size provided is larger than usual fixed size. Git

Used to track the application changes. Git consists of 4 stages:Working directory,staging,Local Repository,Remote repository

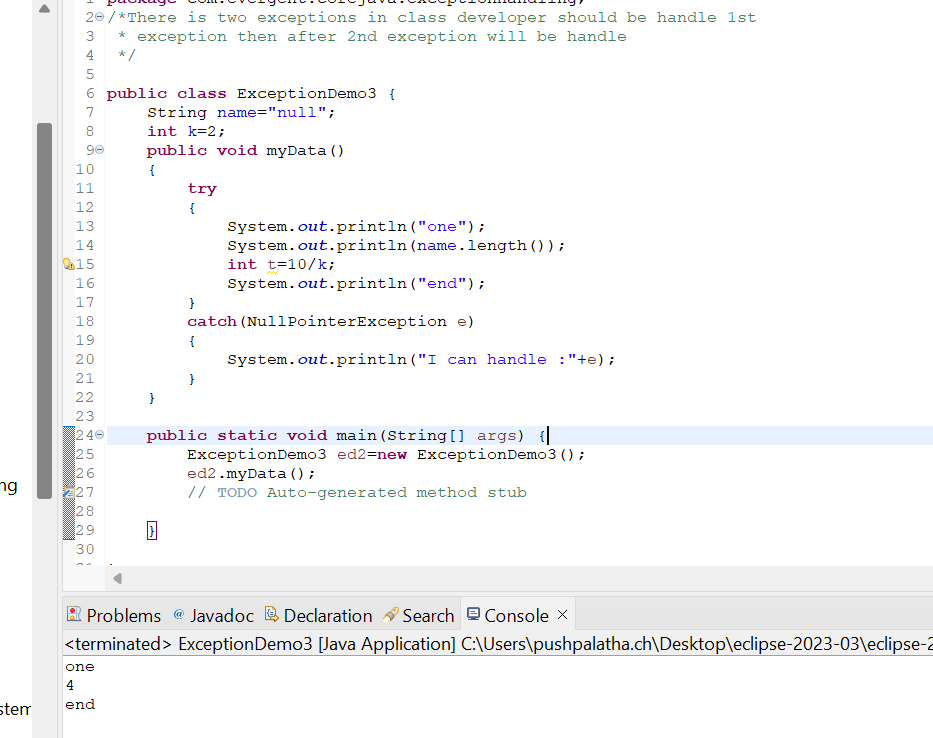
Program 1



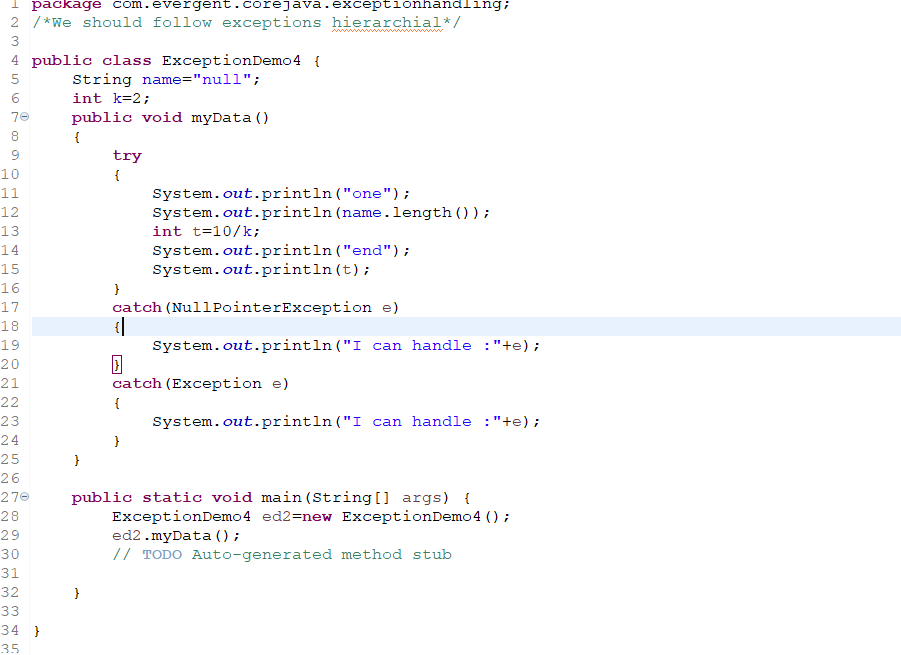
Program 2



Program 3

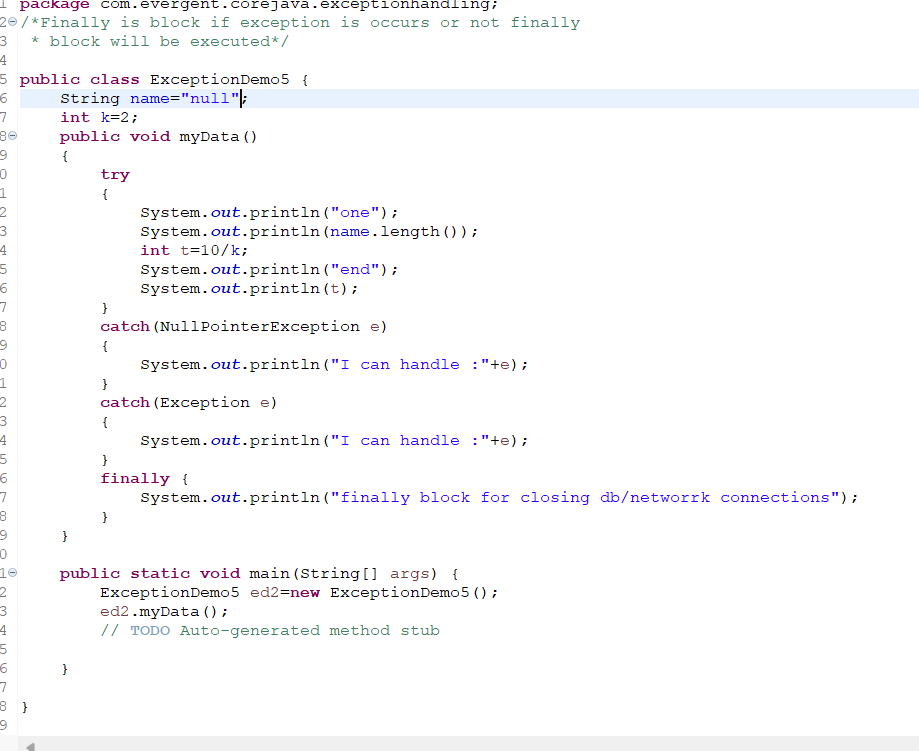


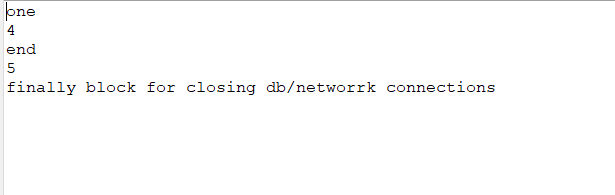
Program 4





Program 5

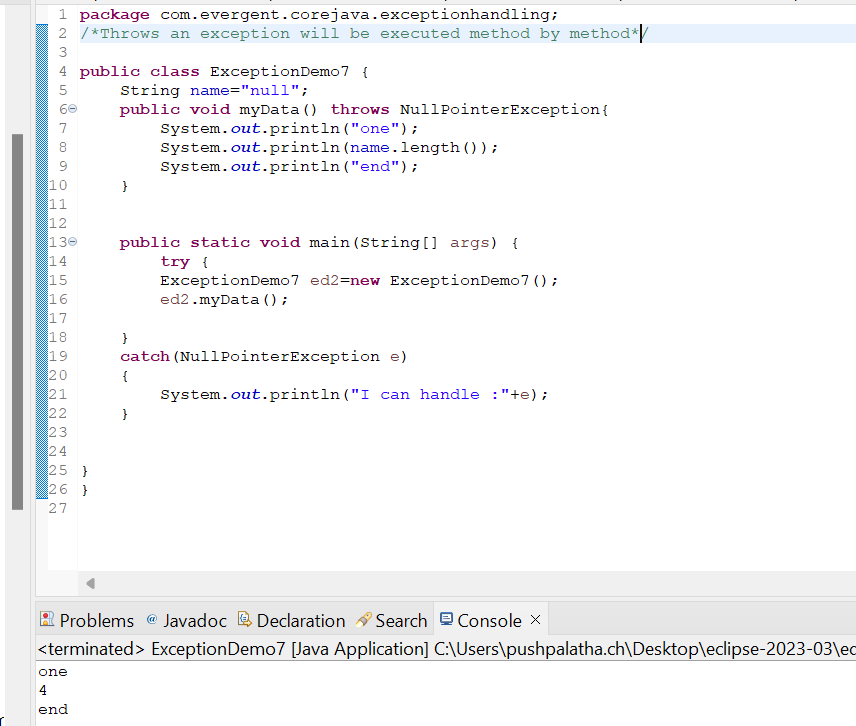




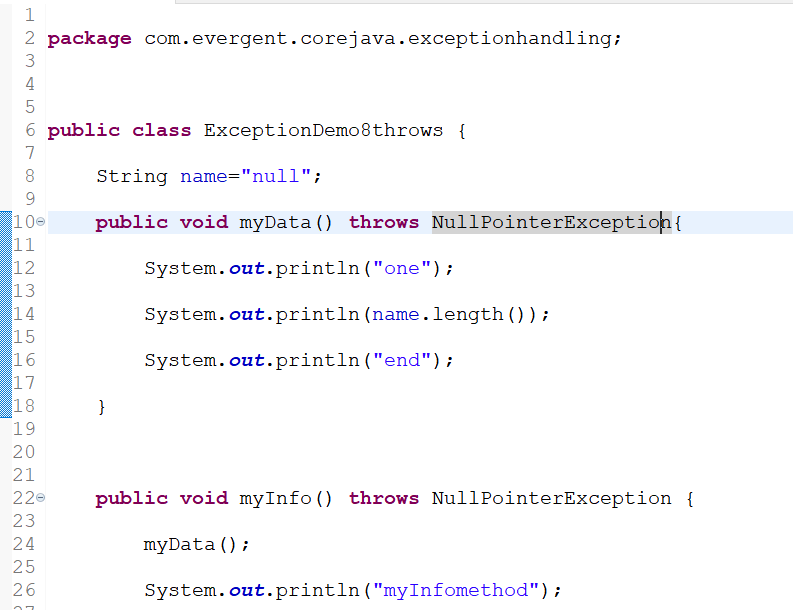
Program 6

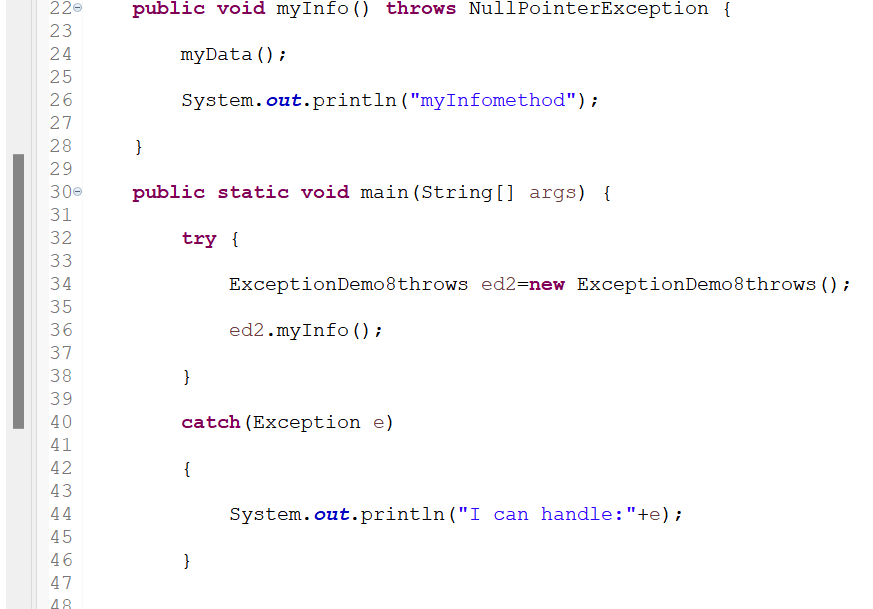


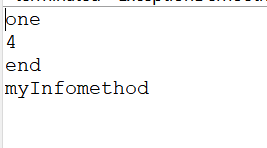
Program 7



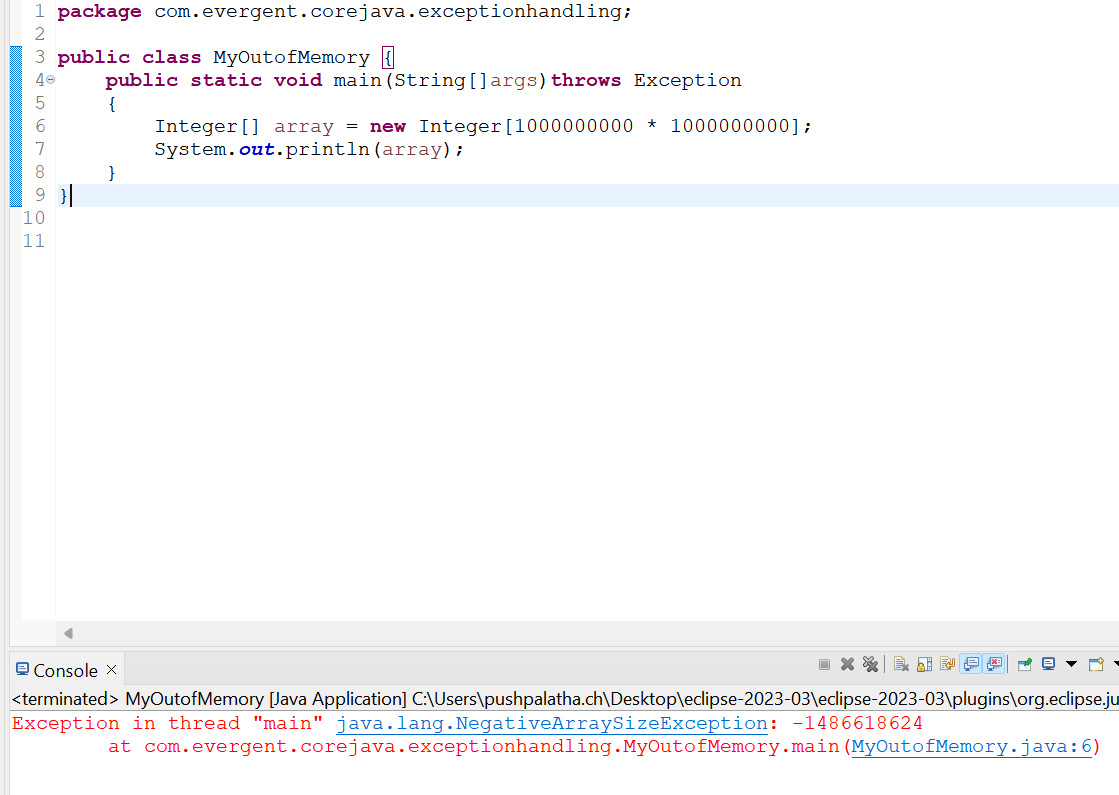
**Program 8**





**Program 9**

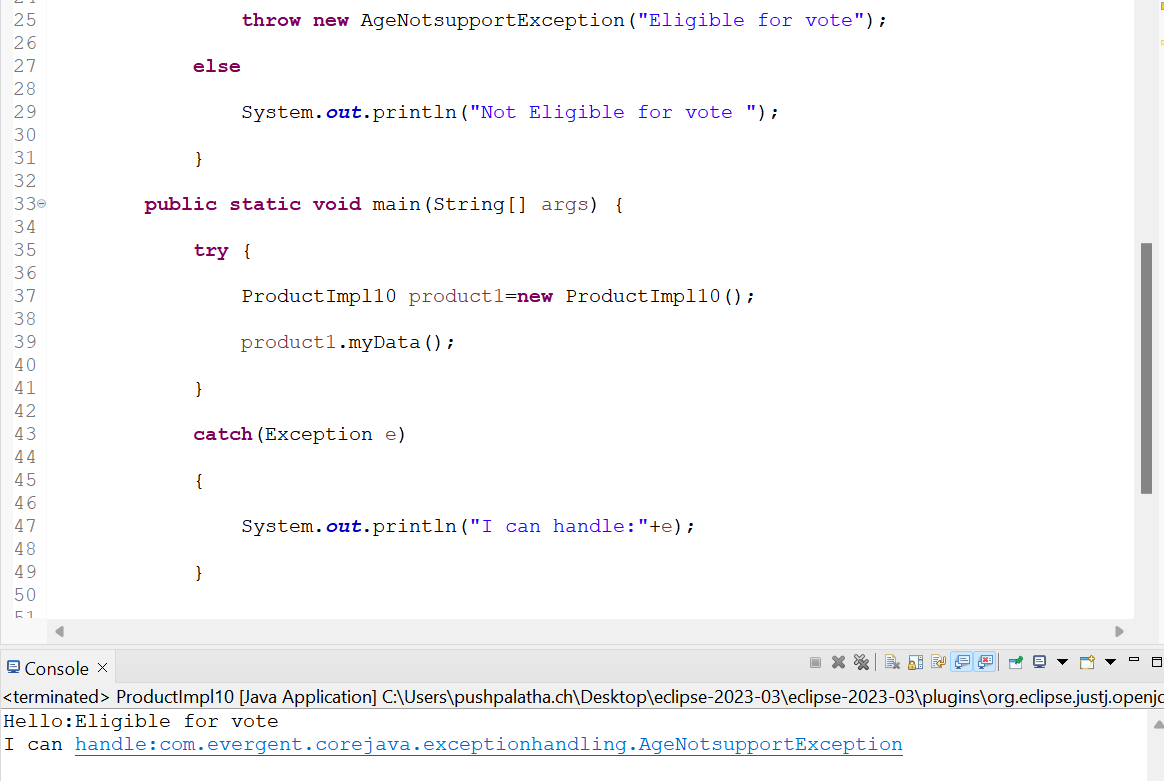
OutOfMemory Error:



**Program 10**

UserDefinedException:AgeNotSupportedException.



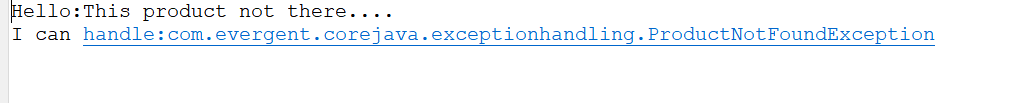


**Program 11**

UserDefinedException:ProductNotFoundException







**Program 12**

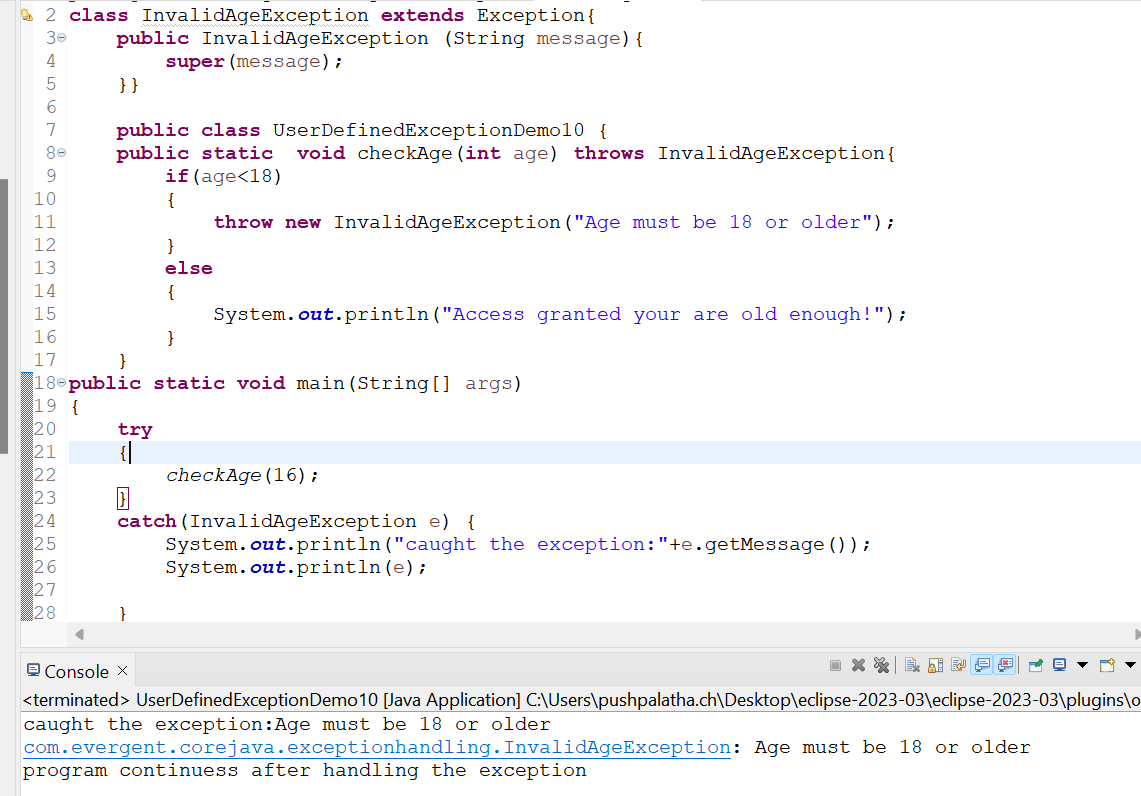
Errors:StackOverFlowError

Developers cannot handle errors



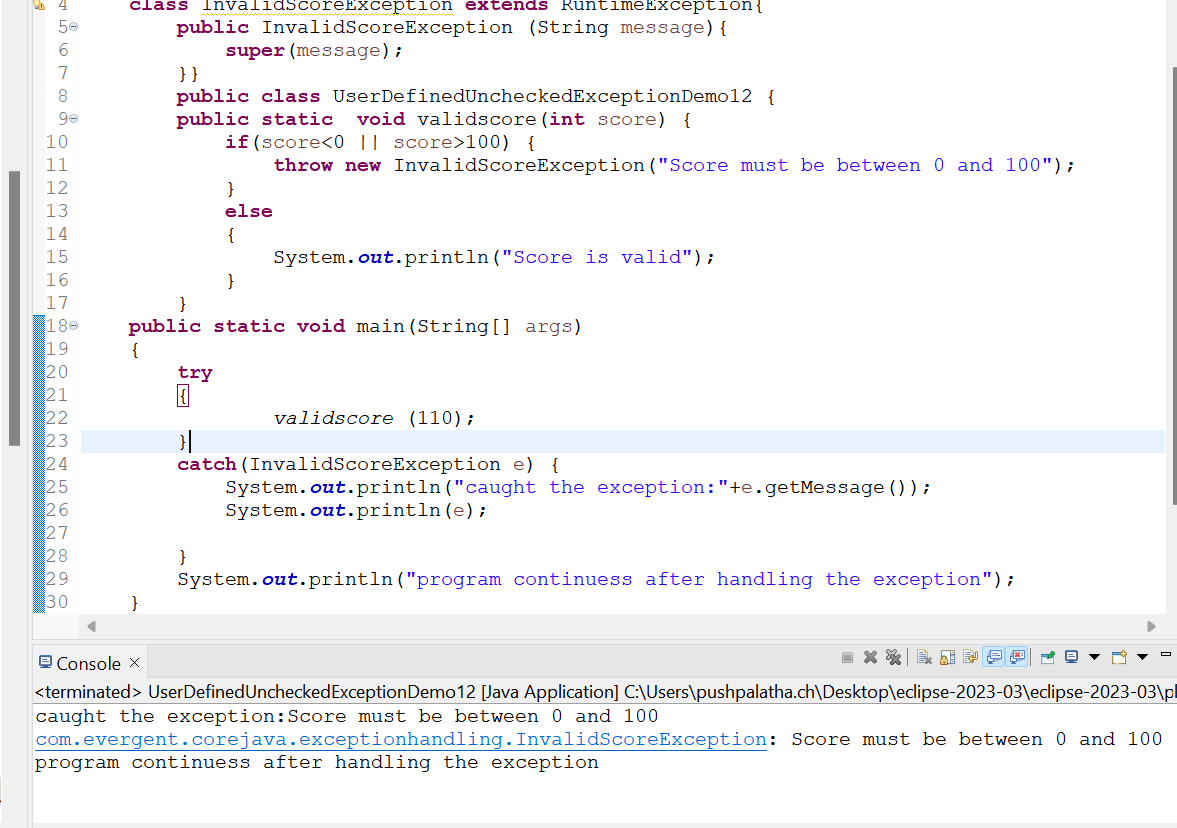
**Program 13**

**UserDefinedException:InvalidAgeException**



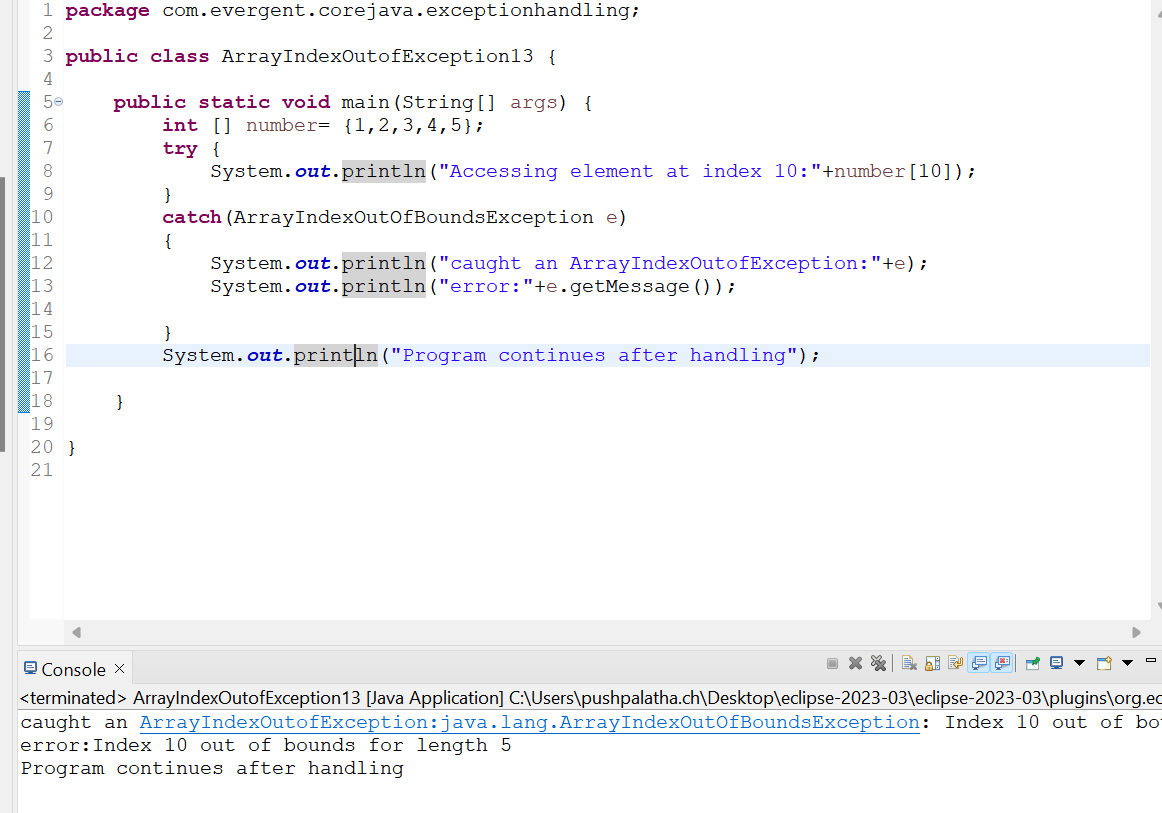
**Program 14**

UserDefinedException:InvalidScoreException



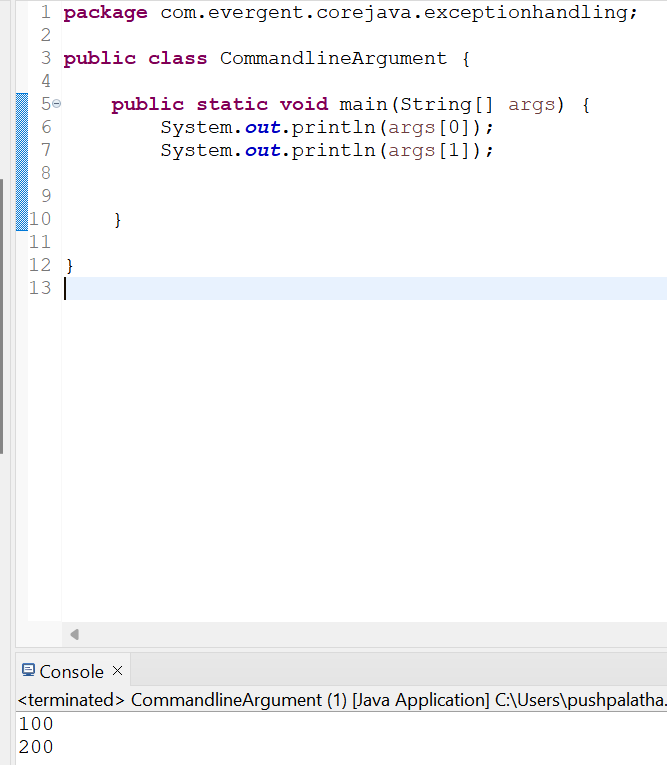
ArrayIndexOutofBound Exception:UncheckedException

**Program 15**



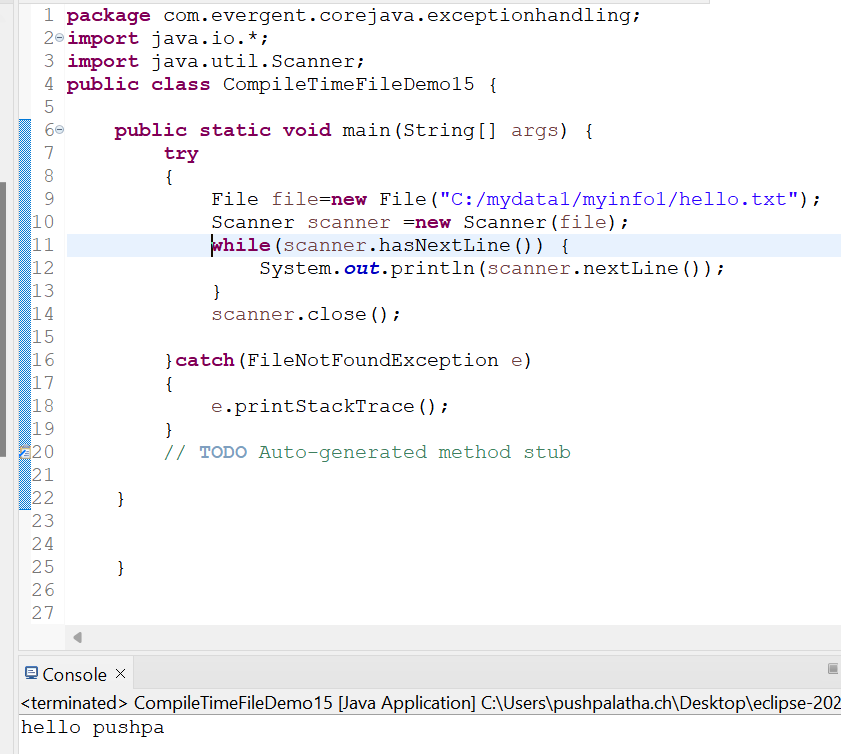
**Program 16**

**Taking args from the main method and passing arguments to it in the command Line arguments in the eclipse.**



**Program 17**

**CompileTimeException:FileNotFoundException**



**Git steps**

git init:To initialize the values. 

git status:To check the status of the file. 

git add . :To add all the files . 

git commit -m “filename/folder name” : to commit the changes of the file. 

git branch:To check the branch. 

git remote add origin url : to add the repository created file which is ready to push in

the nextstep. 

git push --force origin master : to push the file into the repository.

%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D1%3Brotatable%3D0%3Bdashed%3D0%3BlabelPosition%3Dright%3Balign%3Dleft%3BverticalAlign%3Dmiddle%3BspacingTop%3D0%3BspacingLeft%3D6%3Bpoints%3D%5B%5D%3BportConstraint%3Deastwest%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-33%22%20y%3D%22138%22%20width%3D%22450%22%20height%3D%2210%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%223%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-35%22%20y%3D%22143%22%20width%3D%2210%22%20height%3D%2234%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%224%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22411%22%20y%3D%22141%22%20width%3D%2210%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%225%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22184%22%20y%3D%22125%22%20width%3D%2210%22%20height%3D%2219%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%226%22%20value%3D%22java.lang.Throwable%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22108.5%22%20y%3D%2297%22%20width%3D%22161%22%20height%3D%2231%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%227%22%20value%3D%22Errors%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-63.5%22%20y%3D%22181.5%22%20width%3D%2262%22%20height%3D%2231%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%228%22%20value%3D%22Exceptions%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22363.5%22%20y%3D%22185.5%22%20width%3D%2296%22%20height%3D%2231%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%229%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-33%22%20y%3D%22213%22%20width%3D%2210%22%20height%3D%2219%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2210%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22407%22%20y%3D%22216.5%22%20width%3D%2210%22%20height%3D%2219%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2211%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22231%22%20y%3D%22235.5%22%20width%3D%22352%22%20height%3D%2210%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2212%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22228%22%20y%3D%22238%22%20width%3D%2210%22%20height%3D%2237%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2213%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22578%22%20y%3D%22238%22%20width%3D%2210%22%20height%3D%2237%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2214%22%20value%3D%22checked%20exceptions%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22143.5%22%20y%3D%22284.5%22%20width%3D%22159%22%20height%3D%2231%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2215%22%20value%3D%22Unchecked%20exceptions%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22461.5%22%20y%3D%22282.5%22%20width%3D%22179%22%20height%3D%2231%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2216%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22221%22%20y%3D%22315.5%22%20width%3D%2210%22%20height%3D%2219%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2217%22%20value%3D%22%22%20style%3D%22line%3BstrokeWidth%3D2%3Bdirection%3Dsouth%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22556%22%20y%3D%22307%22%20width%3D%2210%22%20height%3D%2219%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2218%22%20value%3D%22compile%20time%20exceptions%26lt%3Bdiv%26gt%3Bfile%20not%20found%20exception%26lt%3B%2Fdiv%26gt%3B%26lt%3Bdiv%26gt%3Binterupted%20exception%26lt%3B%2Fdiv%26gt%3B%26lt%3Bdiv%26gt%3BIo%20exception%26lt%3B%2Fdiv%26gt%3B%26lt%3Bdiv%26gt%3BSQL%20exception%26lt%3B%2Fdiv%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22137%22%20y%3D%22332.5%22%20width%3D%22189%22%20height%3D%22108%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2219%22%20value%3D%22Run%20time%20exception%26lt%3Bdiv%26gt%3Bnull%20pointer%20exception%26lt%3B%2Fdiv%26gt%3B%26lt%3Bdiv%26gt%3BArithmetic%20exception%26lt%3B%2Fdiv%26gt%3B%26lt%3Bdiv%26gt%3BArray%20index%20out%20of%20bound%20exception%26lt%3B%2Fdiv%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3BfontSize%3D16%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22428.5%22%20y%3D%22326%22%20width%3D%22265%22%20height%3D%2289%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3