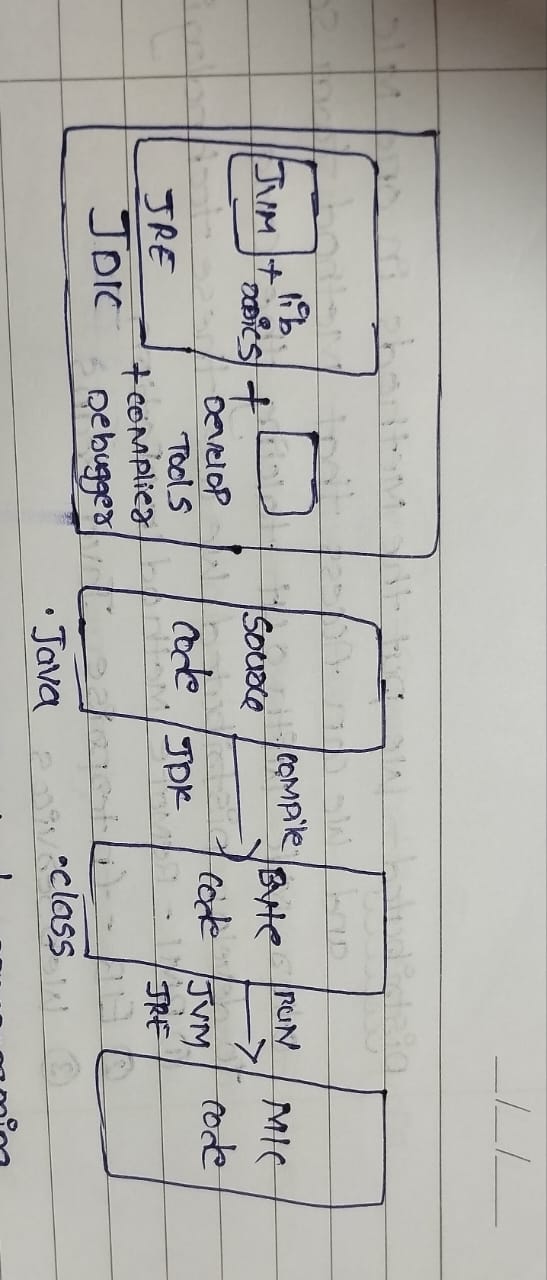
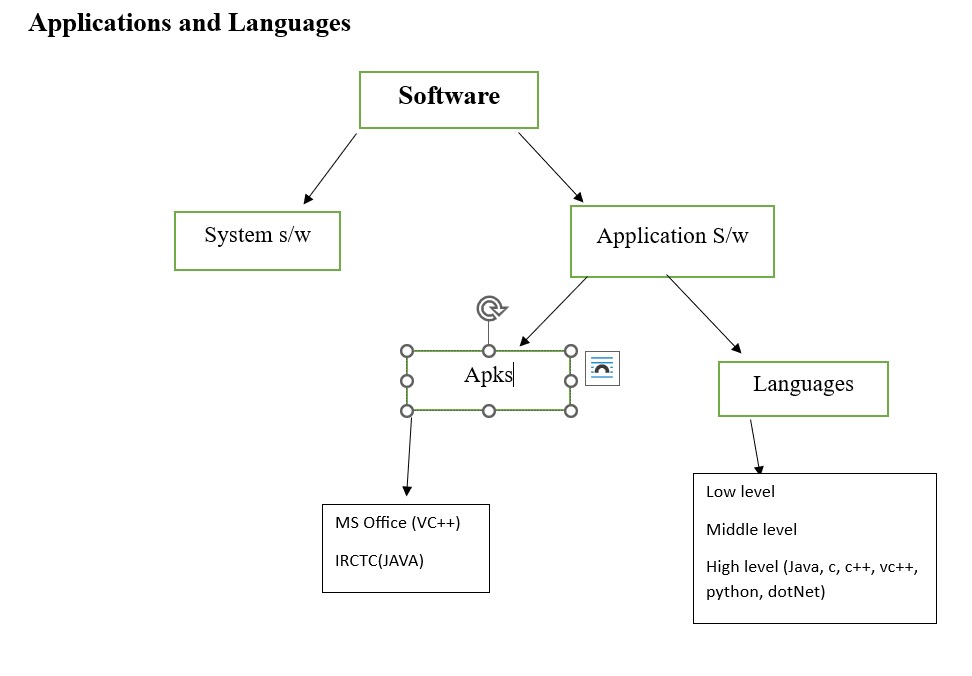
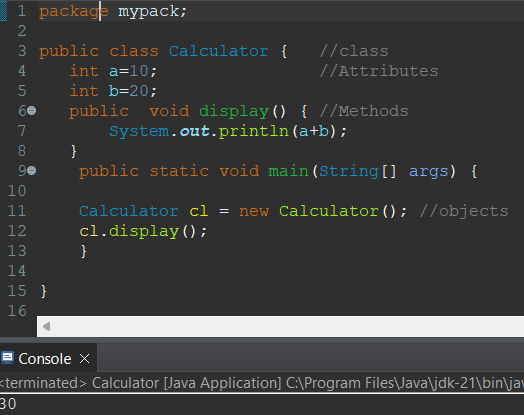
**CORE JAVA**

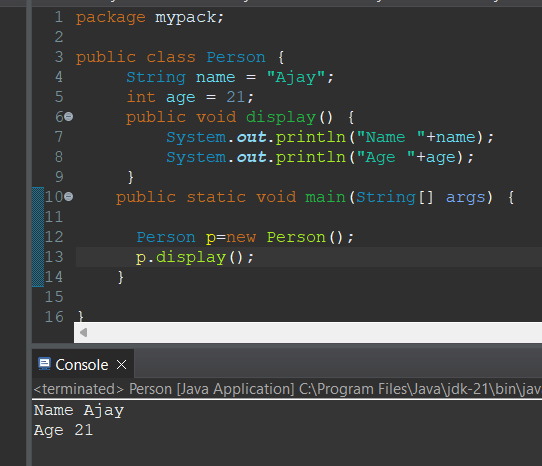
******Def:** Java is a concept oriented language

**Why Platform Independent:**

**OOPS CONCEPTS:**

**Encapsulation:** It is a mechanism Binding Attributes & Methods together inside a class & Object creation.

****

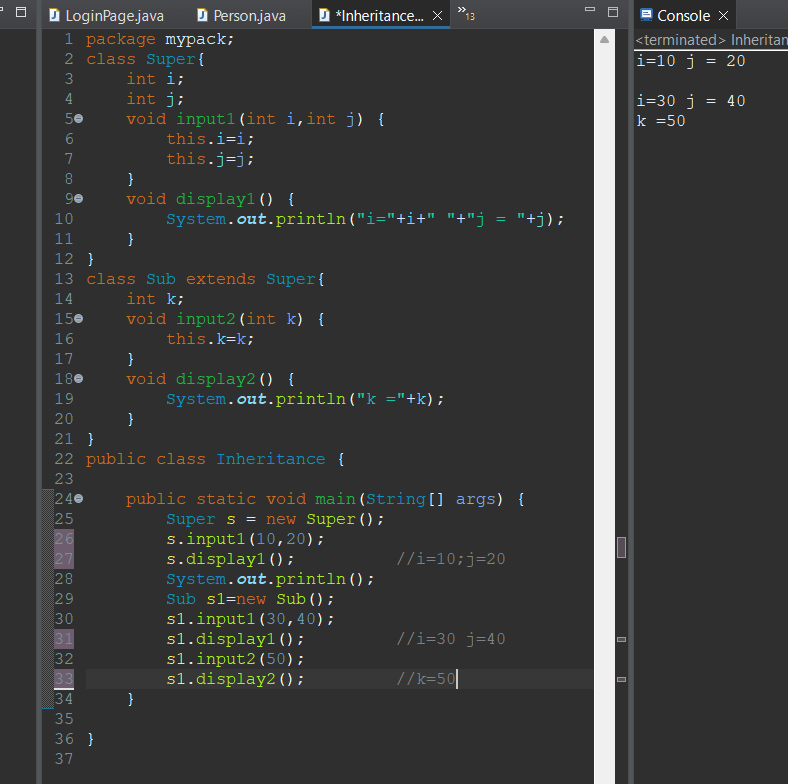


**Inheritance**: Reusability of existing code from super class to sub class

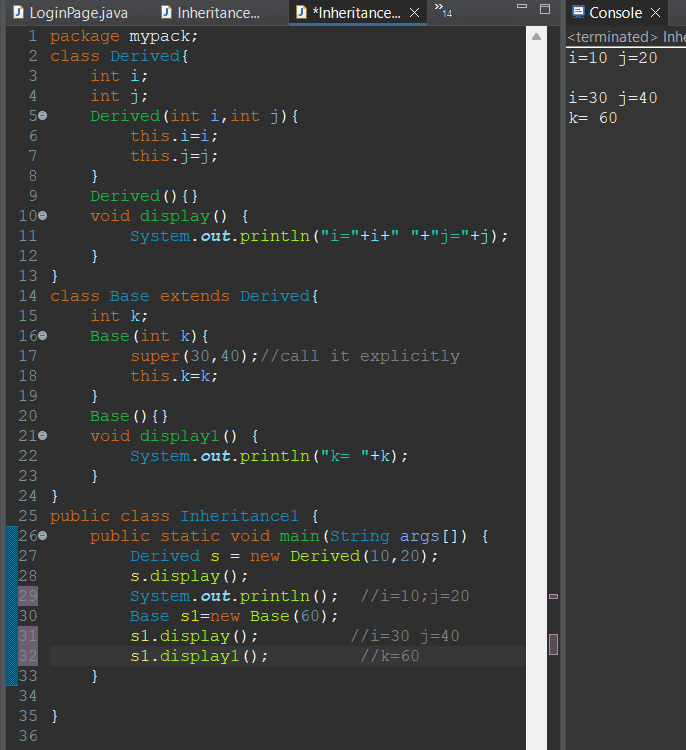
-it support multi-level inheritance.

-it won’t support multiple inheritance through java due to ambiguity.

-it supports multiple inheritance through interface.

-to implement this we use extends keyword.

Ex-2: when we are assigning the value to a variable through class constructor



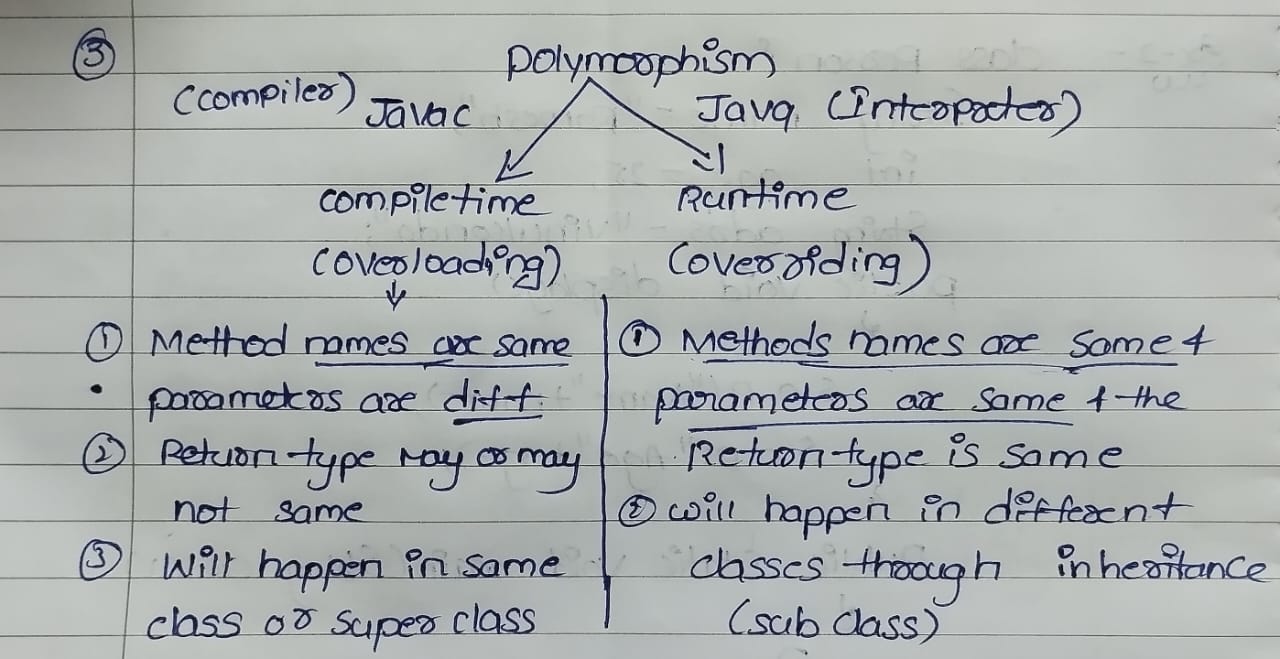
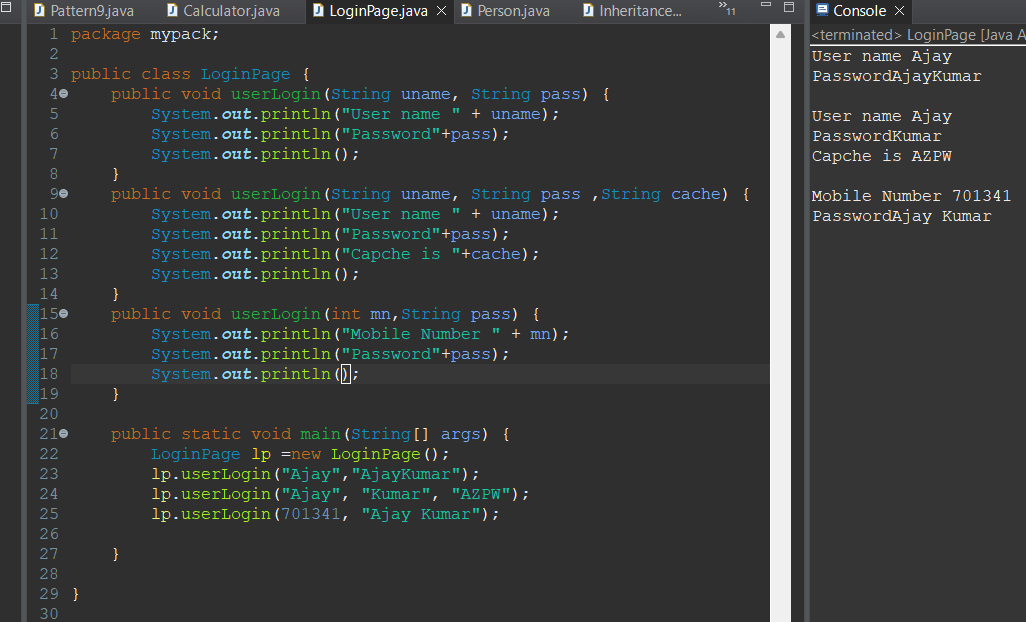
**Note:**

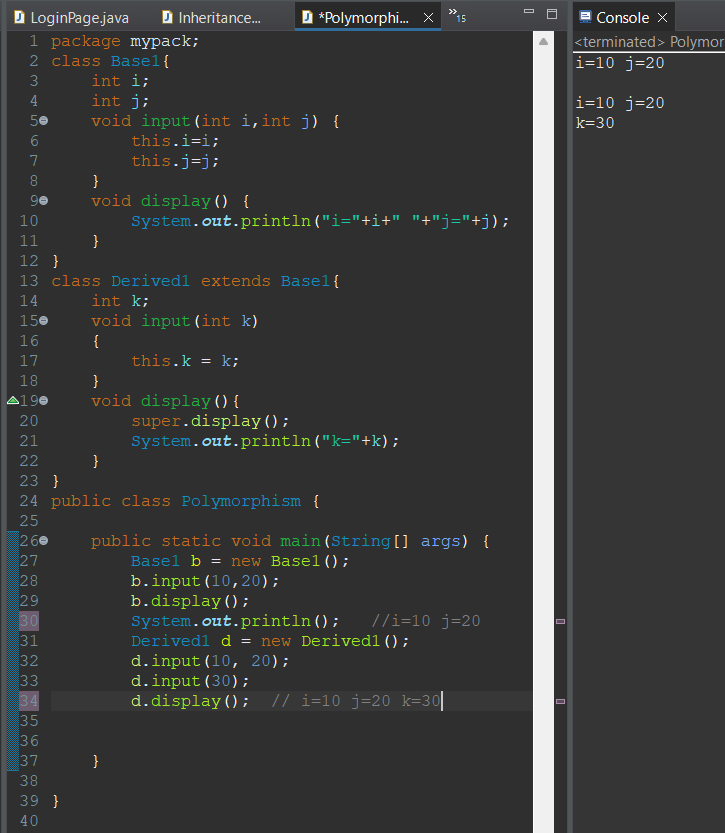
-If the class is public then the name of the source code file should be same as the class name

- In a source code file we can have only one public class

**Polymorphism:** Many forms

One thing can act as differently in different situations is called polymorphism.

ex-2: A login method which is used in various ways to login.

Ex: Method overloading and over riding

**Abstraction:** Hiding the internal details showing only Necessary one

**Day-2**

**Constructors:**

**-**class name & Constructor name should be same

-it will call while object creation

-there are two types of constructor(default &Parameter)

-always a constructor is overloaded.

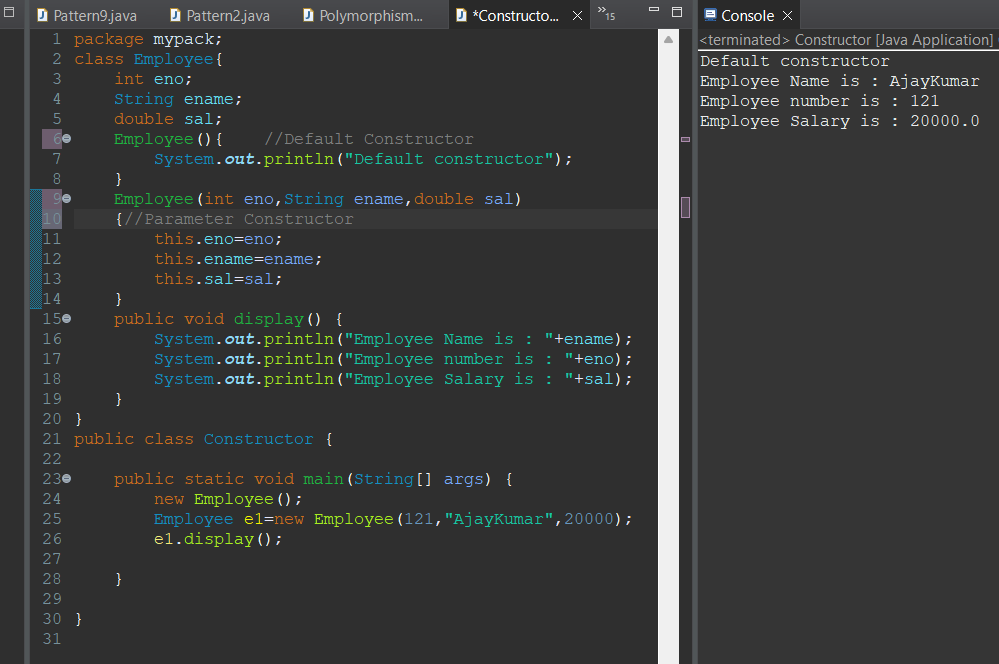
-Constructors are mainly for initialize purpose

-“this” is a keyword for pointing to instance variables.

-Constructor does not have any return type not even void also.

-Each and every class need one default constructor.

- If a class contains a parameterized constructor then it is the responsibility of the programmer to create default constructor

-If a method name is same as the class name with return type then this method is treated as ordinary method but not as constructor

**Static:**

**-**Static is Keyword

- We can declare static as variables and methods

-we can access static variables and methods (clasname.varnam,clasnam.methna)

-Static block will be called while class is loaded into JVM .

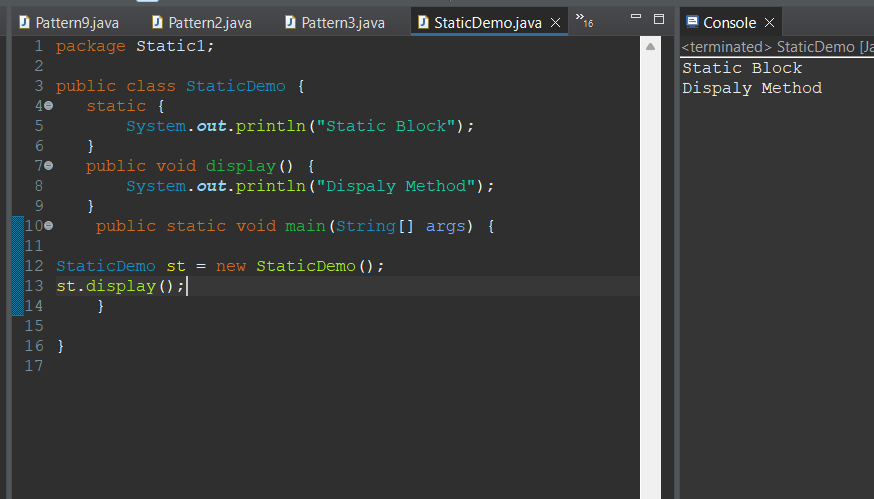
- A static method can call only other static methods & Variables directly

-Static Method Cannot call Non-Static Methods

-Non-Static Methods Can Access static methods & Variables

1)Static block is initialized when the class is loaded in the jvm

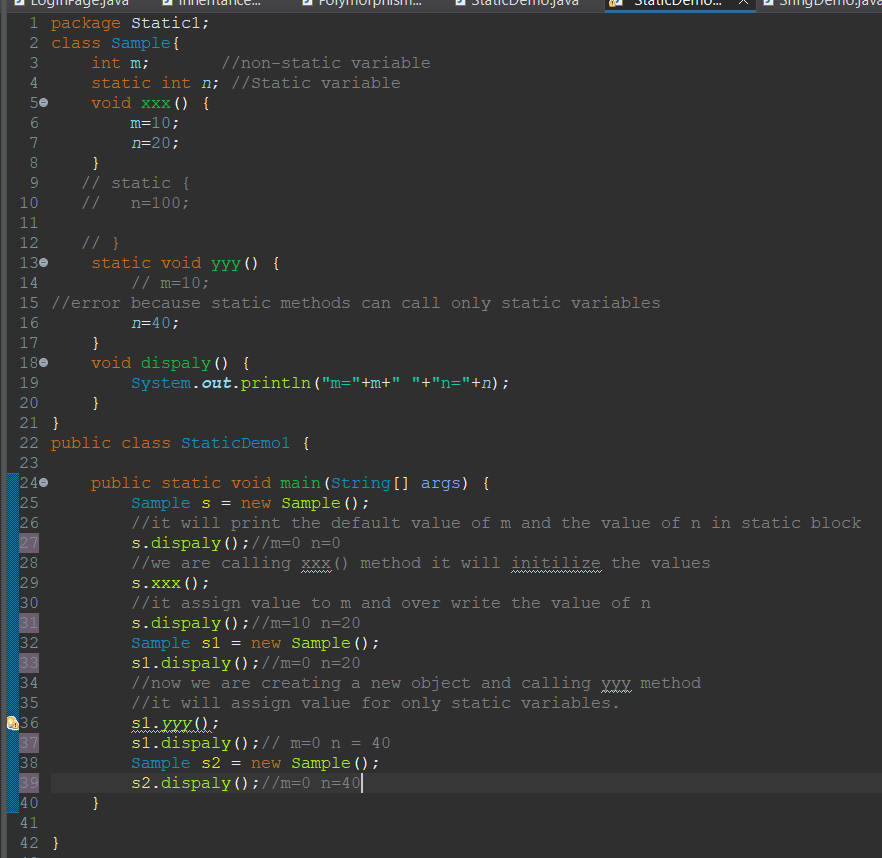
Note: Static Block is Implicitly loaded In java whenever a class is Loaded



**Note:**

When we create the Static Instance variables in class and if we create a object it will modify the original memory value;

-when we create a non-static Instance variables in a class and if we create a object it will have only reference not the original values when you create a another new object you get default values.

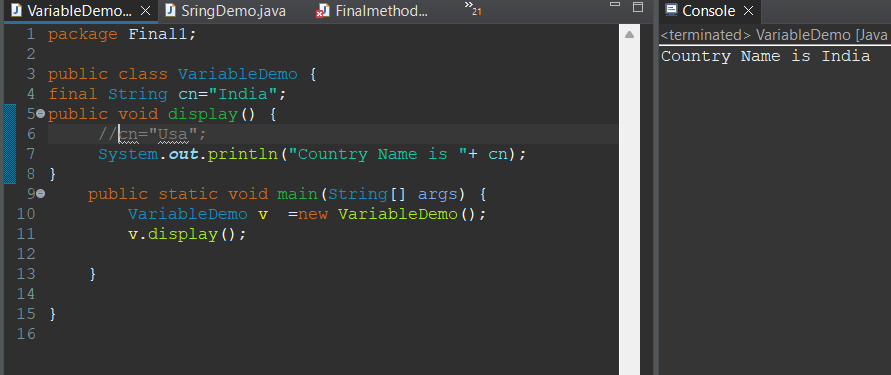


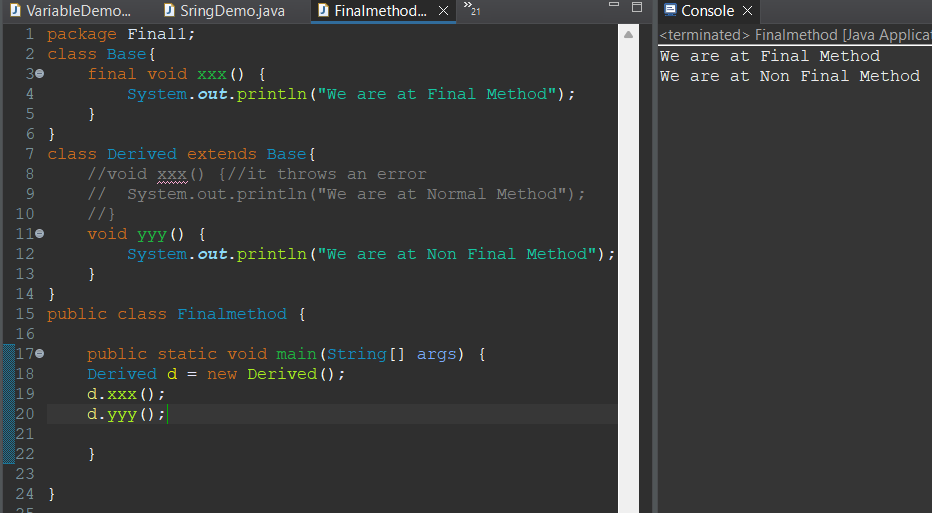
**Final:**

-Final is a Keyword

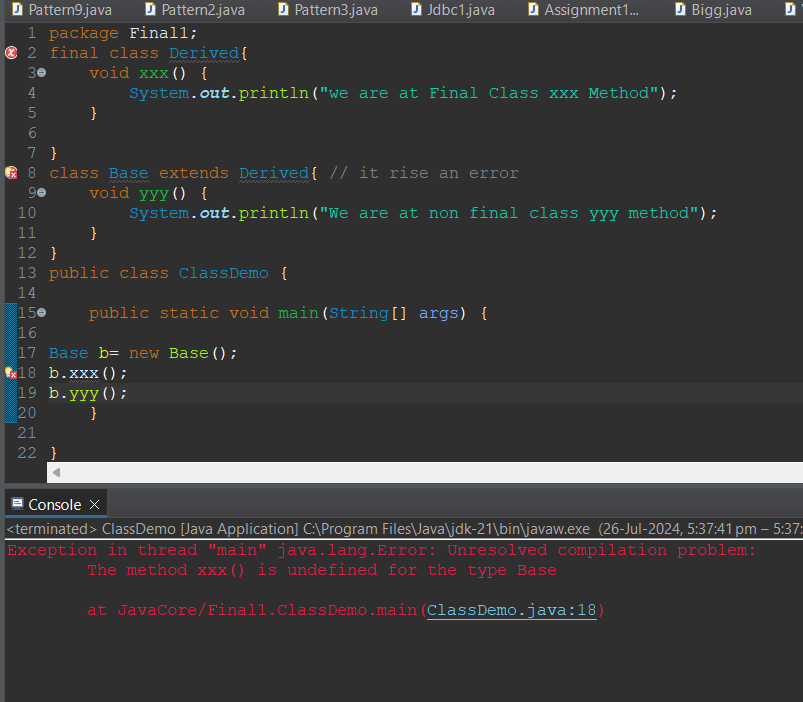
-we can declare as a final as variables ,methods ,class.

-Final Variable we cannot be Modify



-Final Methods we cannot be Override

-Final Class We cannot be Inherited.



**Note:** All Wrapper classes are Final

Integer, Float, Double, Boolean, Byte, Short, Character, Long.

Q)Difference Between Method and Constructor?

-A Constructor helps in initialization an object

-A Constructor has no return type & cannot be inherited

-A method has a return type can be inherited by subclass.

**Packages:**

**-**A Package is a collection of classes and interfaces

-package Keyword is used to create a package

Eg:

Package mypack;

-package Statement should be the first statement in the source code file

-In a source code file we can have only one package statement.

-Only public class/members(\*) are accessible outside the package

-To use the class of one package into another we need to import the package

Eg: import packagename.classname;

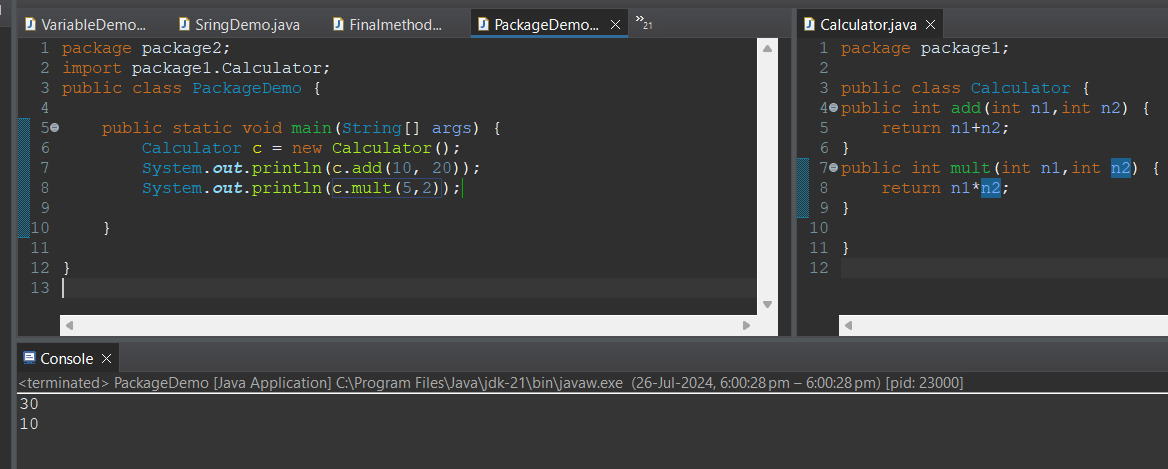
**Predefined Packages :**

**-**java.lang(default package not mandatory to import by default it will import)

-java.util

-java.io

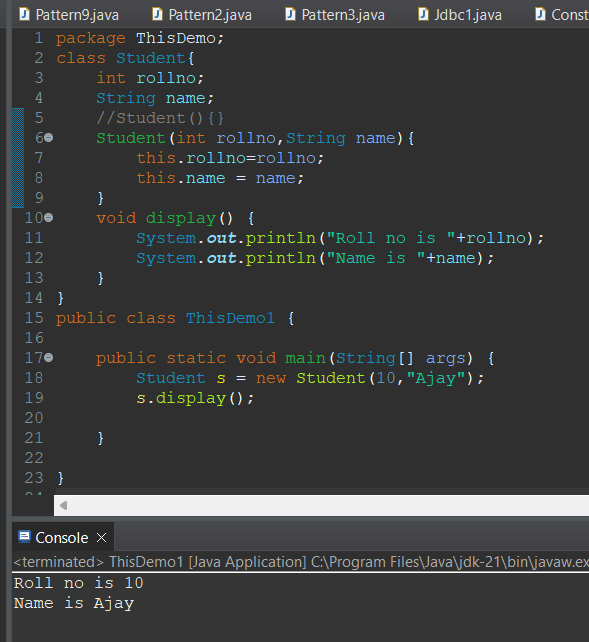
-java.sql



**This:**

**-**This is a keyword

-It is used to refer the Instance Variables



**Object Class:**

**-**Object Class is a top most class .

-Any Class in java is a sub class of Object Class

-Object class is also called as Cosmic Super class

**Methods in Object class :**

* **clone()** : used to create a copy of object

Eg : Emp e = new Emp();

Emp e1 = e.clone();

* **equals()** :used to compare hashcode of two objects

eg: sop(e.equals(e1));

* **finalize():** used for cleaning operation like closing the file and database connections.

-it is called just prior to the garbage collection

Eg:

Public void finalize(){

Out.close();

Con.close();

}

* **getClass()** : used to return the class name of an object

eg:

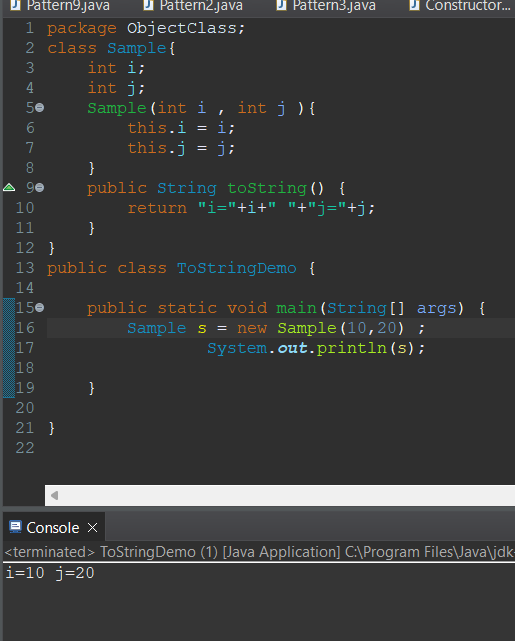
sop(e.getclass());//class name

* **hashcode():** used to return the hash code of an object

eg:

sop(e.hashCode());

* **toString()**: return the class name of object along with hashcode;

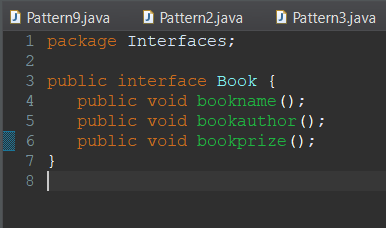


* wait()
* wait(long)
* wait(long,int)
* notify()
* notifyAll()

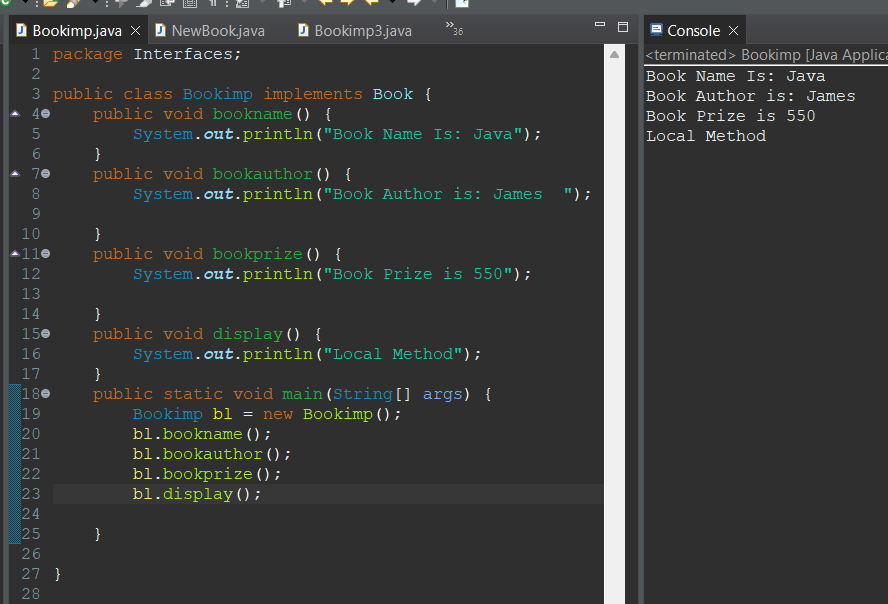
**Day-3:**

**Interface:**

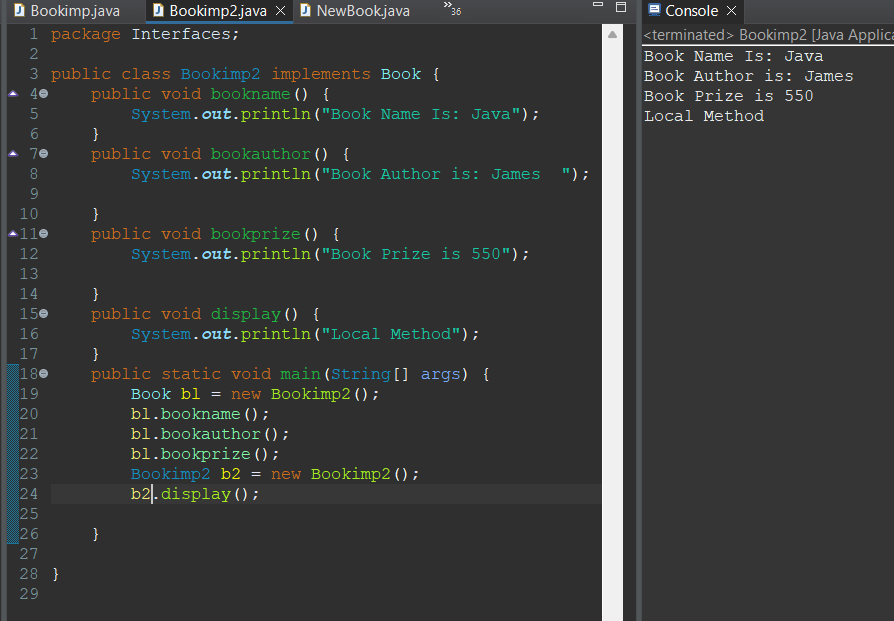
**-**Interface is a Keyword

-Interface having Method signature only there is no body

-If a class implements Interface that class should override all interface methods otherwise the class showing compile time error.



- We cannot create objects for interfaces but we can create references referring to the sub class object. By using this reference we can access only interface methods we cannot access local methods.

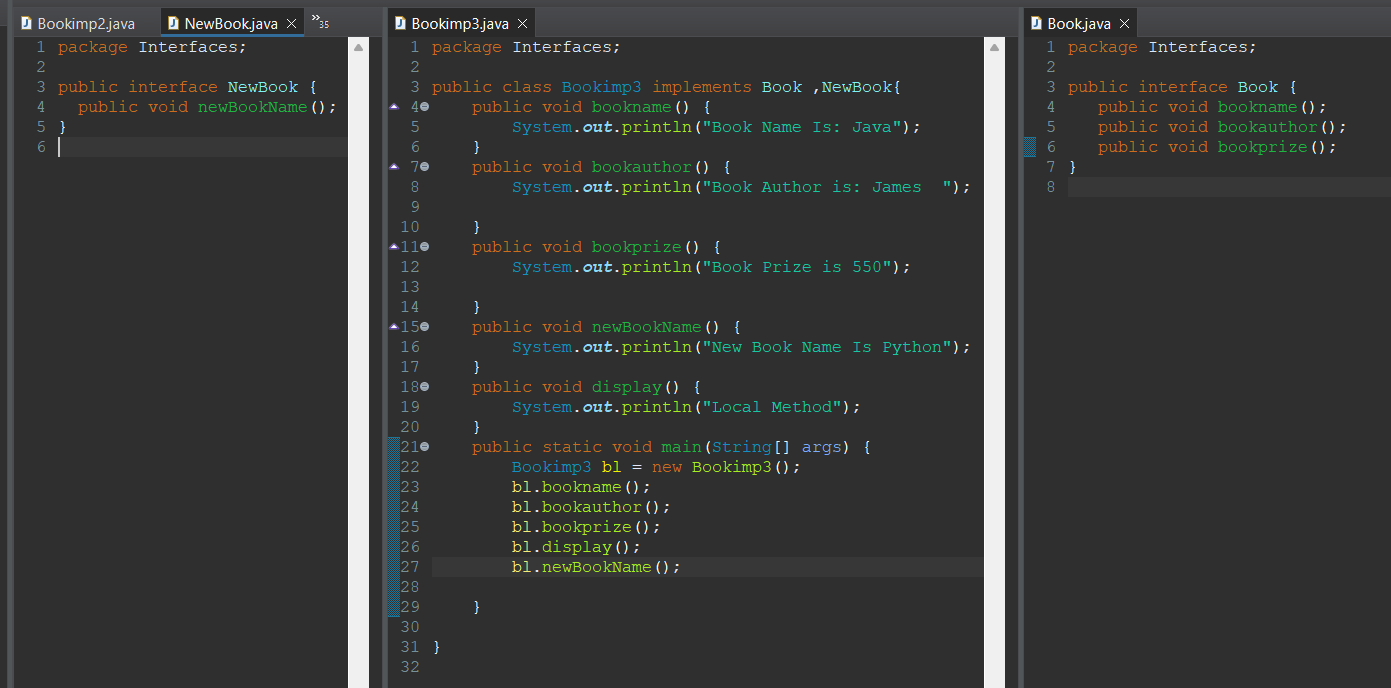


-In an interface all methods are public and abstract by default

- In an interface all variables are public, static and final by default.

-If a class implements an interface then we need to override all methods else make the class as abstract .

-java will support multiple inheritance through interface (our class will implements more than one interface).

**Abstract**:

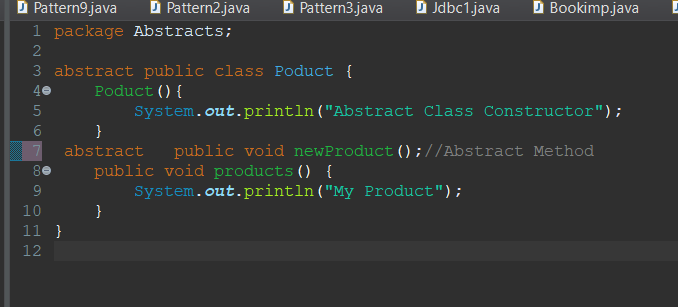
-Abstract is a Keyword

-A method which is just declared is called as abstract method.

-An abstract class contains zero or more abstract methods

-Abstract class having abstract methods and non-Abstract Methods(concrete)

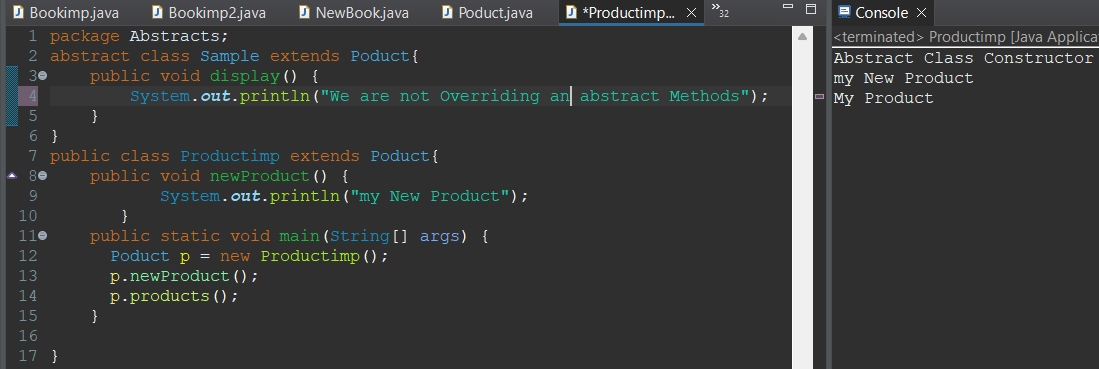
-if any class having one abstract method that class should be declared as abstract class.

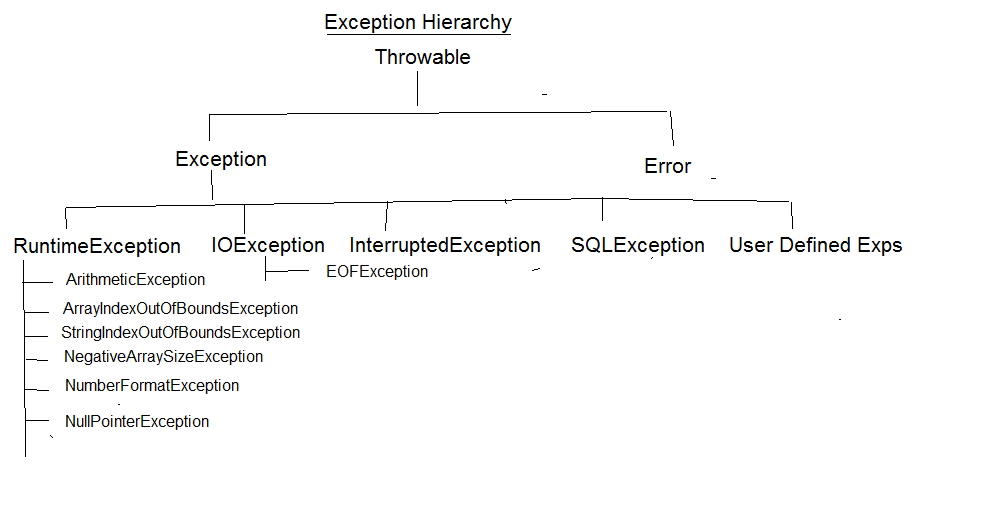


**-** We cannot create objects for abstract classes but we can create references referring to the sub class object.

-We can Create Constructor to Abstract class.

-We can Access Abstract Class Constructor While Sub Class Object Creation.

****

**Exception Handling**

**Categories of Exceptions**:

checked exceptions: All Compile Time exceptions are checked exceptions like IOException, SQLException etc

unchecked exceptions: All Runtime Exceptions are unchecked exceptions like ArithmeticException, ArrayIndexOutOfBoundsException etc

-Any Exception occurs during runtime is a sub class of RuntimeException

**RUNTIME EXCEPTIONS:**

1)Arithmetic exception -divided by zero

2)ArrayindexoutOfBoundException -access element out of index

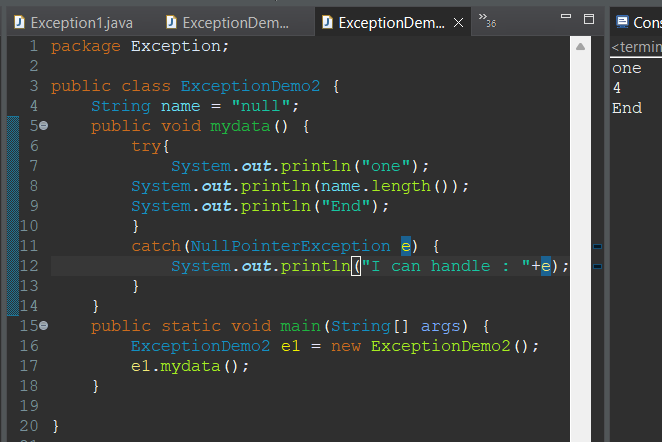
3)StringIndexOutOfBoundException -access when element out of index

4)NegativeArraySizeException -when we initialize with negative index

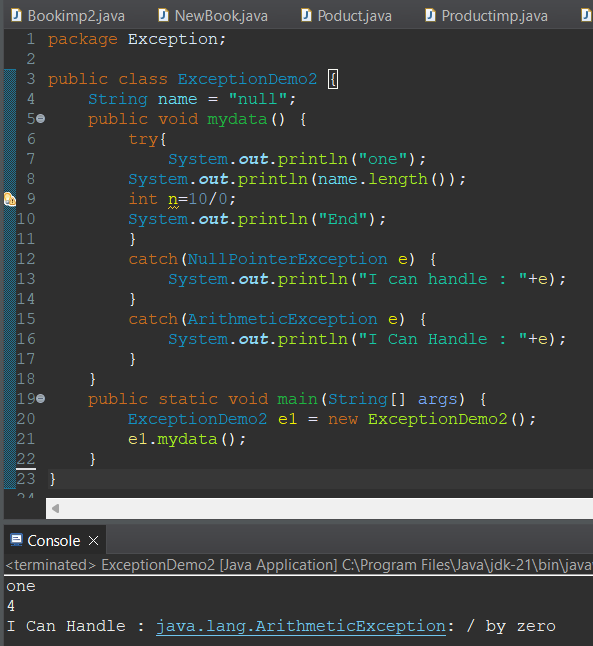
5)NumberFormatException -when we try to convert a string to a numeric which is not possible.

6)NullPointerException -any obect pointing to null by using that object if you access method of String class then is throw exception

-Normal Flow it wont execute any catch block abnormal flow only execute catch block. Below is the example for Normal Flow

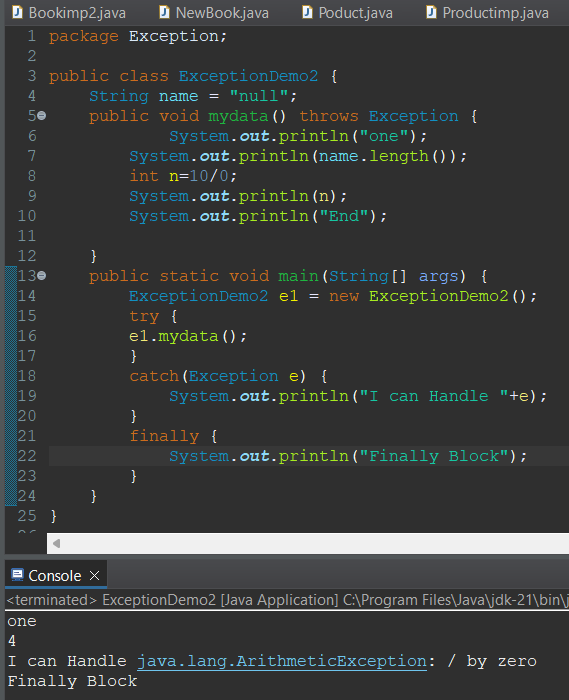


Below is Example for Abnormal Flow



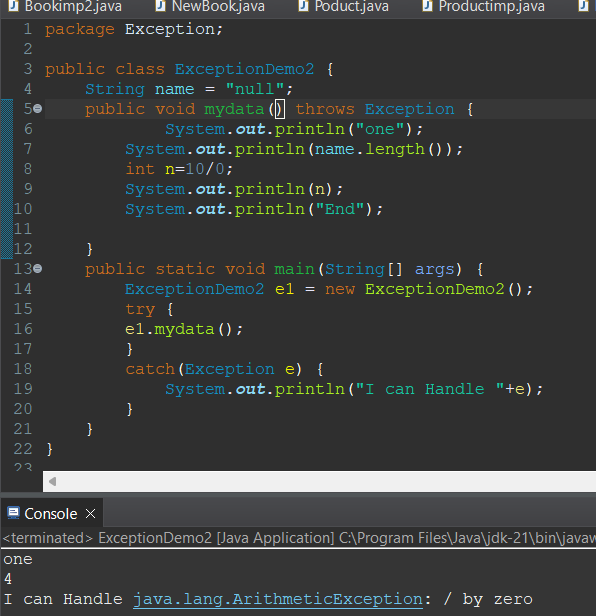
-Finally Block will be executed if exception may occur or may not occur it will executed.

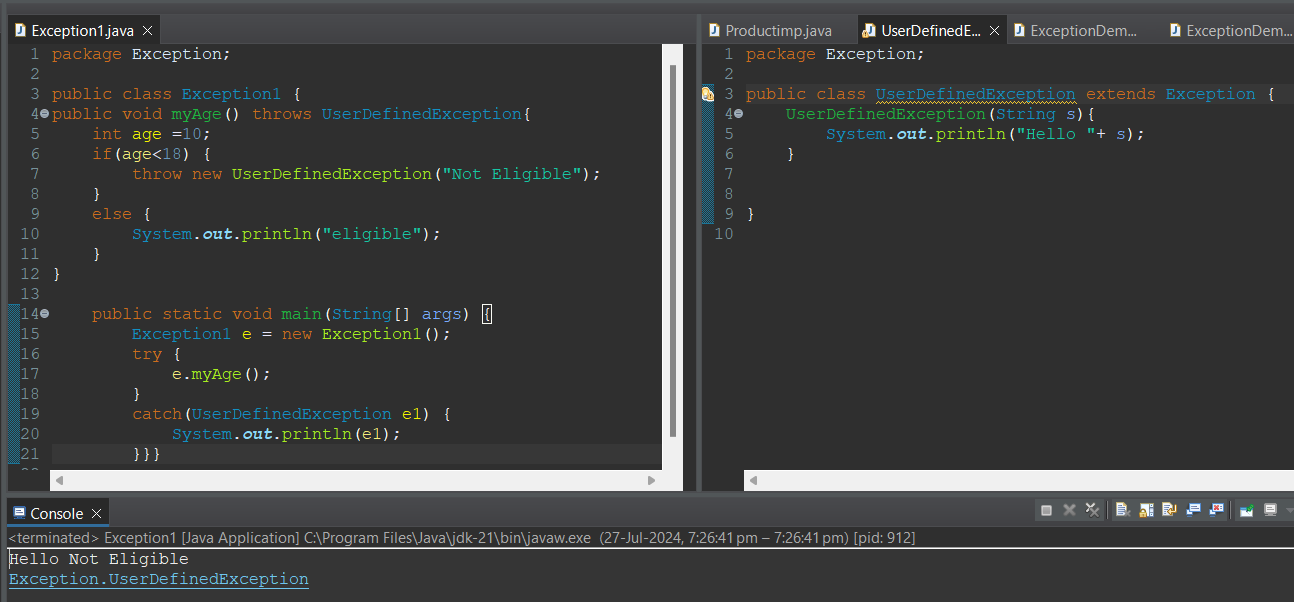
Below is the example code.



-Try followed by catch block or Finally Block

-Throws is keyword it will be executed layer by layer.

e

-Throw is a Keyword it will be executed for runtime and using our own exceptions.

**String:**

-A string is a final class it is present in java.lang.String package.

-String is a group of characters

-string is immutable

Ways to create a string:

-String str = new String(“Ajay”);

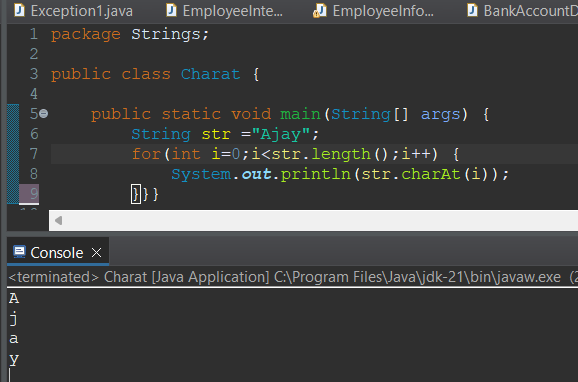
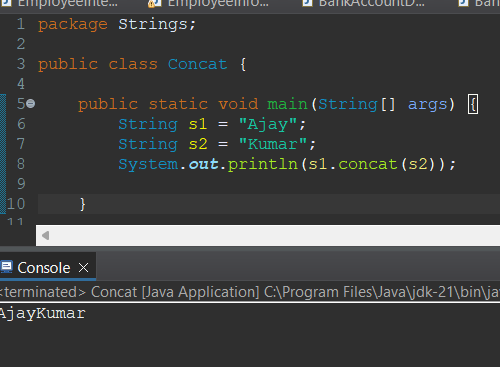
-String str = “Ajay”;

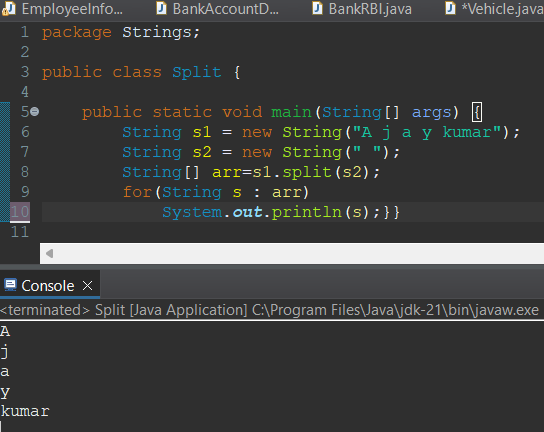
**Note:**

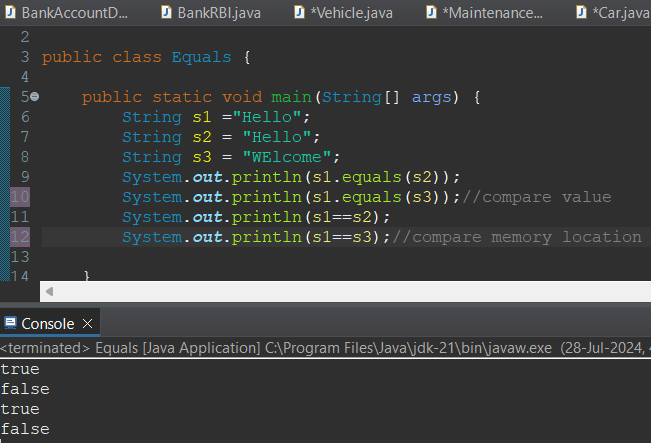
**== ->** Compares the memory location of two objects.

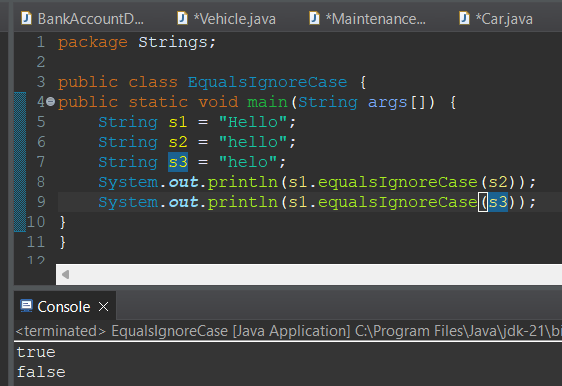
equals -> compares the content of two objects

**Methods of String Class:**

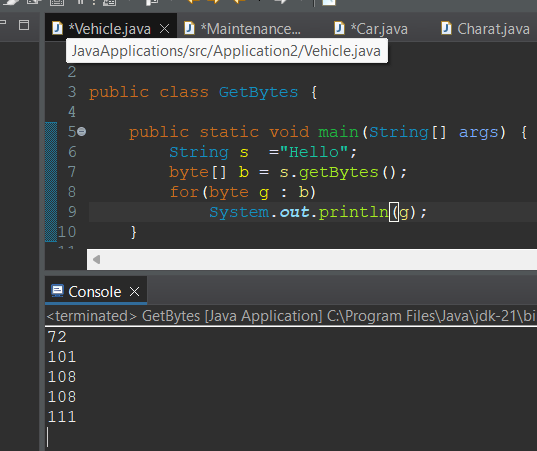
* ****char charAt(index) – used to access each character in a String
* **String concat(String):** combine two Strings
* **split(String):**

By using split() method we can divide the string in to tokens.

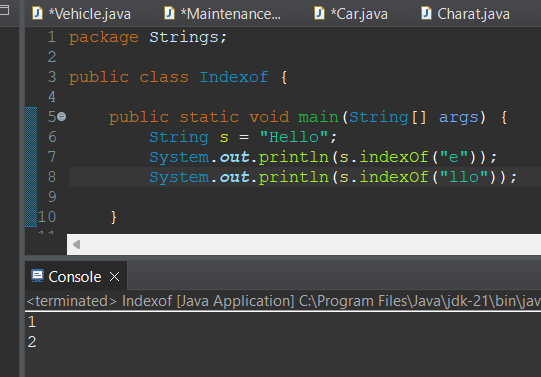
* **Boolean equals(String):** checks if two strings are equal or not.
* **Boolean equalsIgnoreCase(String):** Checks If two Strings are equal ignoring case differences.

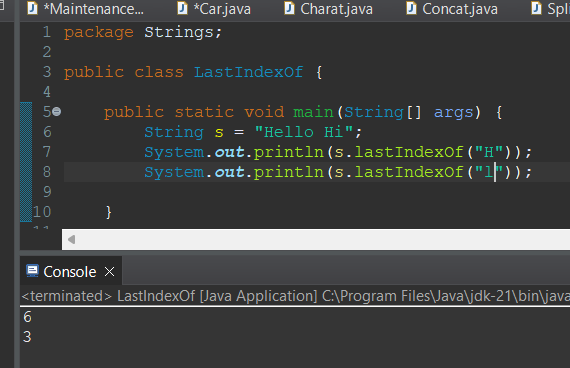
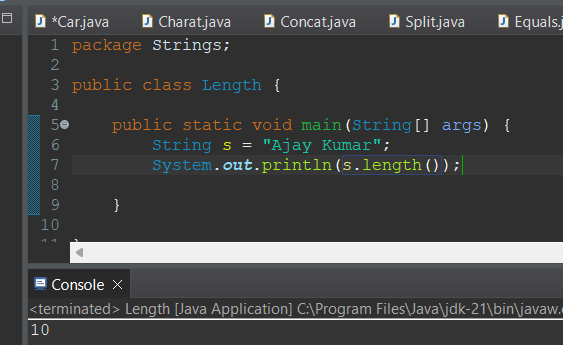


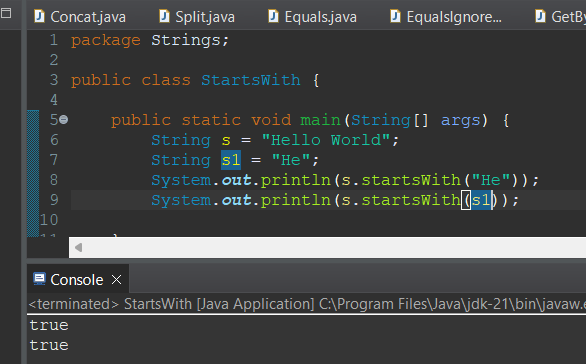
* **getBytes():** Converts a String to a byte array

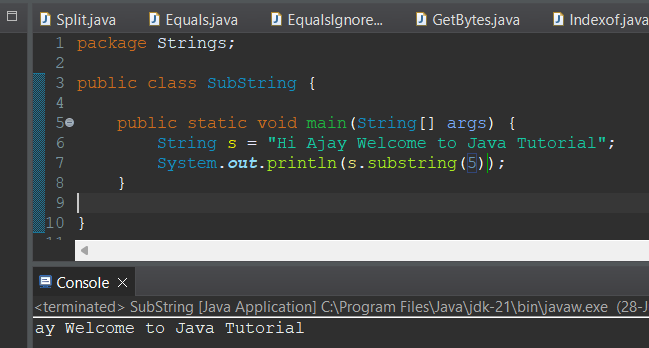
****

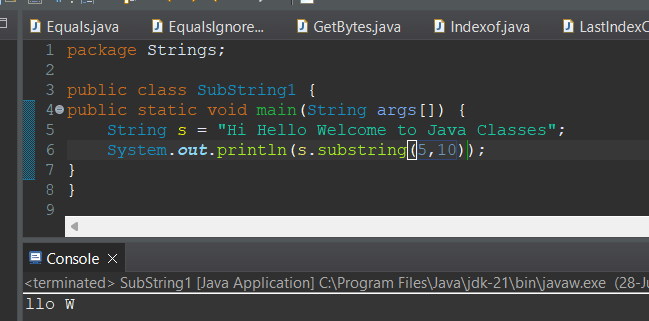
* **int indexOf(string):** Returns the index within the string of the first occurrence of the specified substring.

****

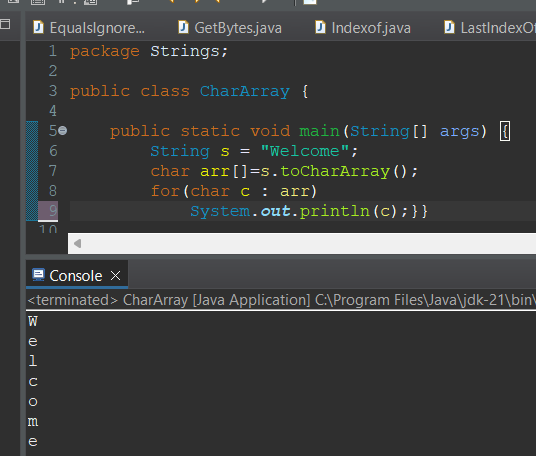
* **int LastIndexOf(string):** Return the index within the string of the last occurrence of the specified substring.
* **int length():** Returns the Length of the string
* **boolean startsWith(string):** checks if the string starts with the specified prefix.

****

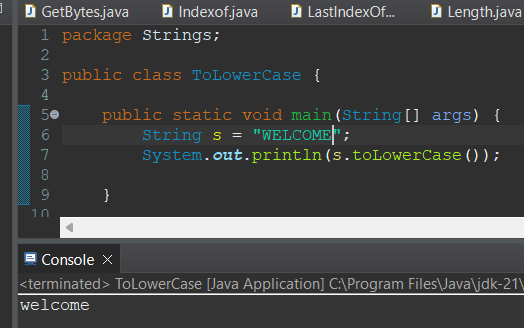
* **String substring(int):** Returns a new string that is a substring starting from the specified index to the end
* **String substring(int,int):** Returns a new String that is a substring starting from the specified start index to the end index.

****

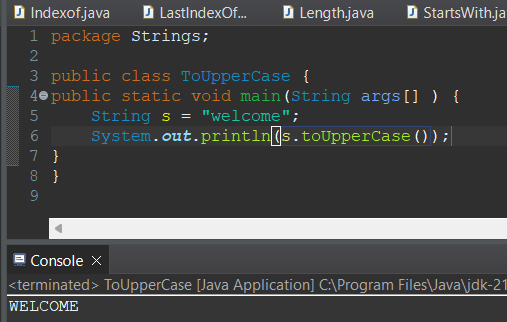
* **Char[] toCharArray():** Converts the string to a character array.

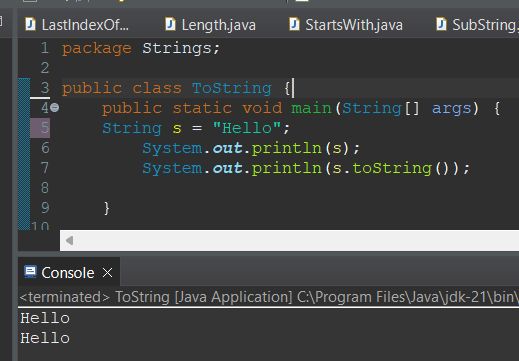
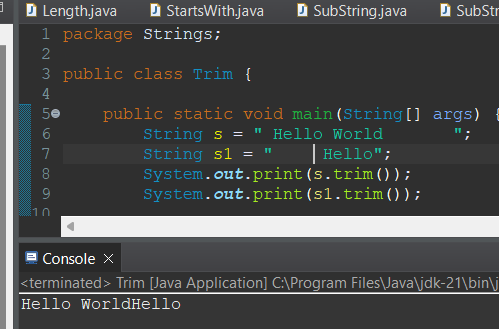
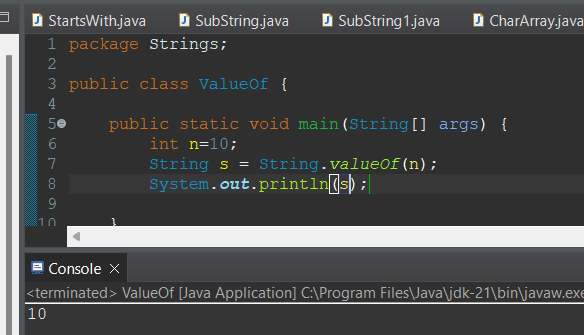
****

* **String toLowerCase():** converts all characters in the string to lowercase.

****

* **String toUpperCase():** Converts all characters in the string to Upper Case

****

* **String toString():** Returns the String itself
* **String trim():** removes leading and tralling white spaces.
* **String valueOf(int):** Converts the specified value to a string.

**Why Strings are immutable:**

-When we create a string object it is constant, if we are trying to modify existing string it will create another memory location.

-existing object is eligible for garbage collector.

**Where it will store:**

-When we create with new it store in heap

-No new keyword store in constant pool

Interview Questions:

1)What is String?

String is a final class it is immutable.

2)Why string is immutable?

3)Difference between String and StringBuffer , StringBuilder?

**String:**

-immutable

-All methods are not synchronized

**StringBuffer:**

-Mutable

-All Methods are synchronized

**StringBuilder:**

-Mutable

-All Mwthods are noy Synchronized

**MultiThreading:**

-Multiple Threads Are Running Simultaneously Reducing CPU Ideal Time.

- Multithreading is used to increase the performance of application

-A Thread Is a Light Weight ,the These Threads are Running on a Separate Path

-Threading is inbuild Mechanism

-Object is Heavy weight

-Thread scheduler allocate the time for threads

-isAlive() : Thread is running or not (It returns Boolean Values).

-Yield(): We a Thred Come to Stop Remaining Thread Time Is Allocated to next Priority Thread.

-join() : Communication Between Main And Child Class

-Suspend(),Resume() combine one.

Java.lang.Thread(is a class)

Java.lang.Runnable(interface)

**How Many ways to create a Thread:**

-We can create a Thread in two ways

1)extends Thread class

Ex:

eg:

class MyThread extends Thread

{

public void run()

{//logic goes here

}

}

MyThread t = new MyThread();

t.start();

2)implements Runnable interface

Runnable

---------

class MyThread implements Runnable

{

public void run()

{//logic goes here

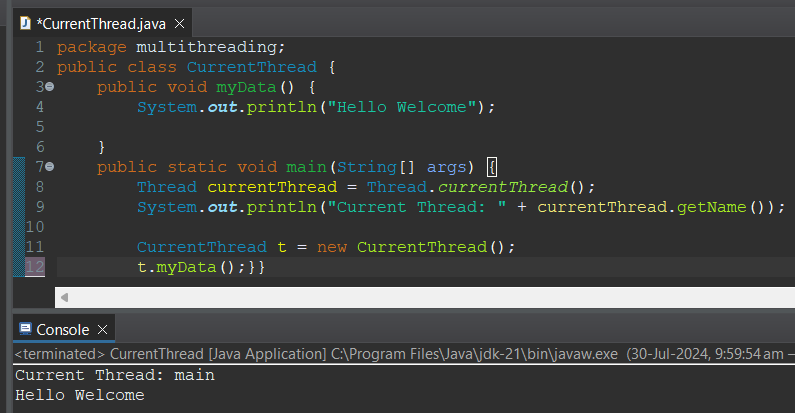
}

}

**Methods of Thread class:**

------------------------e----------

1) **Thread currentThread():** This static method returns a reference to the currently executing thread object.



2) **void setName(String):** This method sets the name of the thread to the specified name.

3) **String getName():**This method returns the name of the thread**.**

**4)** **void setPriority(int):** This method sets the priority of the thread. The valid range is between Thread.MIN\_PRIORITY (1) and Thread.MAX\_PRIORITY (10).

**5)** **int getPriority():**This method returns the priority of the thread.

**6)** **void run():**This method is the entry point for the thread. When a thread is started, the run method is invoked.

**7)** **void start():**This method causes the thread to begin execution; the Java Virtual Machine calls the thread's run method.

**8)** **void join():**This method waits for the thread to die. It allows one thread to wait for the completion of another.

**9)** **void sleep(long):** This static method causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

**10)** **String toString():**This method returns a string representation of the thread, including its name, priority, and thread group.

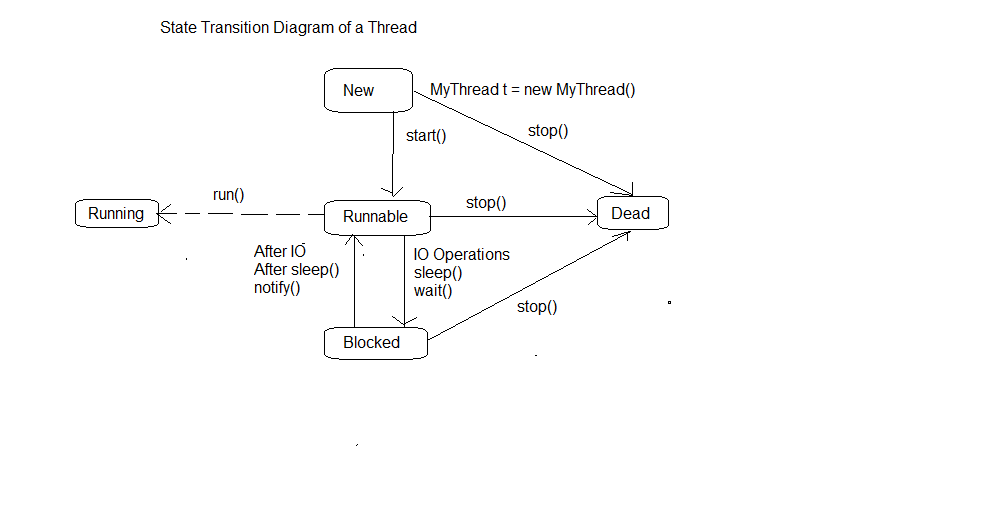
**11)** **void yield():**This static method causes the currently executing thread object to temporarily pause and allow other threads to execute.

**Advantages of Runnable interface over Thread class**

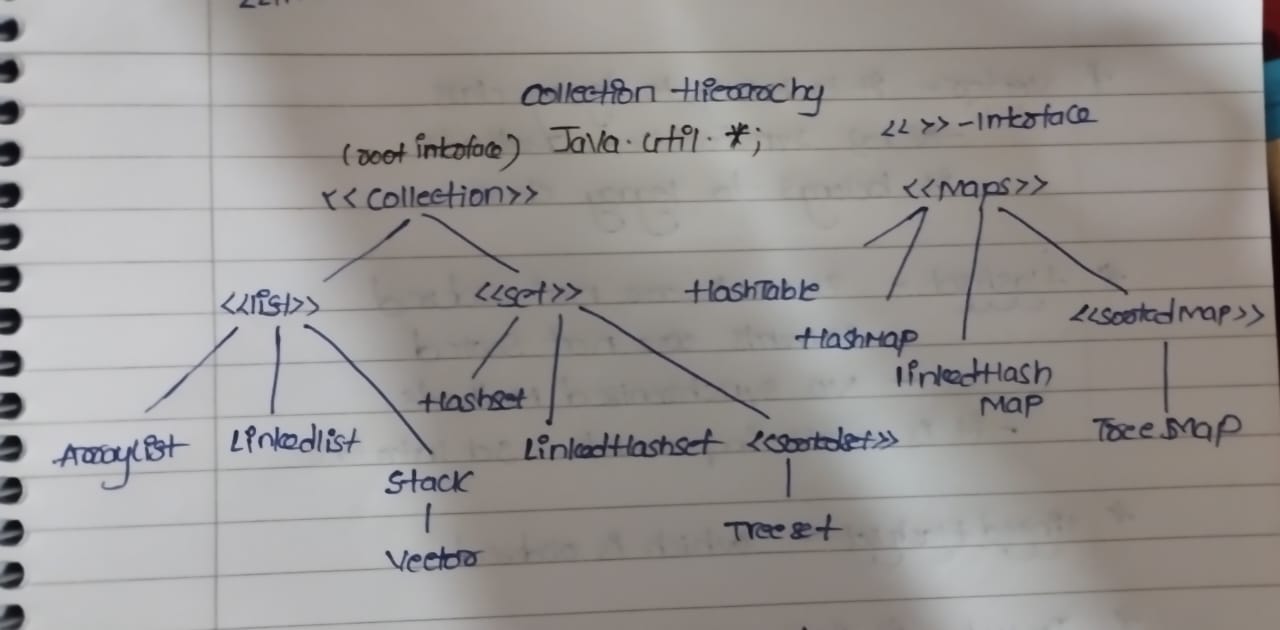
--------------------------------------------------

- If a class already extends another class, to make this class as thread, the class should implement Runnable interface as Java does not support multiple inheritance

- Ensures that run() is overridden

**Life Cycle of a Thread:**

**Collection Framework:**

****

**What is collection Framework:**

**-**it is a predefined framework it will present in the java.util.Collection.

-It will Allow different data type

-Increase their memory location

-Business Logic Methods Can be implement (Add,delete).

**Collection(interface):**

-it is an root interface

**List(interface):**

-It is a Interface which extends Collection interface.

-List allow duplicate elements

-List is Ordered

-The List size is growing and allow any type of data.

**Methods in List Interface:**

**-**boolean add(object)

-void add(int, object)

-void set(int,object)

-boolean remove(int)

-object get(int)

-int size()

**Arraylist(class):**

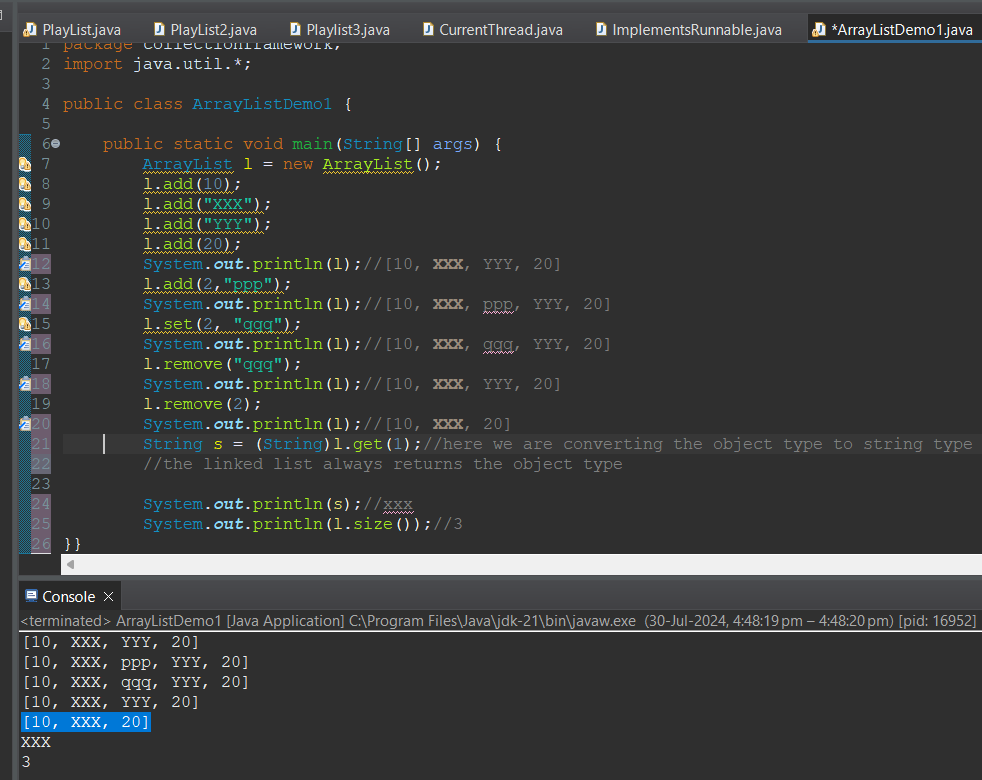
-ArrayList is a class which implements List Interface.

-It is a continuous Memory location

-There is no synchronized Methods

-if initial size is 10 after filling the 10 elements its size increase to 15.

Dis adv: if we want to store only 11 elements the remaining memory is waste.



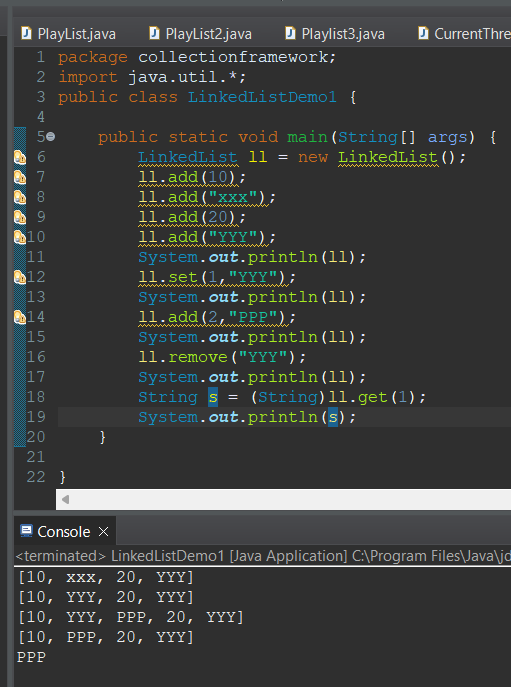
**Linked List(class):**

-Linked List is a class that implements the List Interface.

-the is no continuous memory location in ArayList

-The Methods are not synchronized

-If we want to add & remove an element we can use the Linked List.



**Vector:**

-Vector is a class implements list interface

-Vector Is a legacy class.

-The Methods Inside the Vector or Synchronized.

-same as above the size will be doubled.

**Set(interface):**

-Set is an interface which extends Collection Interface.

-Set will not allow duplicates.

-it is unordered.

-Size is growing allow any type of data.

**Hashset(class):**

-Hashset is a class which implements Set Interface.

-elements are not ordered.

-elements are not sorted.

-no Synchronization methods.

**LinkedHashSet(class):**

-it is ordered.

-it has no synchronization methods.

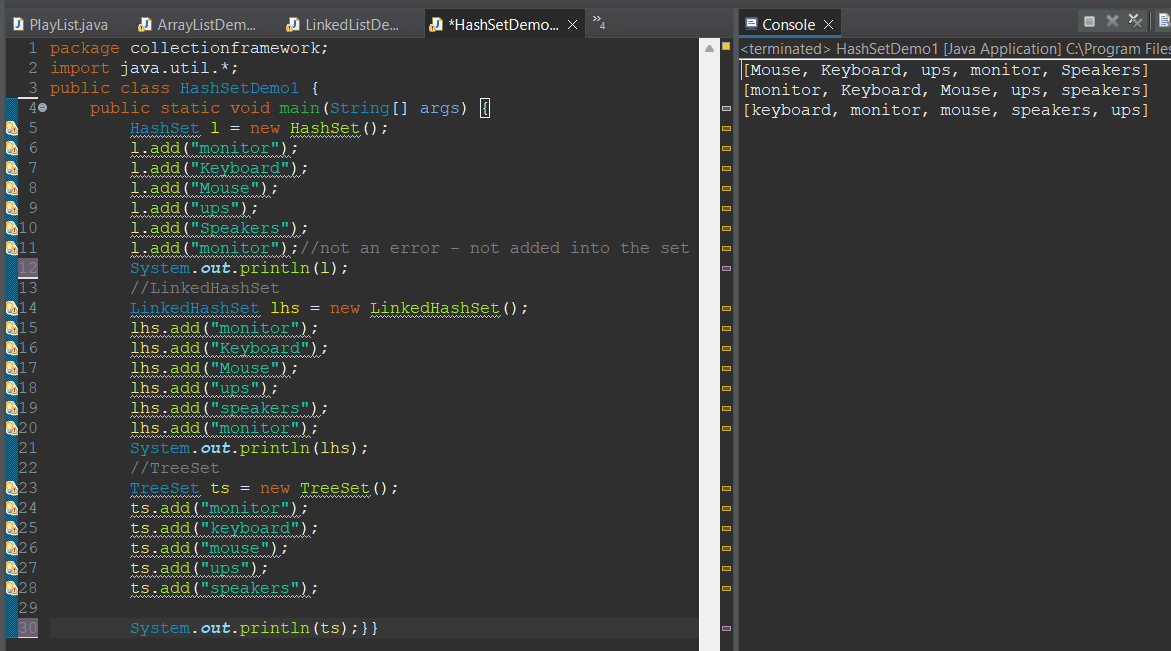
**Treeset(class):**

-it is class which implements the Set interface

-it have synchronized methods

-it belong to legacy class

Note: in tree set all elements should be of same type.



**Maps(interface):**

-map is an interface which is present in java.util.map

-Map is in form of key value pair.

**HashMap(class):**

-Hashmap is a class which implements the Map interface.

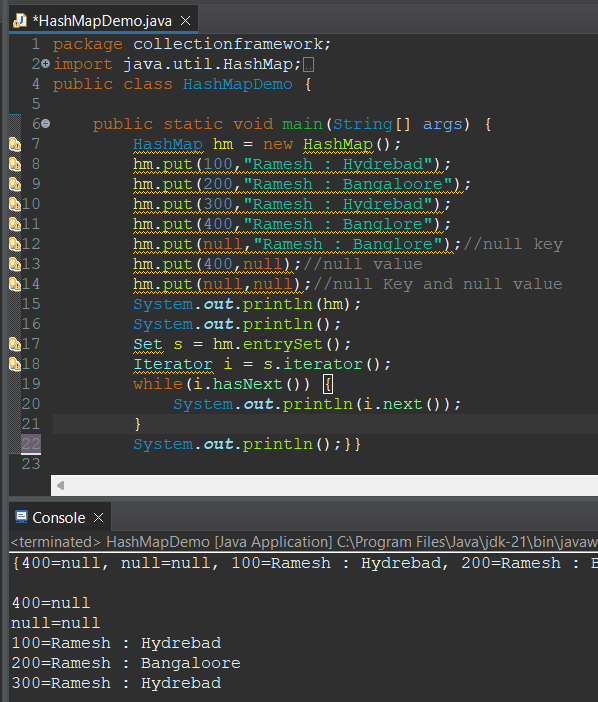
-it is present in java.util.hashmap.

-it is in the form of key value pair.

-it allow null key & nullvalue.

-All methods are not synchronized.

-it is not ordered.



**Hashtable(class):**

-Hashtable is a class which implements map interface

-it is in the form of key value pair.

-it don’t allow null key and null value.

-all methods are synchronized and it is a legacy class

**Linked Hashmap(class**)

-it is a class which implements the map interface .

-it is ordered based on keys.

**Treemap:**

-it is sorted

-all methods are synchronized.

**Iterators:**

------------

-iterators are used to iterate in collection hierarchy.

It is of three types

1)Iterator

2)ListIterator.

3)Enumeration.

**Iterator:**

-It is used to iterate in a collection hierarchy.

-it has remove method.

-it will move in forward direction.

Methods:

-boolean hasnext()

-object next().

**ListIterator:**

-it is used to iterate in a collection hierarchy.

-it has both add and remove methods.

-it will move in forward and backward direction.

-initially it points to center based on user preference it will move to start or end.

Methods:

-boolean hasNext()

-object next()

-boolean hasPrevious()

-object previous()

**Enumeration:**

-it is legacy class

-all the methods are synchronized.

-it will use only to retrive the data not to perform any action.

**Special Guest:**

**Generics:**

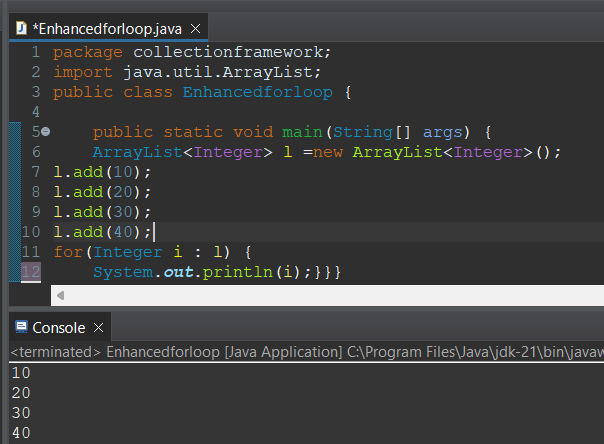
-if we want to store particular type of elements we use generics.

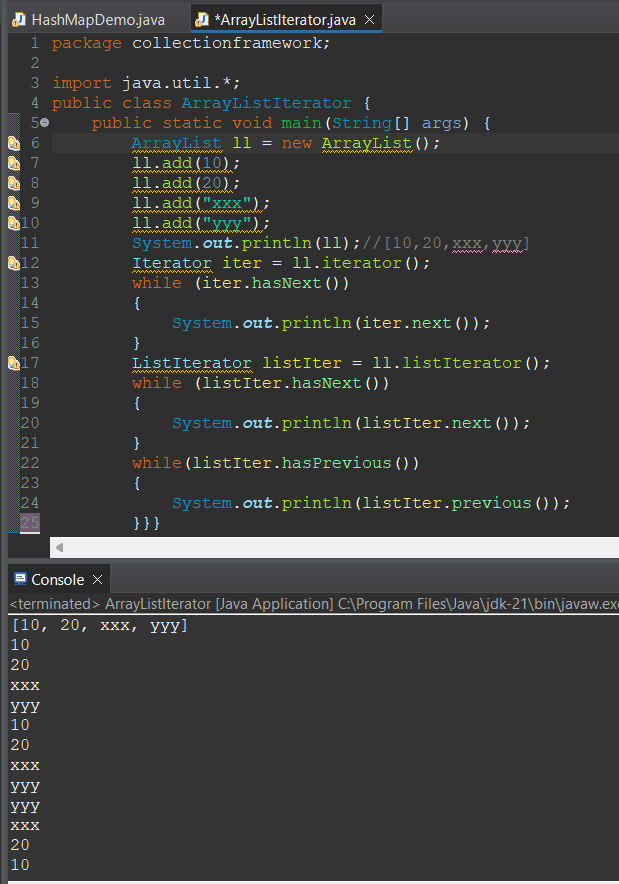
-from jdk 1.5 onwards they introduce generics

**Enhanced for loop:**

-it will used to loop over the collections if we have same type of data.

**-**we use enhanced for loop when we are using collections with generics.





**Arrays vs Collections**

----------------------

- In Arrays the size is fixed where as in Collections the size is not fixed

- In Arrays we can collect only similar type of elements where as in Collections we can collect different type of elements

- In Arrays we can store primitive type as well as reference type of elements where as in collections we can store only reference type of elements

**Collection vs Map**

-----------------

- Collection is a collection of elements where as Map is a collection of key-value pairs

**List vs Set**

-----------

- List is ordered where as Set is unordered

- List allows duplicate elements where as Set does not allow duplicate elements

**ArrayList vs LinkedList**

-----------------------

- In ArrayList the elements are stored in continuous memory locations where as in LinkedList the elements are stored in non-continuous memory locations

- The cost of insert and delete operations are more in ArrayList where as the cost of insert and delete operations are less in LinkedList

**Stack**

-----

- A Stack is collection of elements in the form of Last In First Out (LIFO) operations

**Operations on Stack**

-------------------

- push -> inserts an element into the stack at the top

- pop -> deletes an element from the stack at the top

- peek -> retrives the top element from the stack

**ArrayList vs Vector**

-------------------

- In ArrayList the methods are not synchronized where as in Vector the methods are synchronized

- ArrayList is not thread safe where as Vector is thread safe

**HashSet vs LinkedHashSet**

--------------------------

- In HashSet the elements are not ordered where as in LinkedHashSet the elements are ordered

**HashSet vs TreeSet**

------------------

- HashSet is not sorted where as TreeSet is sorted

**HashMap vs Hashtable**

--------------------

- In HashMap the methods are not synchronized where as in Hashtable the methods are synchronized

- HashMap is not thread safe where as Hashtable is thread safe

**HashMap vs LinkedHashMap**

------------------------

- In HashMap the elements are not ordered where as in LinkedHashMap the elements are ordered based on keys

**HashMap vs TreeMap**

------------------

- In HashMap the elements are not sorted where as in TreeMap the elements are sorted based on keys

**Additional LinkedList methods**

-----------------------------

- void addFirst(Object)

- void addLast(Object)

- Object removeFirst()//it is faster than index values

- Object removeLast()