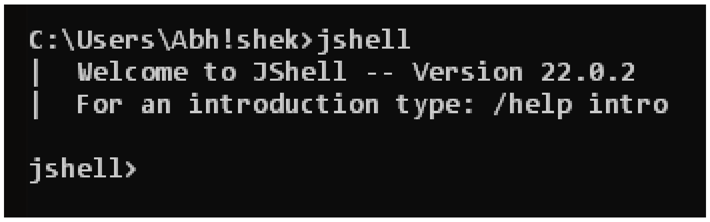
1. GITHUB
2. Youtube Video

When try to create new java file use .java as extension

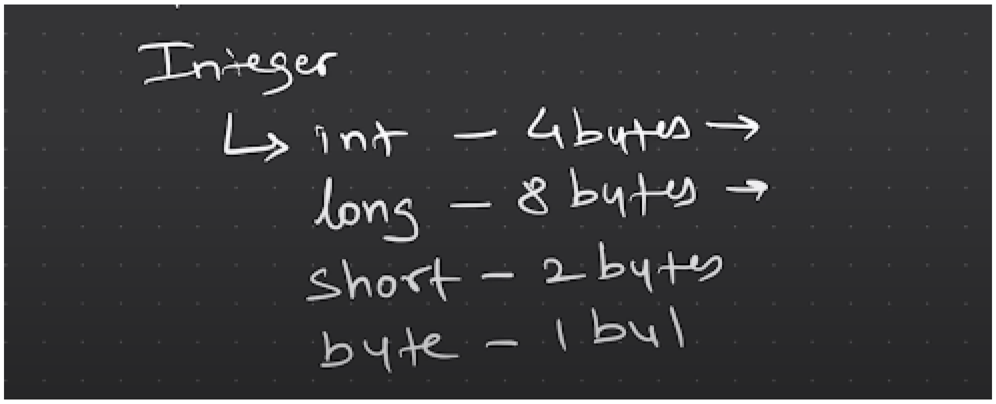
Use Jshell for prompt



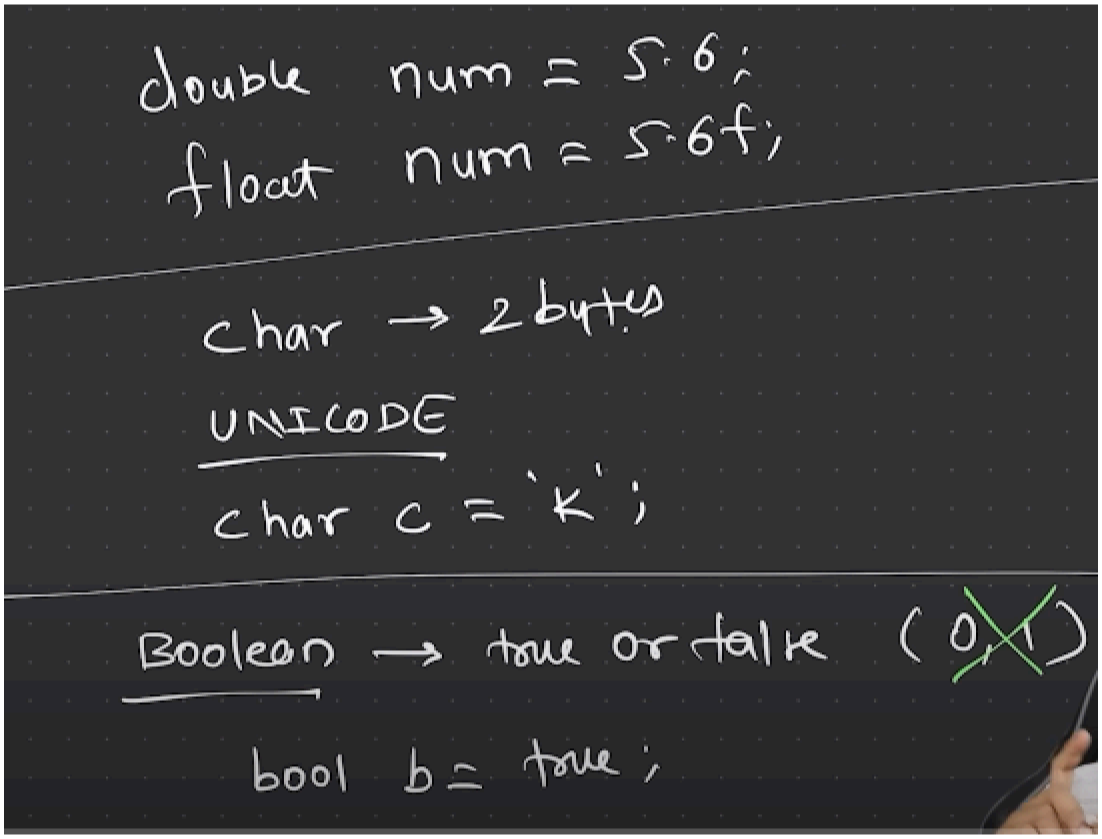
```
C:\Users\Abh!shek>jshell
|  Welcome to JShell -- Version 22.0.2
|  For an introduction type: /help intro

jshell>
```

Java file -> Java compiler -> Byte Code -> JVM
JVM -Java virtual machine
JRE- Java Runtime Env
JAVA is WORA ->> Write once and Run Anywhere

Data type



```
Integer
  ↳ int  – 4bytes →
    long – 8 bytes →
    short – 2 bytes
    byte – 1 byt
```

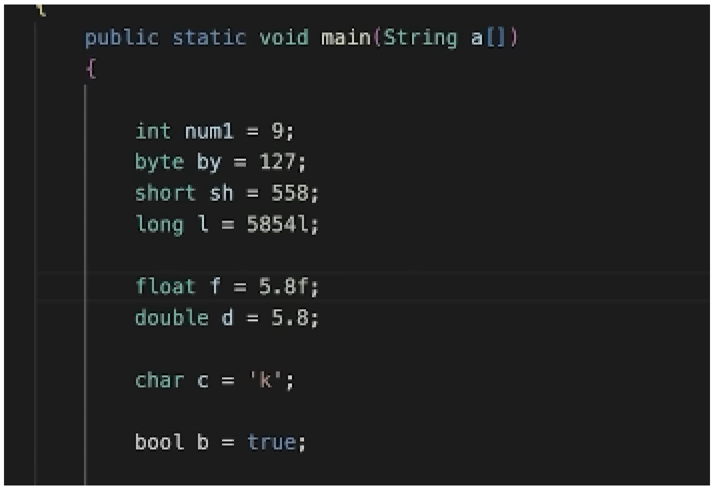

double num = 5.6;
float num = 5.6f;

char → 2 bytes
UNICODE
char c = 'k';

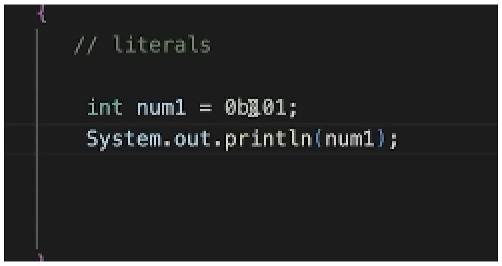
Boolean → true or false (~~0, 1~~)

bool b = true;

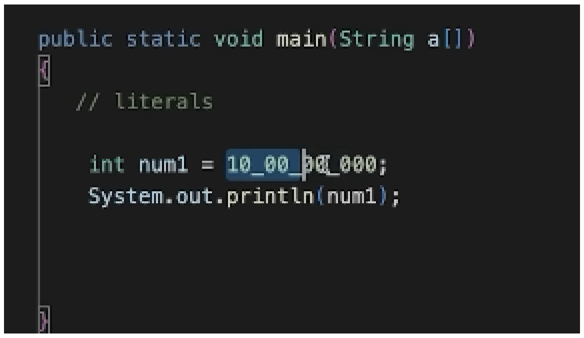
Primitive data type


```
    public static void main(String a[])
    {

        int num1 = 9;
        byte by = 127;
        short sh = 558;
        long l = 5854l;

        float f = 5.8f;
        double d = 5.8;

        char c = 'k';

        bool b = true;
```

Literals binary format

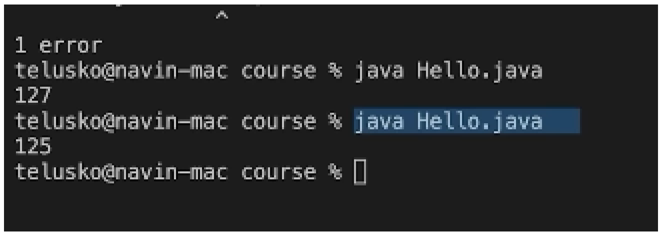

```
    // literals

     int num1 = 0b101;
     System.out.println(num1);
```

0x7e

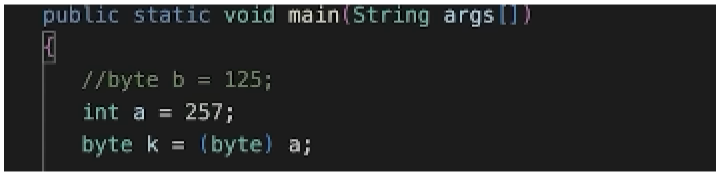


```
public static void main(String a[])
{
    // literals

     int num1 = 10_00_00_000;
     System.out.println(num1);



}
```

Type Conversion and type casting

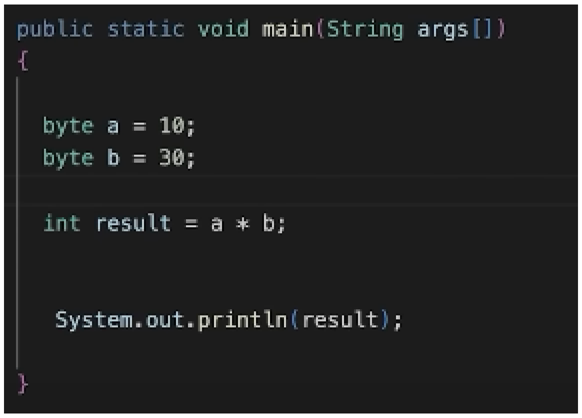


```
1 error
telusko@navin-mac course % java Hello.java
127
telusko@navin-mac course % java Hello.java
125
telusko@navin-mac course % 
```



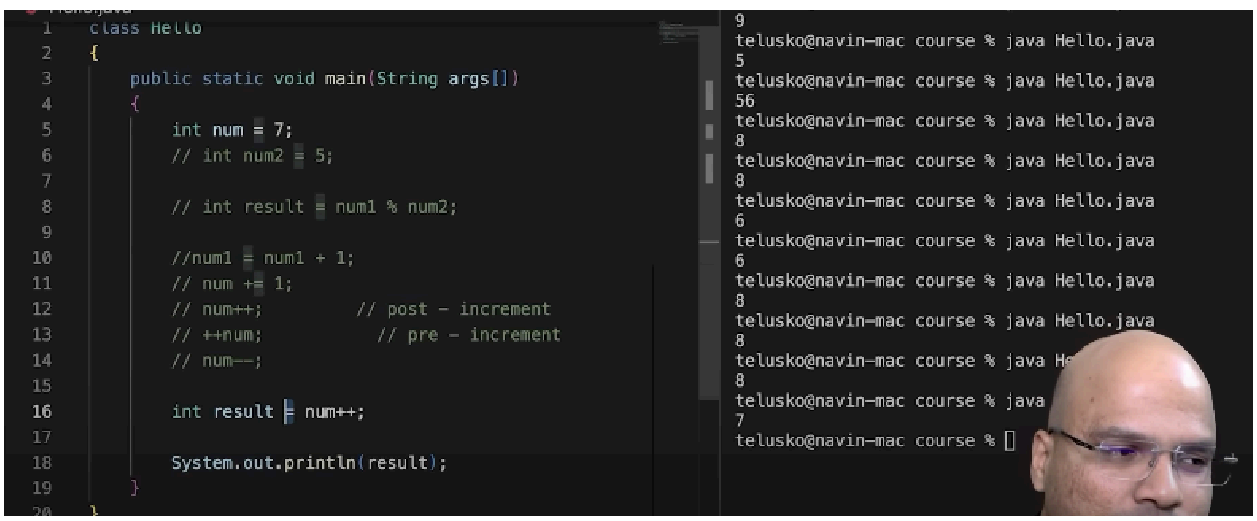
```
public static void main(String args[])
{
    //byte b = 125;
    int a = 257;
    byte k = (byte) a;
```

Type prompting



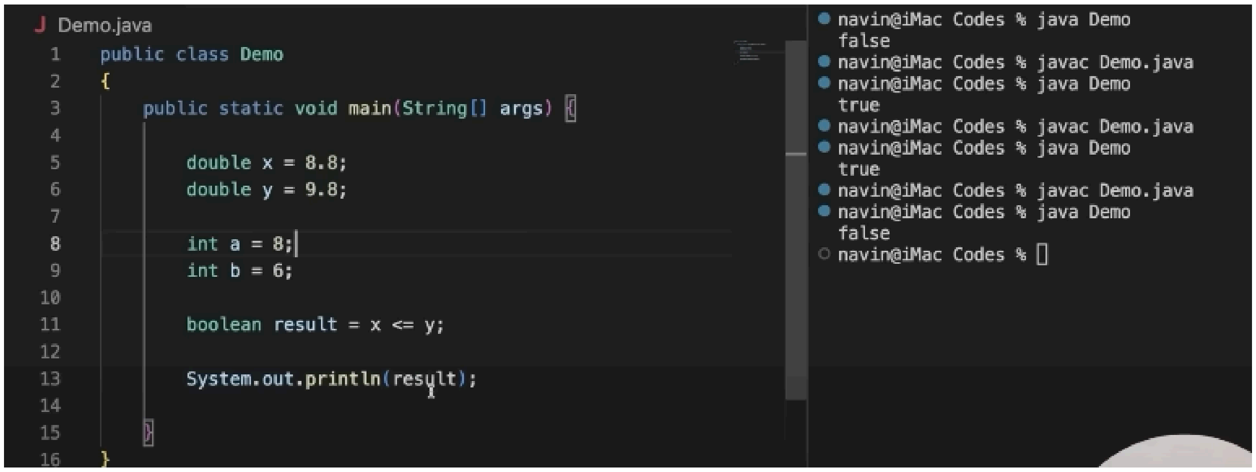
```
public static void main(String args[])
{

  byte a = 10;
  byte b = 30;

  int result = a * b;


   System.out.println(result);

}
```

Arithmetic Operator



```java
class Hello
{
    public static void main(String args[])
    {
        int num = 7;
        // int num2 = 5;

        // int result = num1 % num2;

        //num1 = num1 + 1;
        // num += 1;
        // num++;          // post - increment
        // ++num;            // pre - increment
        // num--;

        int result = num++;

        System.out.println(result);
    }
}
```

```
9
telusko@navin-mac course % java Hello.java
5
telusko@navin-mac course % java Hello.java
56
telusko@navin-mac course % java Hello.java
8
telusko@navin-mac course % java Hello.java
8
telusko@navin-mac course % java Hello.java
6
telusko@navin-mac course % java Hello.java
6
telusko@navin-mac course % java Hello.java
8
telusko@navin-mac course % java Hello.java
8
telusko@navin-mac course % java He
8
telusko@navin-mac course % java
7
telusko@navin-mac course % 
```

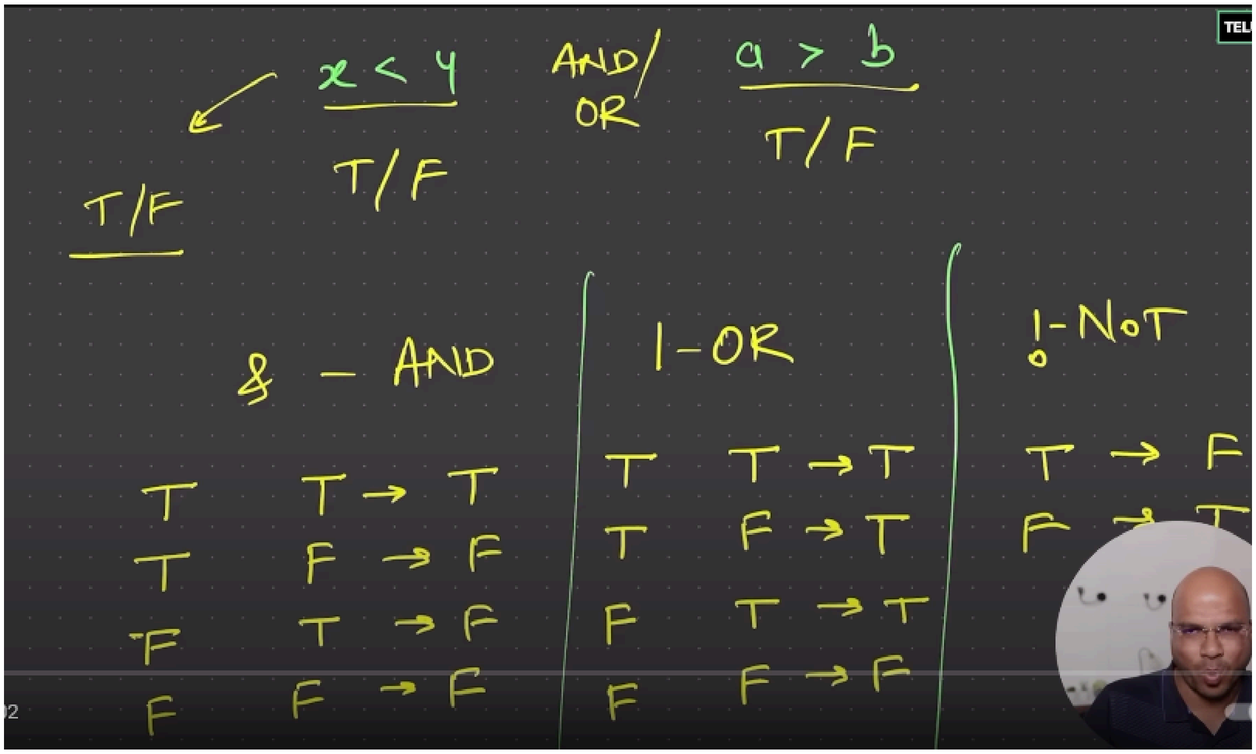


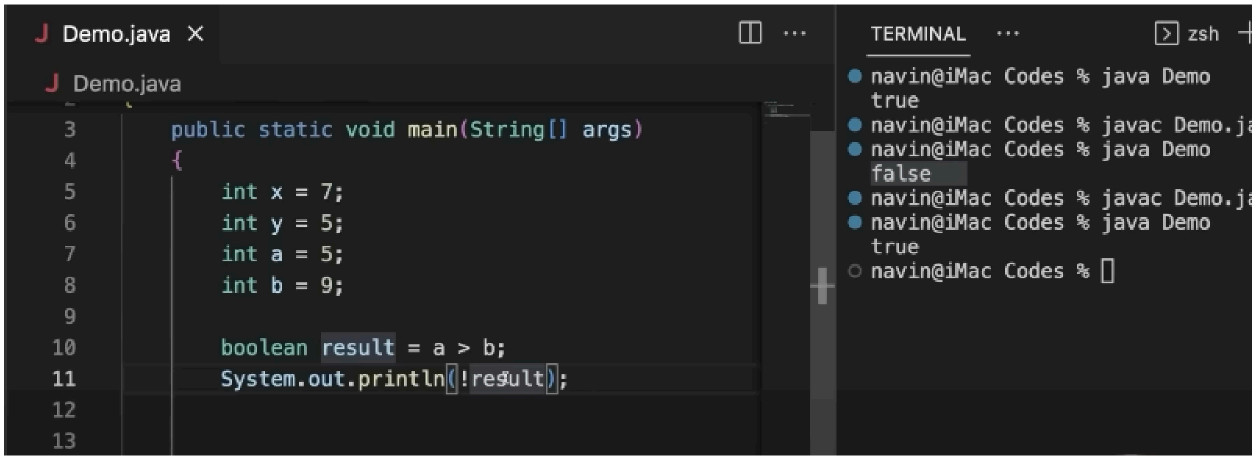
Demo.java

```java
public class Demo
{
    public static void main(String[] args) {

        double x = 8.8;
        double y = 9.8;

        int a = 8;
        int b = 6;

        boolean result = x <= y;

        System.out.println(result);

    }
}
```

```
navin@iMac Codes % java Demo
false
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
true
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
true
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
false
navin@iMac Codes % 
```

Logical operator

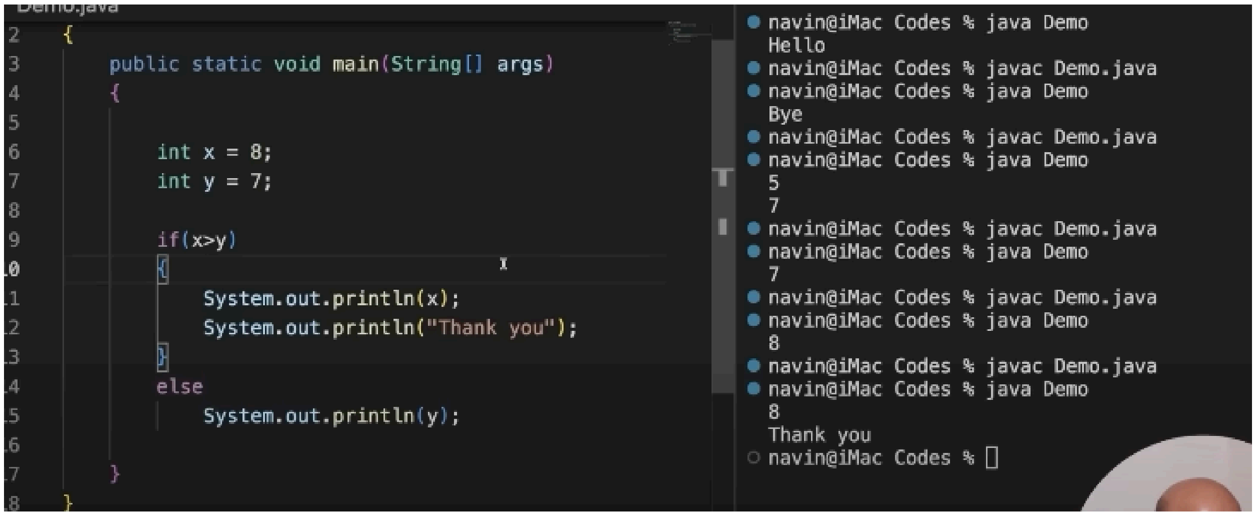




```java
    public static void main(String[] args)
    {
        int x = 7;
        int y = 5;
        int a = 5;
        int b = 9;

        boolean result = a > b;
        System.out.println(!result);


```

```
navin@iMac Codes % java Demo
true
navin@iMac Codes % javac Demo.ja
navin@iMac Codes % java Demo
false
navin@iMac Codes % javac Demo.ja
navin@iMac Codes % java Demo
true
navin@iMac Codes %
```

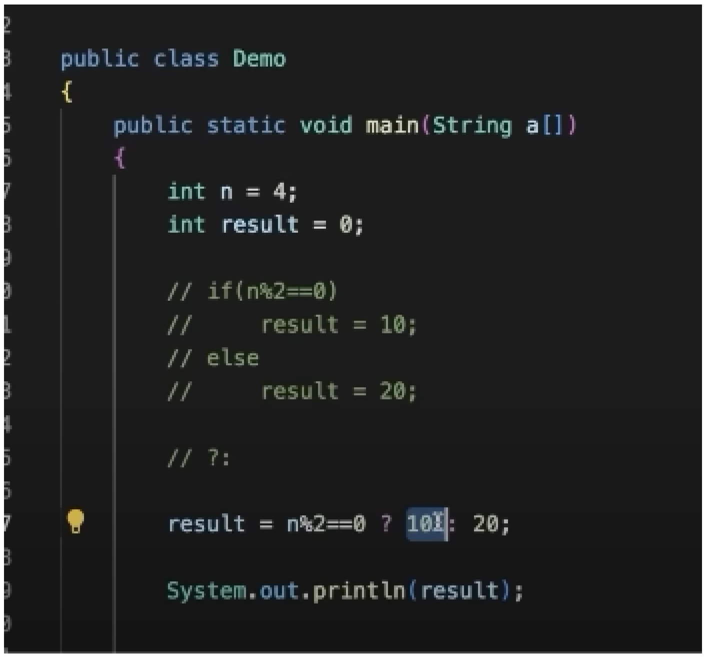
Conditional Statement



```
{
    public static void main(String[] args)
    {

        int x = 8;
        int y = 7;

        if(x>y)
        {
            System.out.println(x);
            System.out.println("Thank you");
        }
        else
            System.out.println(y);

    }
}
```

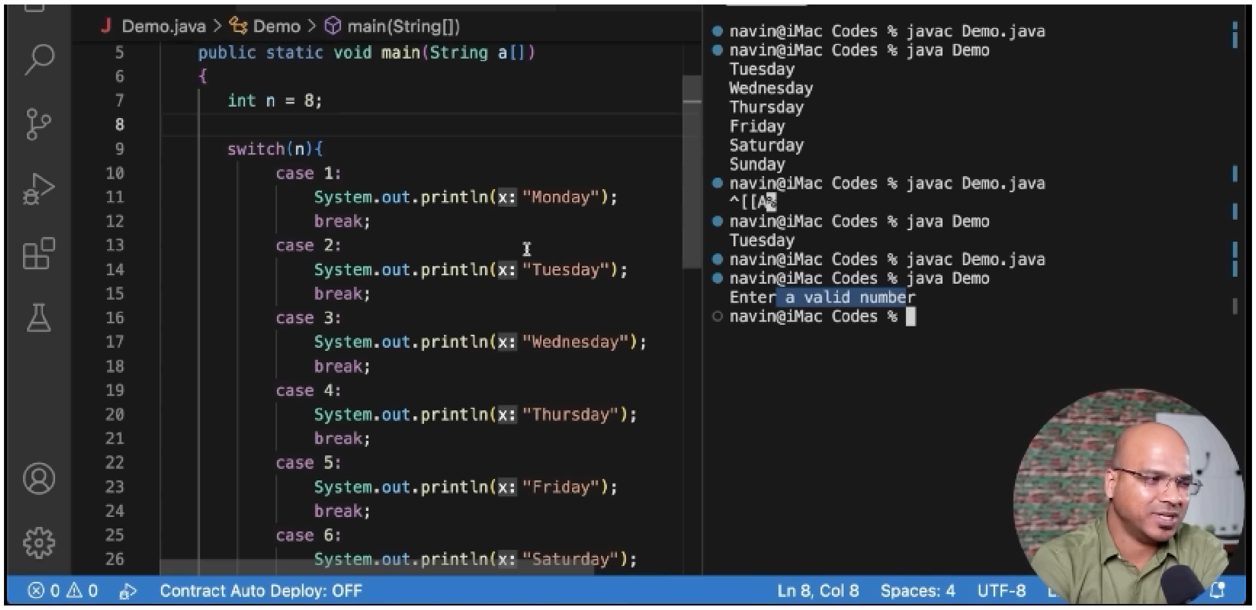
```
navin@iMac Codes % java Demo
Hello
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
Bye
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
5
7
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
7
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
8
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
8
Thank you
navin@iMac Codes % 
```

Ternary operator



```
public class Demo
{
    public static void main(String a[])
    {
        int n = 4;
        int result = 0;

        // if(n%2==0)
        //     result = 10;
        // else
        //     result = 20;

        // ?:

        result = n%2==0 ? 10 : 20;

        System.out.println(result);
```

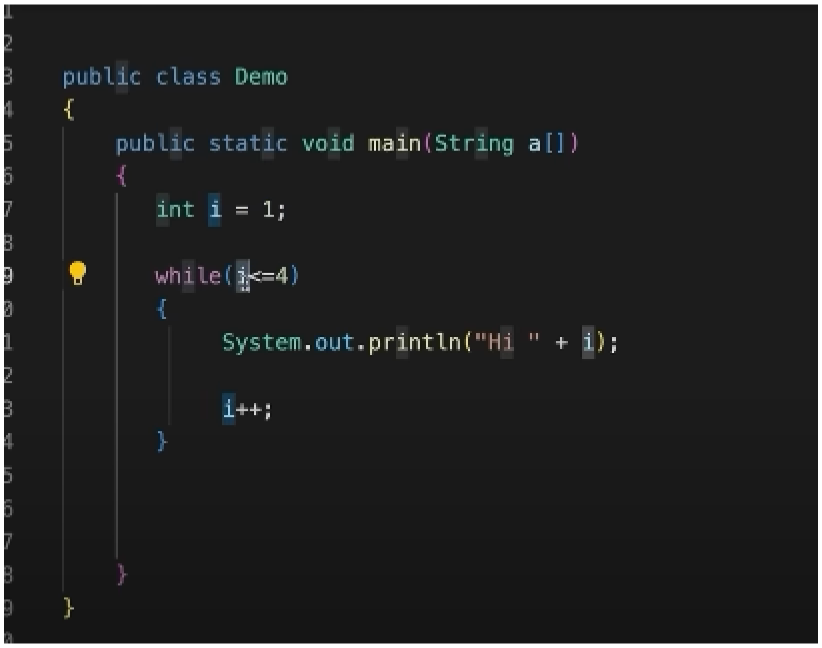
Switch statement



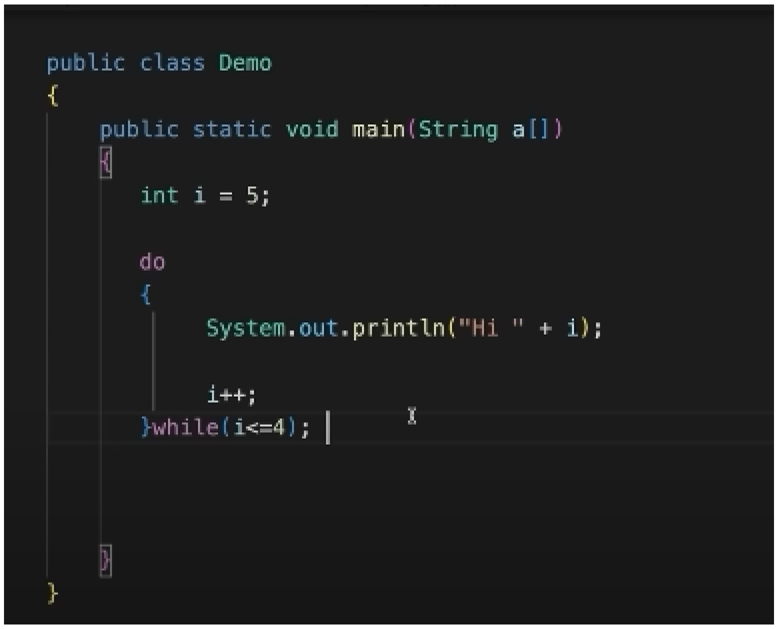
```java
    public static void main(String a[])
    {
        int n = 8;

        switch(n){
            case 1:
                System.out.println(x: "Monday");
                break;
            case 2:
                System.out.println(x: "Tuesday");
                break;
            case 3:
                System.out.println(x: "Wednesday");
                break;
            case 4:
                System.out.println(x: "Thursday");
                break;
            case 5:
                System.out.println(x: "Friday");
                break;
            case 6:
                System.out.println(x: "Saturday");
```

```
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
navin@iMac Codes % javac Demo.java
^[[A
navin@iMac Codes % java Demo
Tuesday
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
Enter a valid number
navin@iMac Codes %
```

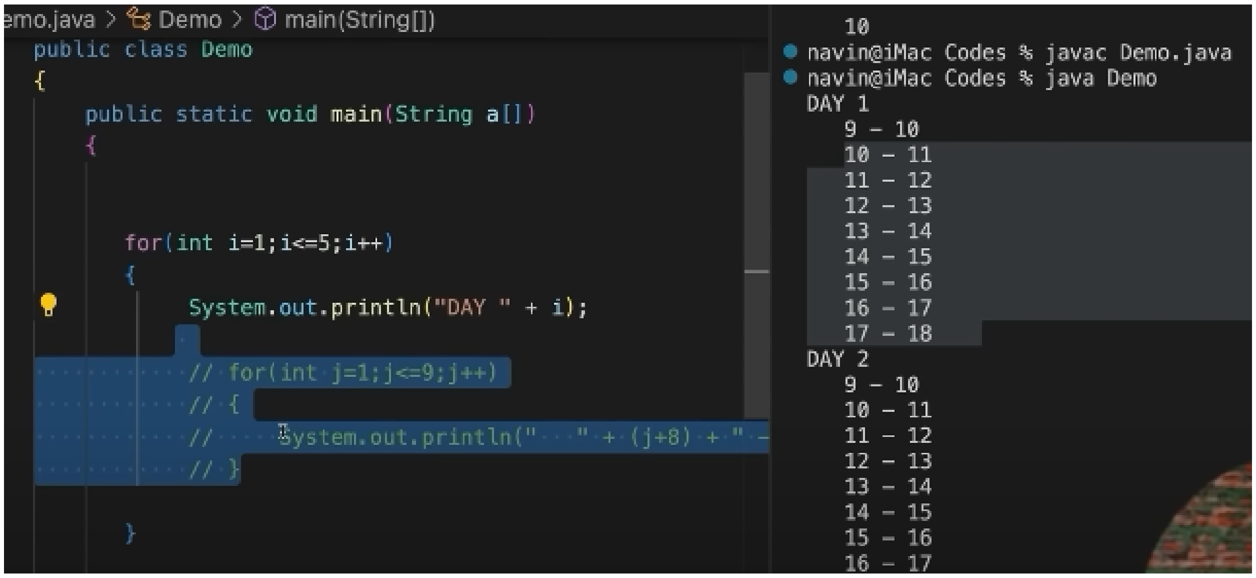
Conditional loop



```java
public class Demo
{
    public static void main(String a[])
    {
        int i = 1;

        while(i<=4)
        {
            System.out.println("Hi " + i);

            i++;
        }



    }
}
```



```
public class Demo
{
    public static void main(String a[])
    {
        int i = 5;

        do
        {
            System.out.println("Hi " + i);

            i++;
        }while(i<=4);


    }
}
```



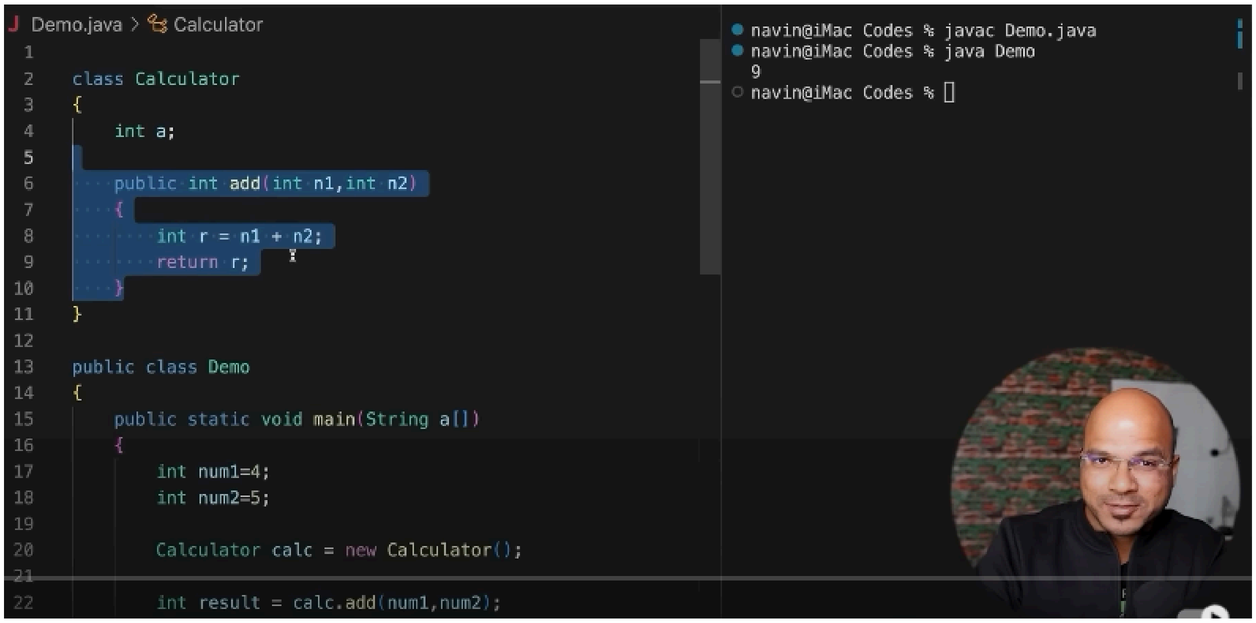
```
emo.java > Demo > main(String[])
public class Demo
{
    public static void main(String a[])
    {


        for(int i=1;i<=5;i++)
        {
            System.out.println("DAY " + i);

            // for(int j=1;j<=9;j++)
            // {
            //     System.out.println("   " + (j+8) + " -
            // }

        }
```

```
    10
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
DAY 1
    9 - 10
    10 - 11
    11 - 12
    12 - 13
    13 - 14
    14 - 15
    15 - 16
    16 - 17
    17 - 18
DAY 2
    9 - 10
    10 - 11
    11 - 12
    12 - 13
    13 - 14
    14 - 15
    15 - 16
    16 - 17
```

Concepts

Object oriented programing

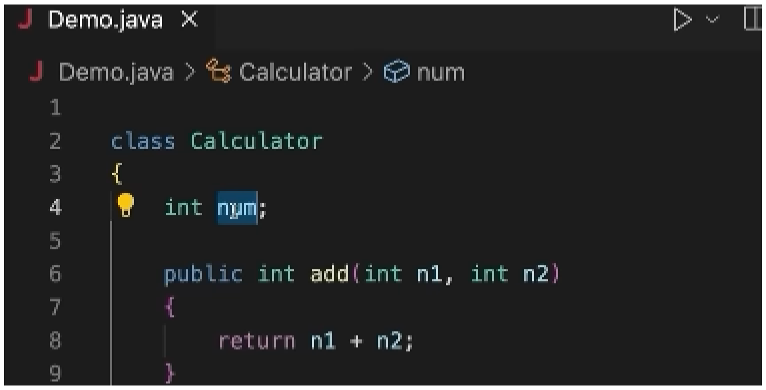
Methods and objects



```
Demo.java > Calculator

class Calculator
{
    int a;

    public int add(int n1,int n2)
    {
        int r = n1 + n2;
        return r;
    }
}

public class Demo
{
    public static void main(String a[])
    {
        int num1=4;
        int num2=5;

        Calculator calc = new Calculator();

        int result = calc.add(num1,num2);
```

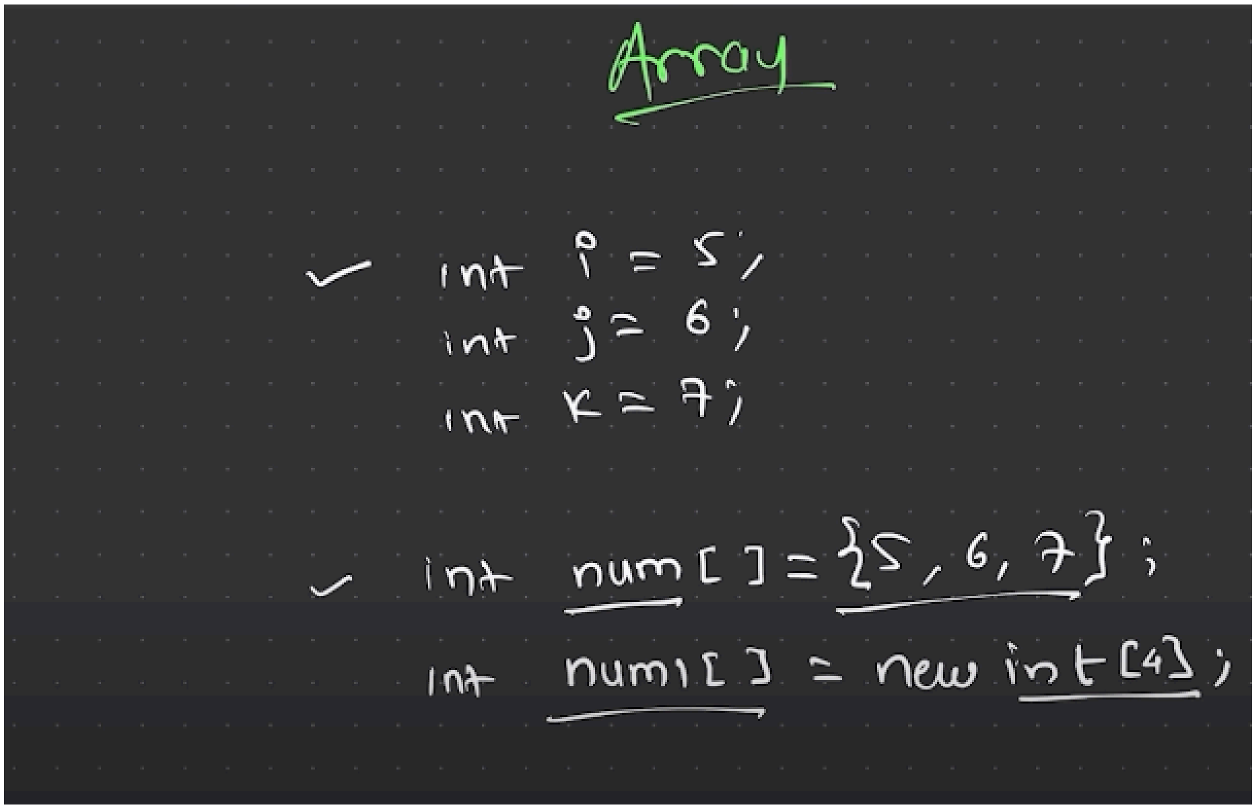
```
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
9
navin@iMac Codes % 
```

- NUm is a instance variable



```
Demo.java > Calculator > num

class Calculator
{
    int num;

    public int add(int n1, int n2)
    {
        return n1 + n2;
    }
```

Array

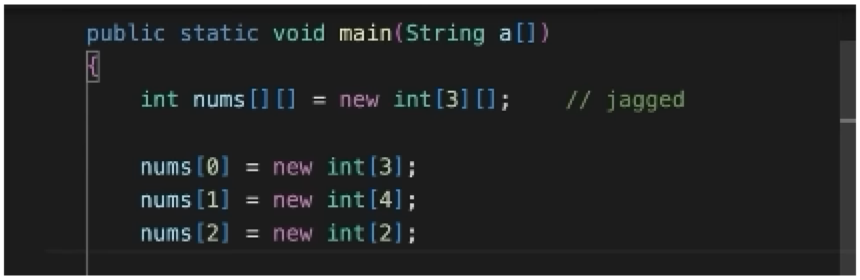


Array

✓ int i = 5;
int j = 6;
int k = 7;

✓ int num[] = {5, 6, 7};
int num1[] = new int[4];

Jagged Array

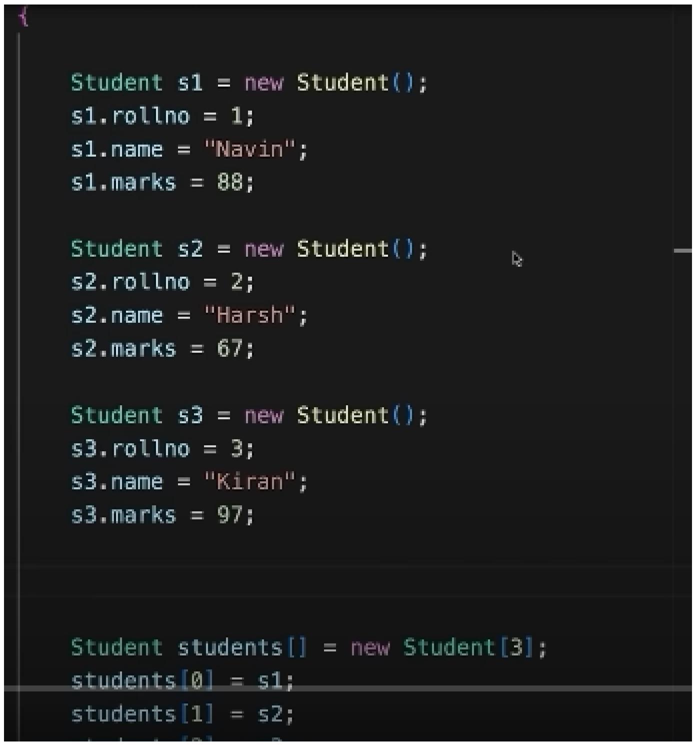


```
public static void main(String a[])
{
    int nums[][] = new int[3][];    // jagged

    nums[0] = new int[3];
    nums[1] = new int[4];
    nums[2] = new int[2];
```

Drawbacks of array

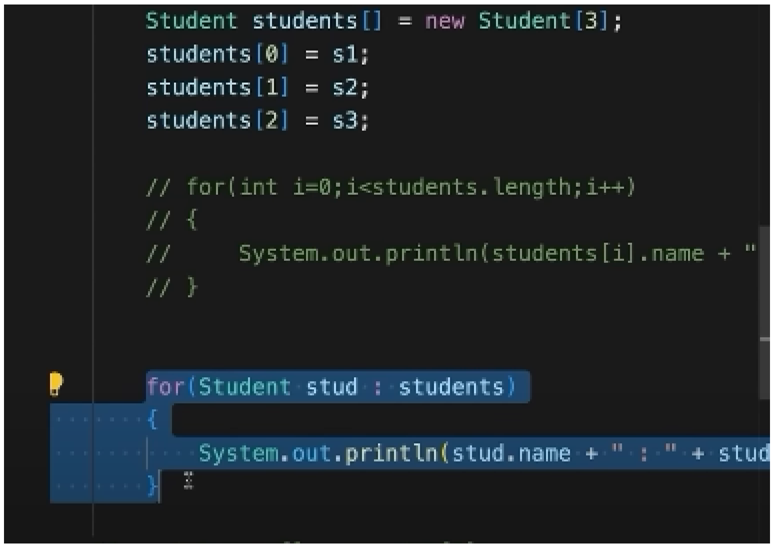
Cant increase size

String obj



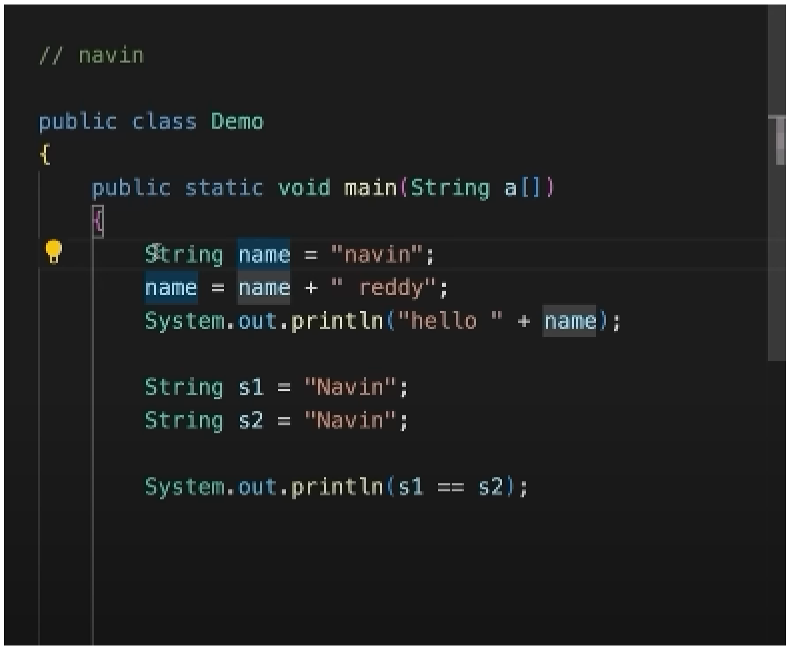
```
{

    Student s1 = new Student();
    s1.rollno = 1;
    s1.name = "Navin";
    s1.marks = 88;

    Student s2 = new Student();
    s2.rollno = 2;
    s2.name = "Harsh";
    s2.marks = 67;

    Student s3 = new Student();
    s3.rollno = 3;
    s3.name = "Kiran";
    s3.marks = 97;



    Student students[] = new Student[3];
    students[0] = s1;
    students[1] = s2;
```

<u>Enhanced for loop or for each loop</u>



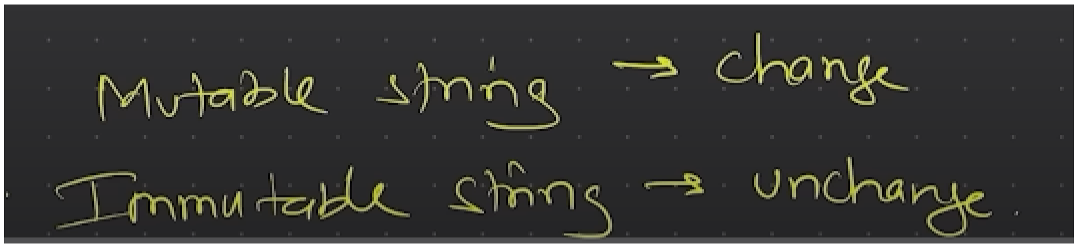
```
        Student students[] = new Student[3];
        students[0] = s1;
        students[1] = s2;
        students[2] = s3;

        // for(int i=0;i<students.length;i++)
        // {
        //     System.out.println(students[i].name + "
        // }


        for(Student stud : students)
        {
            System.out.println(stud.name + " : " + stud
        }
```

Working with stream(string)



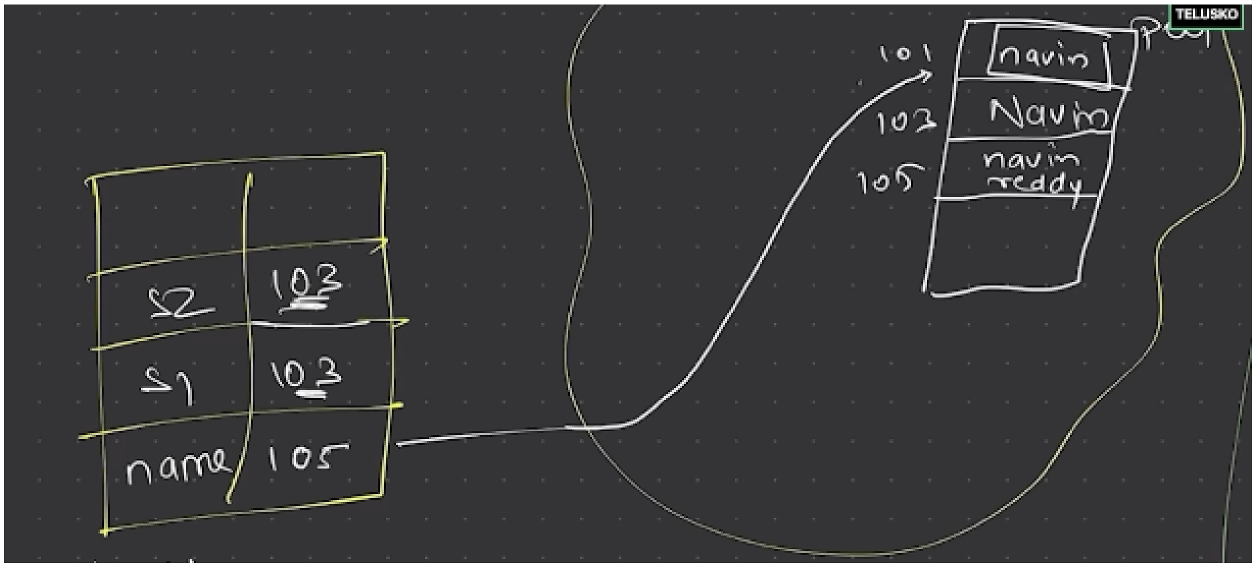
```java
// navin

public class Demo
{
    public static void main(String a[])
    {
        String name = "navin";
        name = name + " reddy";
        System.out.println("hello " + name);

        String s1 = "Navin";
        String s2 = "Navin";

        System.out.println(s1 == s2);
```

Here creating only one object but multiple references



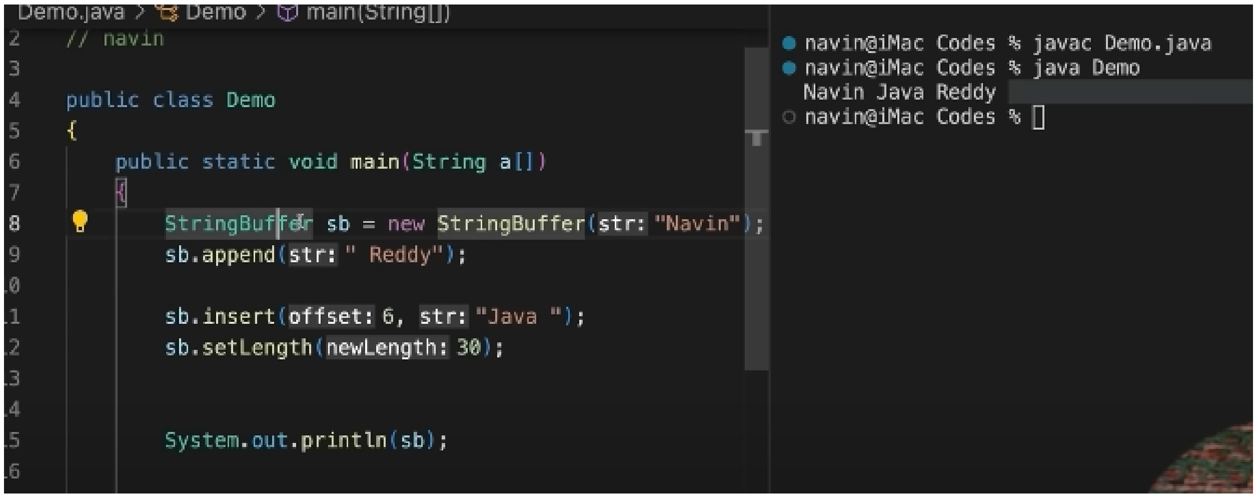
Mutable string → change

Immutable string → unchange.



Mutable string are :
String buffer
String Builder

String buffer



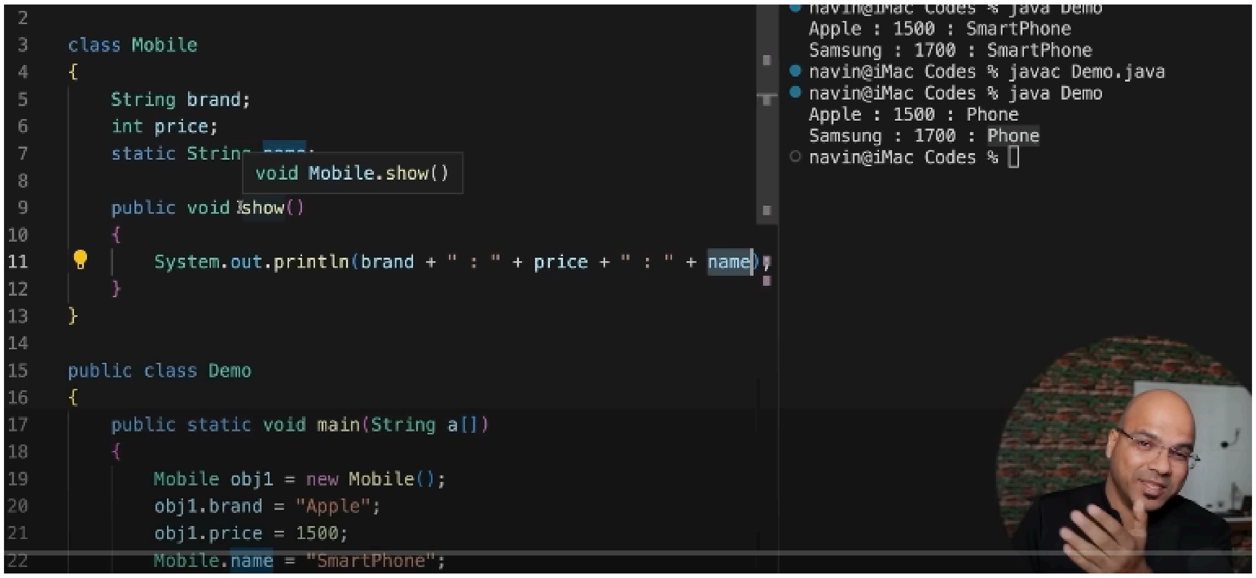
```
Demo.java > Demo > main(String[])
// navin

public class Demo
{
    public static void main(String a[])
    {
        StringBuffer sb = new StringBuffer(str: "Navin");
        sb.append(str: " Reddy");

        sb.insert(offset: 6, str: "Java ");
        sb.setLength(newLength: 30);


        System.out.println(sb);
```

```
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
Navin Java Reddy
navin@iMac Codes %
```

NOTE : String Buffer is tread safe and String builder is not

Static Keywork

Diierence b/w



```
class Mobile
{
    String brand;
    int price;
    static String name;

    public void show()
    {
        System.out.println(brand + " : " + price + " : " + name);
    }
}

public class Demo
{
    public static void main(String a[])
    {
        Mobile obj1 = new Mobile();
        obj1.brand = "Apple";
        obj1.price = 1500;
        Mobile.name = "SmartPhone";
```

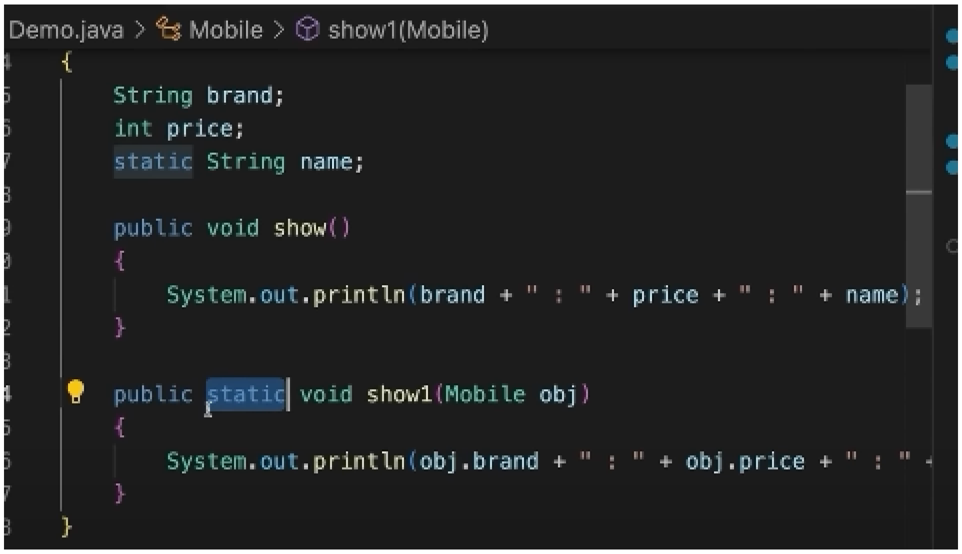
```
navin@iMac Codes % java Demo
Apple : 1500 : SmartPhone
Samsung : 1700 : SmartPhone
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
Apple : 1500 : Phone
Samsung : 1700 : Phone
navin@iMac Codes %
```

- [ ] Static keyword will call by class name not by object (reference)

STATIC METHOD

Only Static Variable can used in Static Method

Calling Non Static inside Static Method



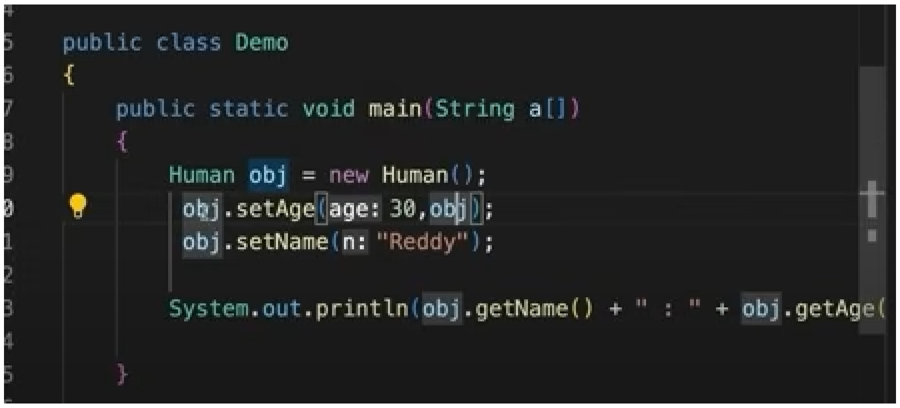
Demo.java > Mobile > show1(Mobile)

```java
{
    String brand;
    int price;
    static String name;

    public void show()
    {
        System.out.println(brand + " : " + price + " : " + name);
    }

    public static void show1(Mobile obj)
    {
        System.out.println(obj.brand + " : " + obj.price + " : "
    }
}
```

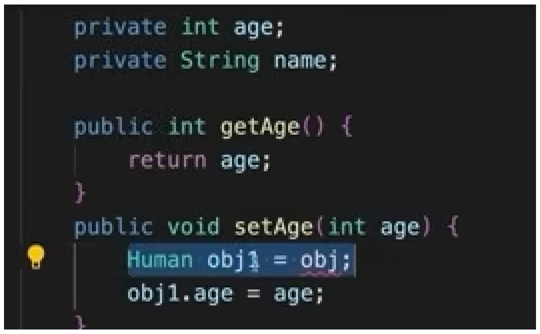
Opps

Encapsulation 👍

This Keyword



```
public class Demo
{
    public static void main(String a[])
    {
        Human obj = new Human();
         obj.setAge(age: 30,obj);
         obj.setName(n: "Reddy");

        System.out.println(obj.getName() + " : " + obj.getAge(

    }
```



```
    private int age;
    private String name;

    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        Human obj1 = obj;
        obj1.age = age;
```

Alternative but not in use
Mainly used **THIS Keyworld**

Constructor

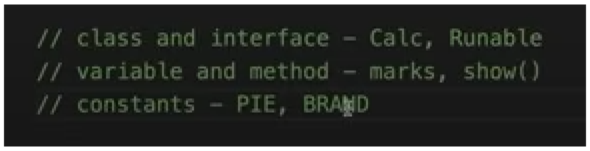
- This used whenever object created to assigning the values.
- Also used when required DB connections 👍
- Super is always exist in Constructor (doesn't matters it declared) 👍

Super Keyword

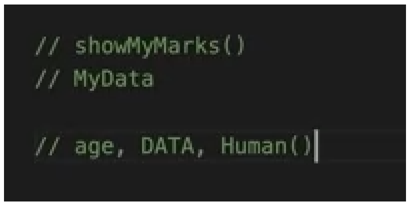
- ☐ Every class in java extend Object class 👍
    - ☐ Like : class A extends Object

Naming convention

- Camel Casing

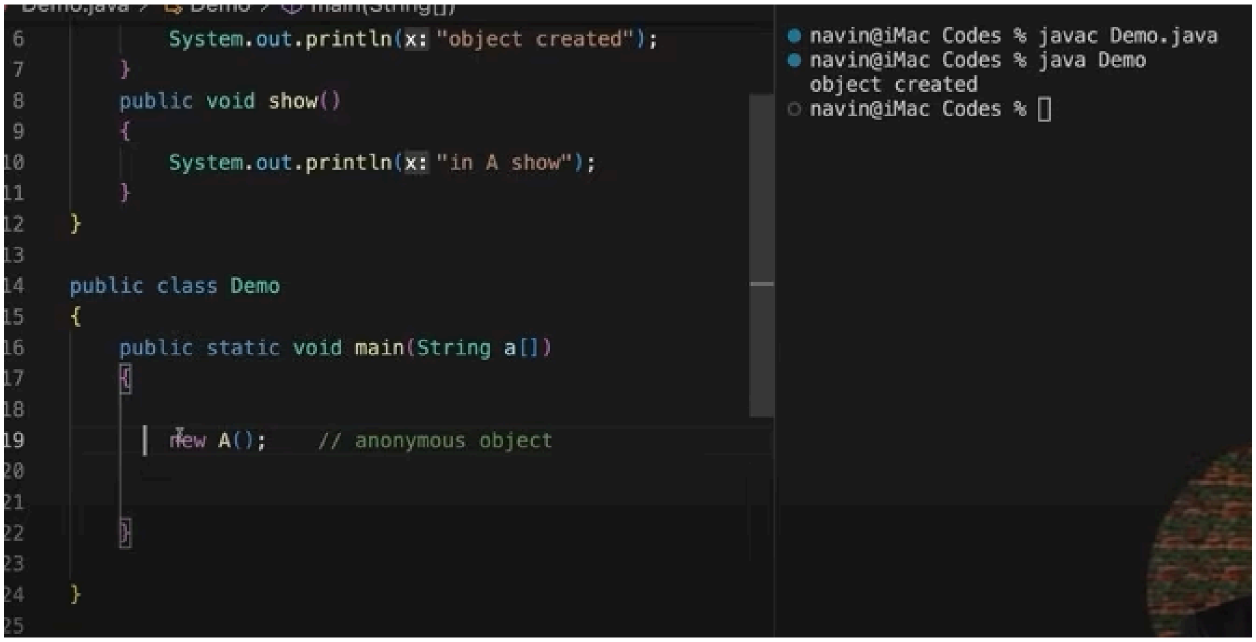
- 

```
// class and interface - Calc, Runable
// variable and method - marks, show()
// constants - PIE, BRAND
```



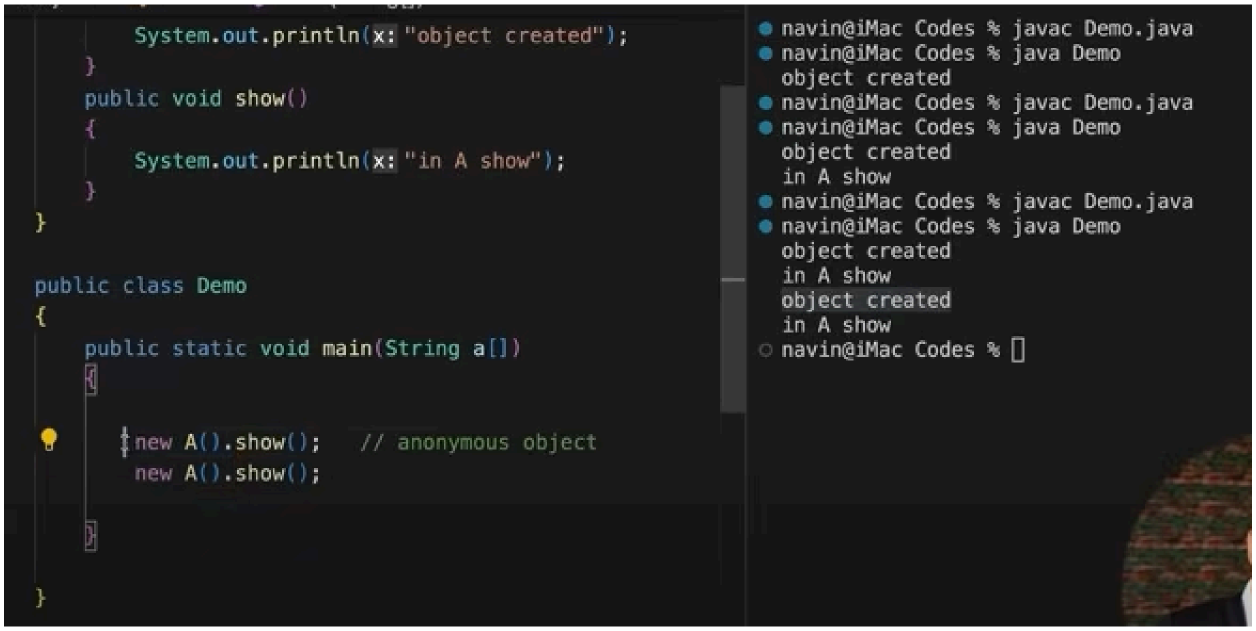
```
// showMyMarks()
// MyData

// age, DATA, Human()
```

Anonymous object



```
        System.out.println(x: "object created");
    }
    public void show()
    {
        System.out.println(x: "in A show");
    }
}

public class Demo
{
    public static void main(String a[])
    {

        new A();     // anonymous object


    }

}
```

```
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
object created
navin@iMac Codes % 
```



```
        System.out.println(x: "object created");
    }
    public void show()
    {
        System.out.println(x: "in A show");
    }
}

public class Demo
{
    public static void main(String a[])
    {

        new A().show();   // anonymous object
        new A().show();

    }

}
```

```
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
object created
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
object created
in A show
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
object created
in A show
object created
in A show
navin@iMac Codes % 
```

Inheritance

- [ ] To inherit use ***extends*** keyword
- [ ] **Multiple inheritanc**e not support by JAVA ~ Support by C++

- **Ambiguity –** if both parents have same name function then child object will confused .thats called…
- **Multi label inheritance** will fix by **interfaces**

Method Overriding

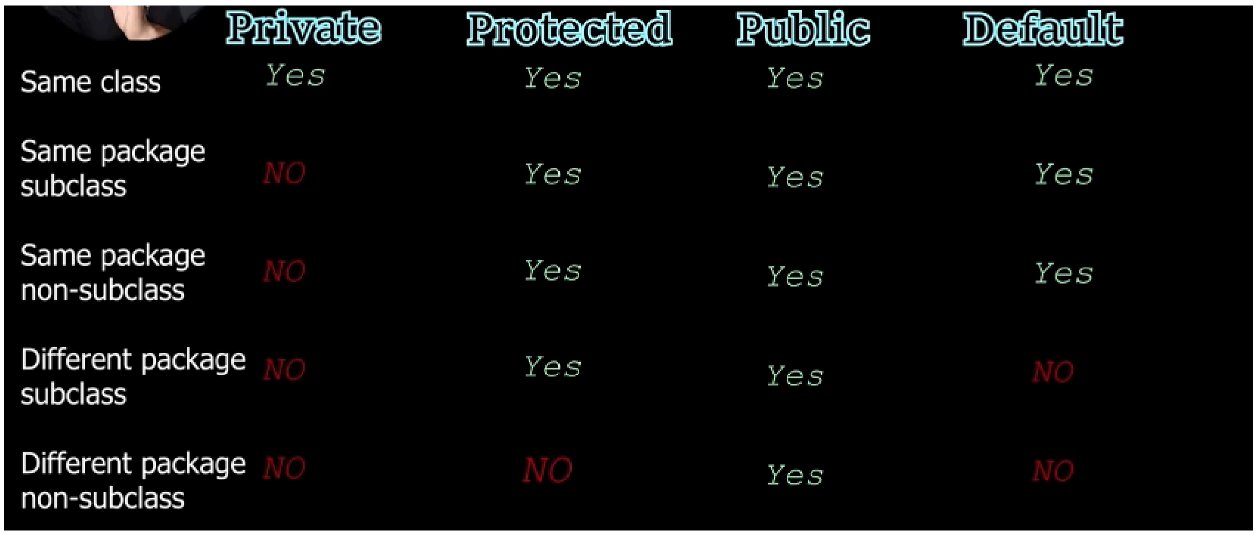
```
class A1
{
   void show ()
   {
       System.out.println("A : ");
   }
}

class B1 extends A1
{
   void show ()
   {
       System.out.println("B : ");
   }
```

Package

- TO create Package just write **package tool** and IDE will create automatically
- To use any class from any location just Use **Impor**t *path* (**tools.calc**).
- To use a file then use Start*() for remove error LIKE :* ***package tools.****

Access Modifier

- To use from outside of package use **public**



	Private	Protected	Public	Default
Same class	Yes	Yes	Yes	Yes
Same package subclass	NO	Yes	Yes	Yes
Same package non-subclass	NO	Yes	Yes	Yes
Different package subclass	NO	Yes	Yes	NO
Different package non-subclass	NO	NO	Yes	NO

Polymorphism

- Two Types
    - Compile time Morphism (acchive at Overloading)
    - Run time morphism (acchive at Override)
- Define – Multi Behavior

Dynamic Method Dispatch

```
// This is Runtime polymorphism
A11 obj=new A11();

obj.show();

obj=new B11();
obj.show();
```

```
obj=new C11();
obj.show();
```

O/p

```
A : show
B : show
C : show

Process finished with exit code 0
```

Final Keyword

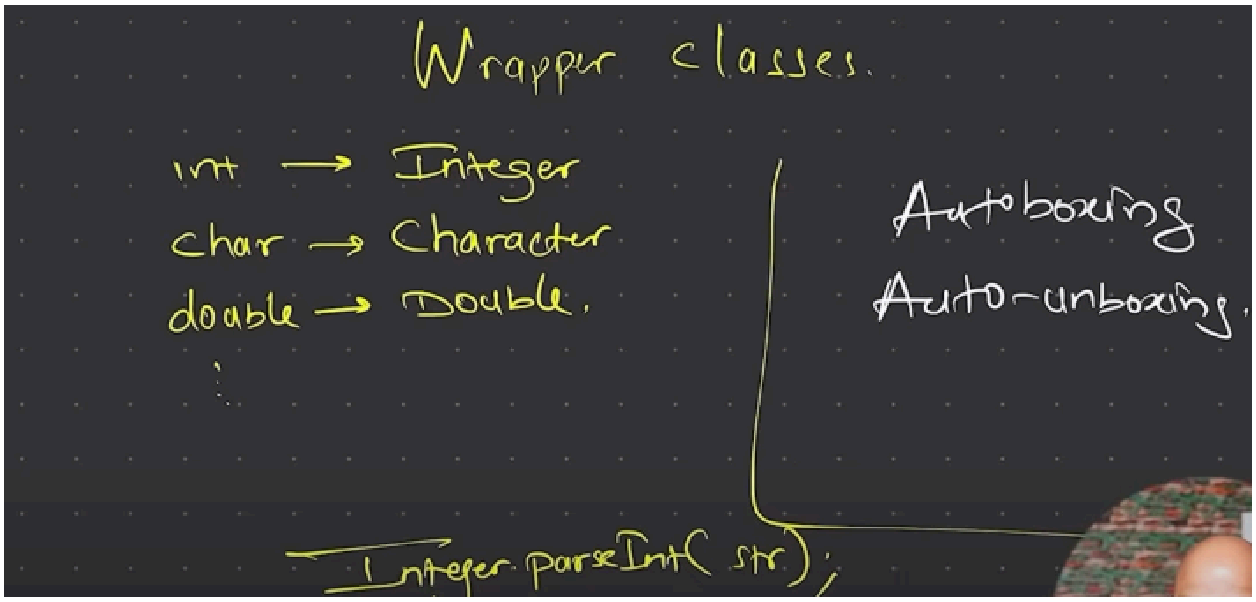
- Use with Variable ,Method and Class
- To stop inherit of class in java

Object Class equals toString hashcode

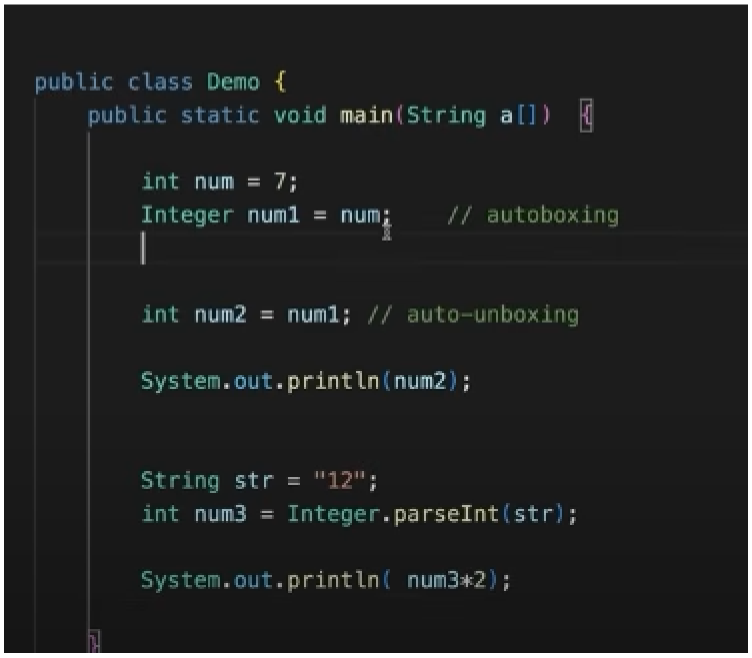
Right click -> generate -> equals and hashcode

Upcasting and Downcasting

Wrapper class



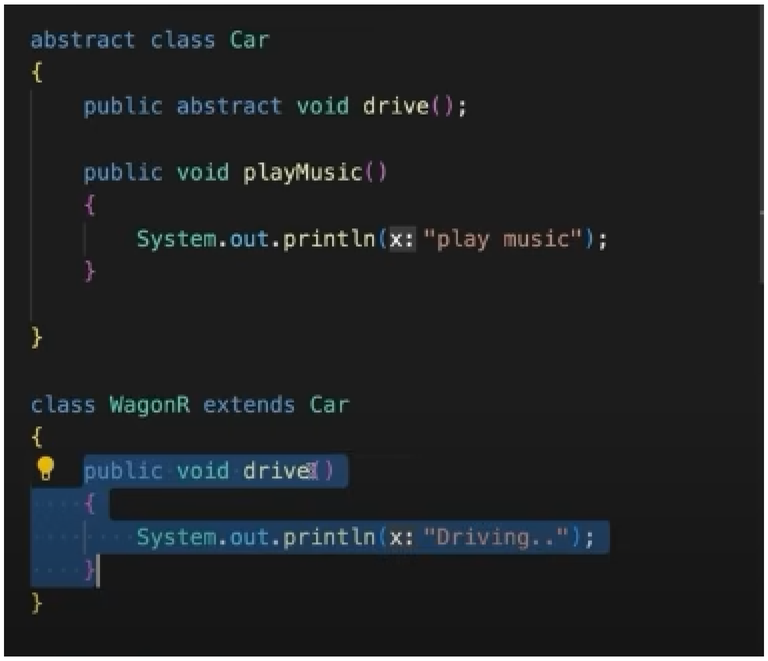
Autoboxing and Autounboxing



```
public class Demo {
    public static void main(String a[])  {

        int num = 7;
        Integer num1 = num;    // autoboxing


        int num2 = num1; // auto-unboxing

        System.out.println(num2);


        String str = "12";
        int num3 = Integer.parseInt(str);

        System.out.println( num3*2);

    }
```

Abstract Keyword

- Can not create object of Abstract class

- 

```
abstract class Car
{
    public abstract void drive();

    public void playMusic()
    {
        System.out.println(x: "play music");
    }

}

class WagonR extends Car
{
    public void drive()
    {
        System.out.println(x: "Driving..");
    }
}
```

- Concrete class is called for non abstract class and you can create object of this
- No need to define Abstract method inside abstract class

Inner Class

- Outer class can not make it as Static it will give error
- You can inner side Static keyword to remove object LIKE
    - `A4.C4 c4=new A4.C4();`
    - Otherwise you can write like
        - `A4.B4 b4=a4.new B4();`

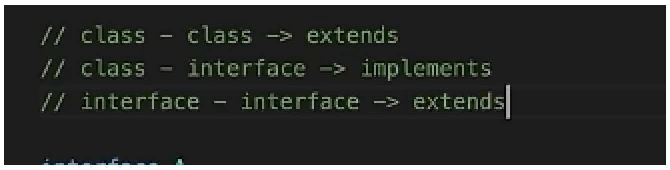
Anonymous Inner class

```
public class AnnonymusInnerClass {
   public static void main(String[] args) {
       A41 a41=new A41()
       {
           void show()
           {
               System.out.println("In AnonymousInnerClass");
           }
       };
       a41.show();
   }
}
```

Anonymous Abstract Inner Class

```
abstract class A42
{
  public abstract void show();
}
public class AbstractAndAnonymousInnerClass {
   public static void main(String[] args) {
       A42 a42=new A42()
       {
           public void show()
           {
               System.out.println("In AbstractAndAnonymousInnerClass Via A42
class");
           }
       };
       a42.show();
   }
}
```

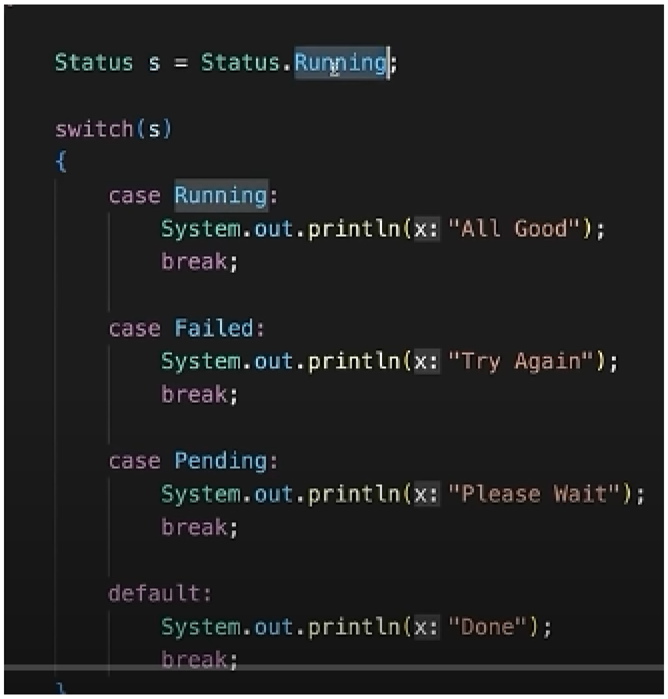
Interface

- Interface's methods are having public abstract by default no need to visible/dectare. EXAMPLE 👍
    - `void show();`
    - This will fetch from main(even not having public keyword)
- All variable inside interface are *final* and *static* so need to assign values at same time.\
- In interface we can implement multiple interfaces with class
- If we want to inert interface then use **extends** keyword
- 

```
// class - class -> extends
// class - interface -> implements
// interface - interface -> extends
```

Enum

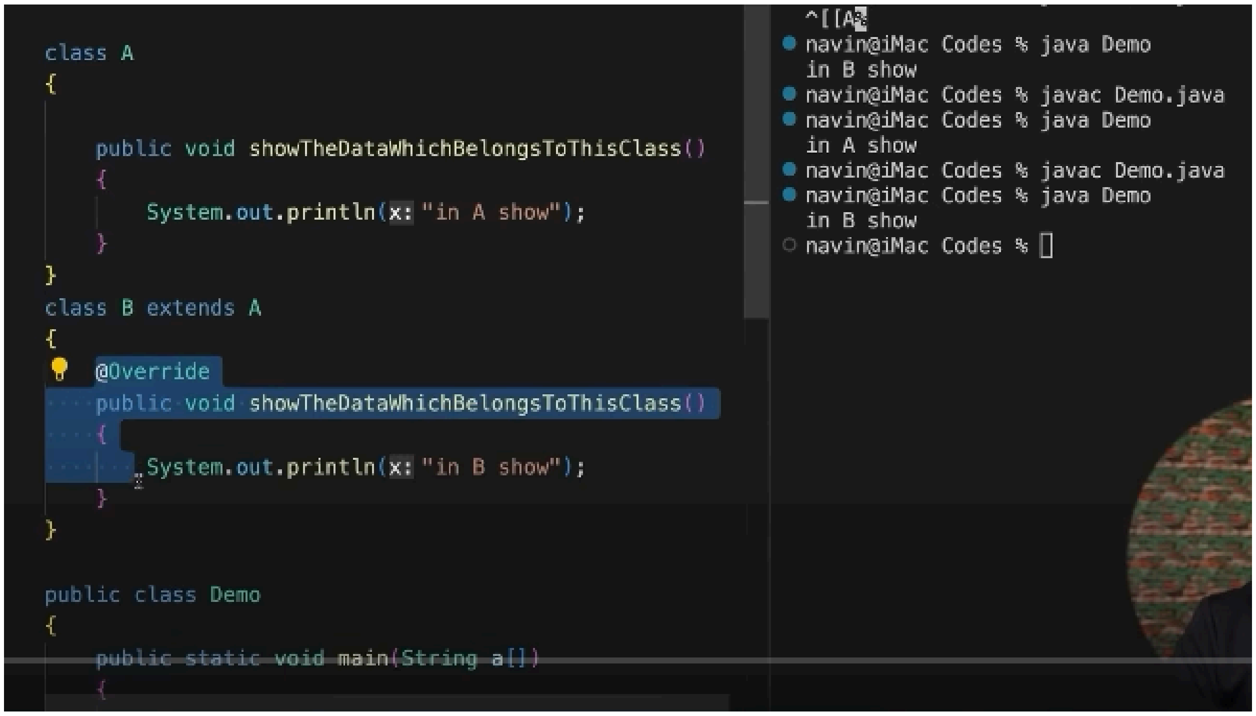
- In Java Enum starts with 0 (0,1,2....)
- .Ordinal is used to check the Order of the Enum
- Switch case looks good with enum
-



```java
Status s = Status.Running;

switch(s)
{
    case Running:
        System.out.println(x: "All Good");
        break;

    case Failed:
        System.out.println(x: "Try Again");
        break;

    case Pending:
        System.out.println(x: "Please Wait");
        break;

    default:
        System.out.println(x: "Done");
        break;
```

- We cannot extend the enums as sam class
- We define class methods, variable etc

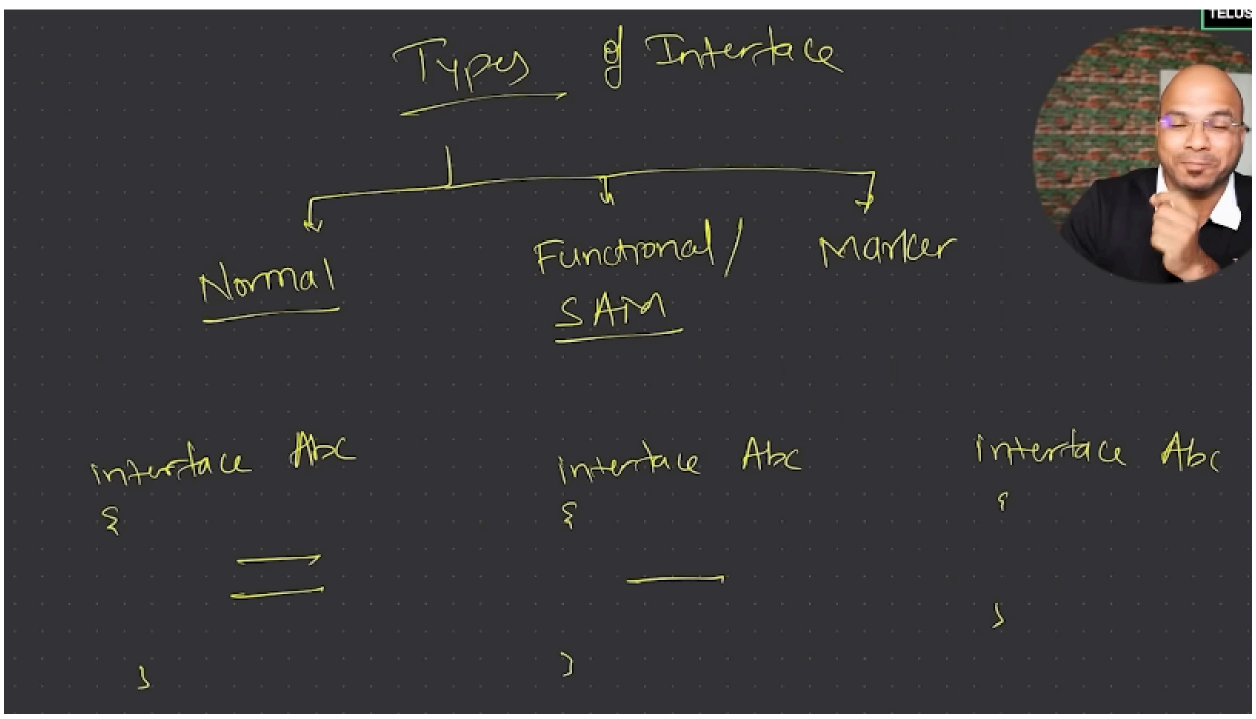
Annotations

- @override
- Deprecated
- 

```java
class A
{

    public void showTheDataWhichBelongsToThisClass()
    {
        System.out.println(x: "in A show");
    }
}
class B extends A
{
    @Override
    public void showTheDataWhichBelongsToThisClass()
    {
        System.out.println(x: "in B show");
    }
}

public class Demo
{
    public static void main(String a[])
    {
```

```
^[[A
navin@iMac Codes % java Demo
in B show
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
in A show
navin@iMac Codes % javac Demo.java
navin@iMac Codes % java Demo
in B show
navin@iMac Codes %
```

TYPE of Interface



SAM- Single abstract Method

- Lambda expression can only be used for SAM/Functional interface.
-

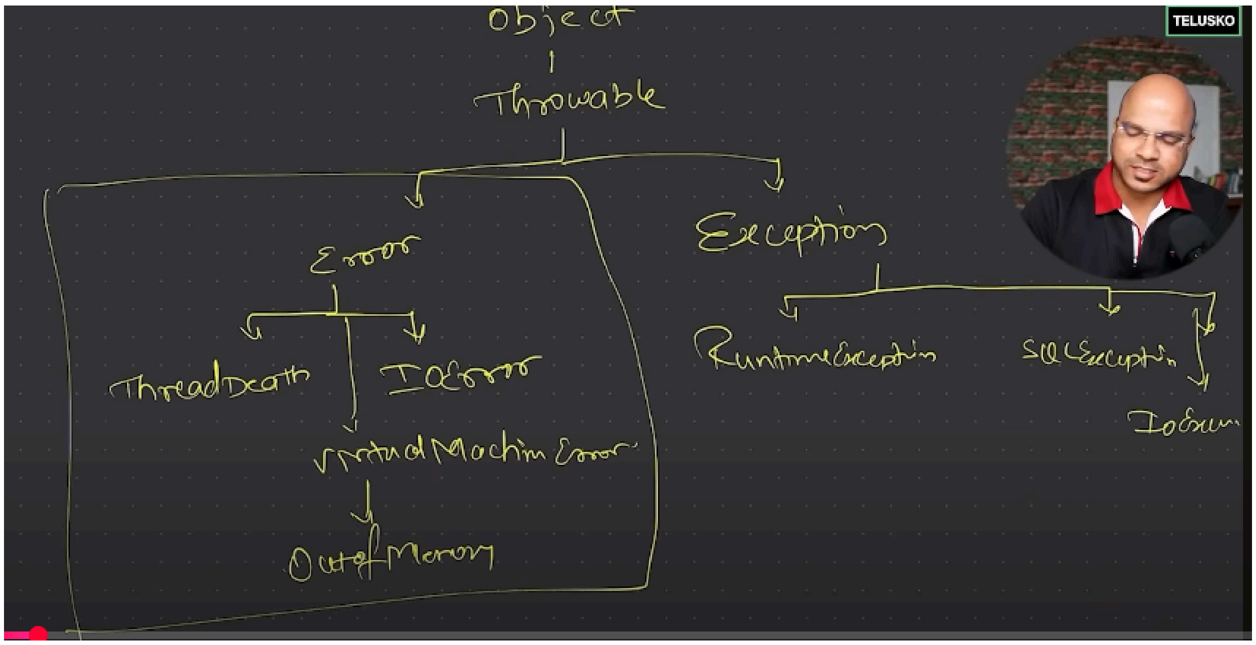
****************** JAVA 8 **********************

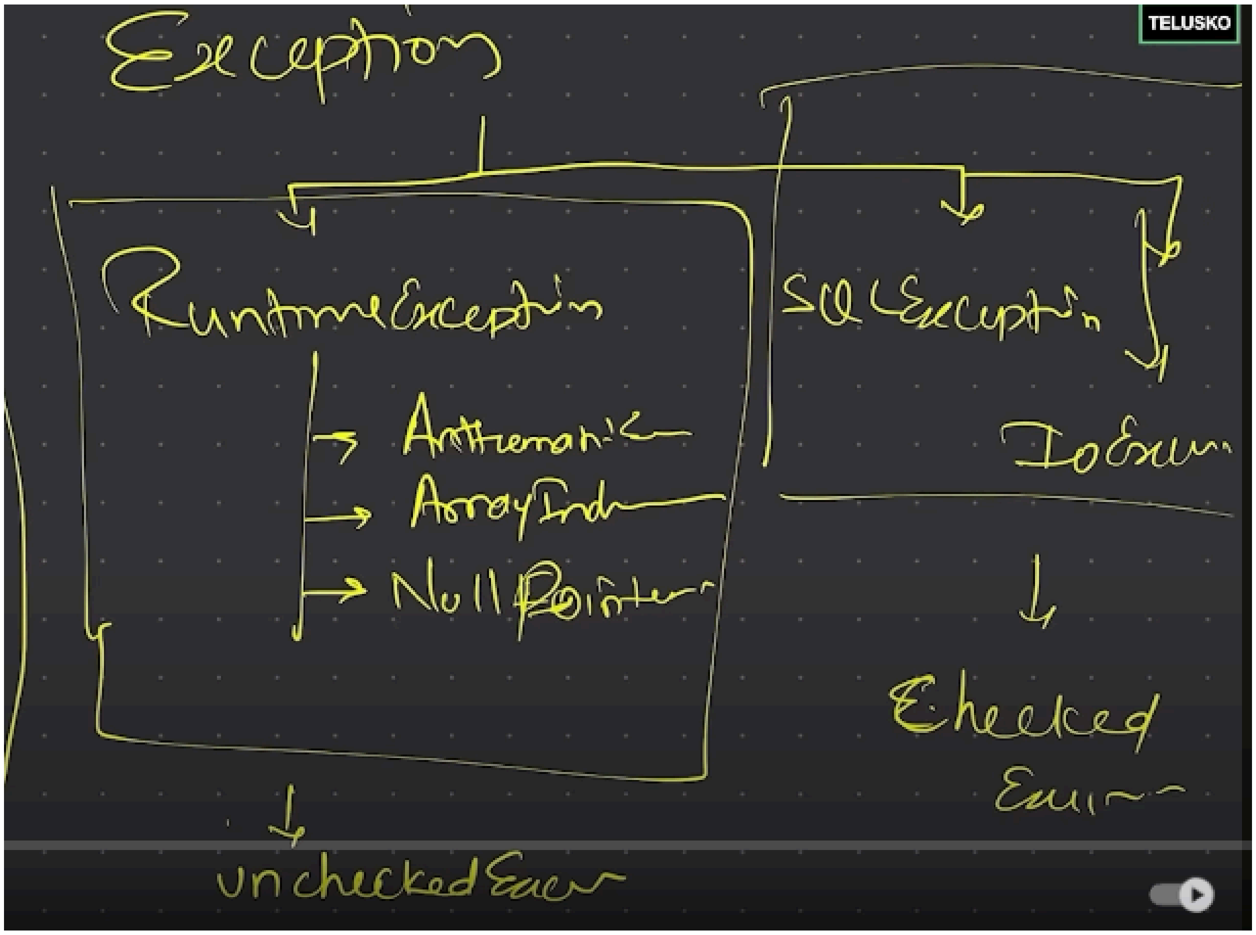
Lambda Expression

- Lambda expressions also do not create anonymous class file.
- Look at code : GITHUB

Exceptions

- Types
    - Compile Time error
    - Run time error
    - Logical error

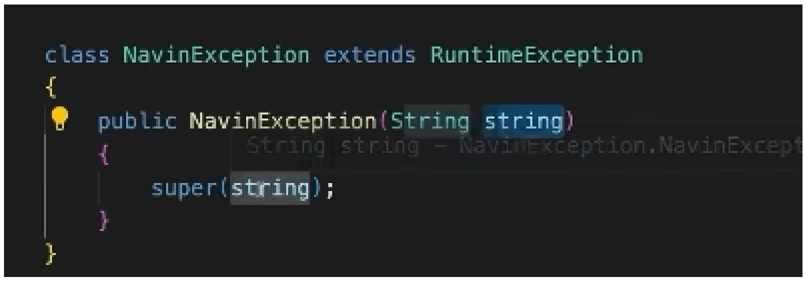




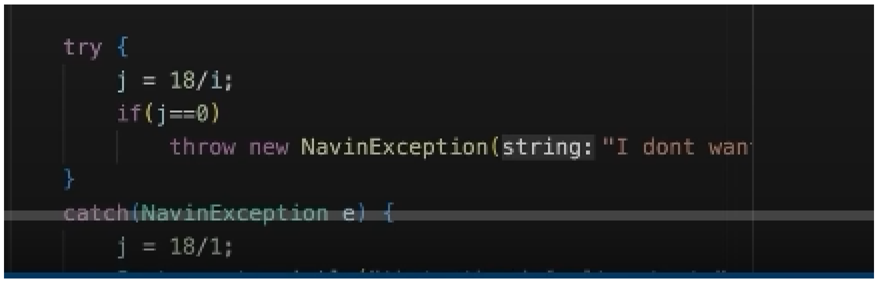
Throw

- Throw is especially use for throwing exception and catch will catch it

You can create your own exceptions.
Just define class of your exceptions and then use throw or other method to call



```
class NavinException extends RuntimeException
{
    public NavinException(String string)
    {
        super(string);
    }
}
```



```
try {
    j = 18/i;
    if(j==0)
        throw new NavinException(string: "I dont wan
}
catch(NavinException e) {
    j = 18/1;
```

Throws Keyword

Ducking the exception using throws keyword LIKE
B()
Try
Catch
Then
C()
Try
Catch
… and so on

StringBuffer and Scanner

- ***Println*** longs to **Printstream** class
- **Out** is a object of **printstream** which is created in **System** class
- System.in.read() ; here read() will give you ASCII value as o/p
- To get same number o/p then just **minus 48** from the input
- (num-48)
- Buffer Reader will take input from anywhere like FILE,NETWORK,KEYBOARD etc

Finally

- New concepts use (TRY with resources)
- But its better to use try with finally

Threads

Complete Java, Spring, and Microservices course
- Every thread must have a run method.
- Start() is used

Threads Priority

- 10 is the highest priority and 1 is the lowest priority and default is 5.
- To get the priority

```
threadsPriorityA ta=new threadsPriorityA();
threadsPriorityB tb=new threadsPriorityB();
System.out.println(ta.getPriority());
```

- To set the priority

```
tb.setPriority(Thread.NORM_PRIORITY);
```

- User can use **implement Runable** instead of ***extends Thread***

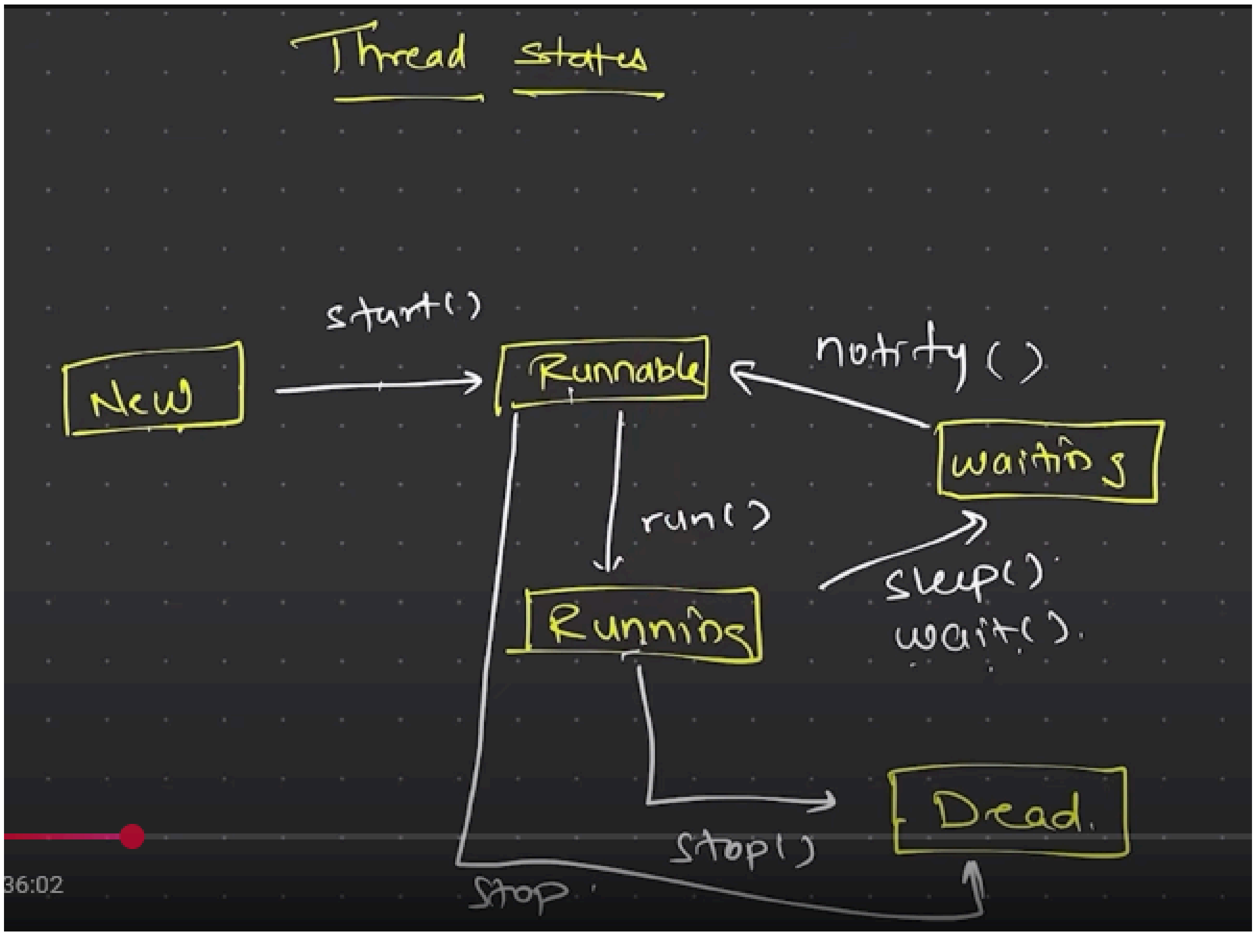
Race Conditions

- Need to use ***synchronized keyword*** to make sure execute one thread at a time. If we can not you then o/p may varies
- Use ***Joint()*** to join last o/p at the counter

```
class Counter
{
   int count;
   public synchronized void increment()
   {
       count++;
   }
}
public class RaceConditionExe {
   public static void main(String[] args) {
       Counter c=new Counter();
       Runnable ta=()->
       {
           for (int i = 0; i <1000; i++)
               c.increment();
       };
       Runnable tb=()->
       {
           for (int i = 0; i <1000; i++)
               c.increment();
       };
       Thread t1=new Thread(ta);
       Thread t2=new Thread(tb);
       t1.start();
       t2.start();
```

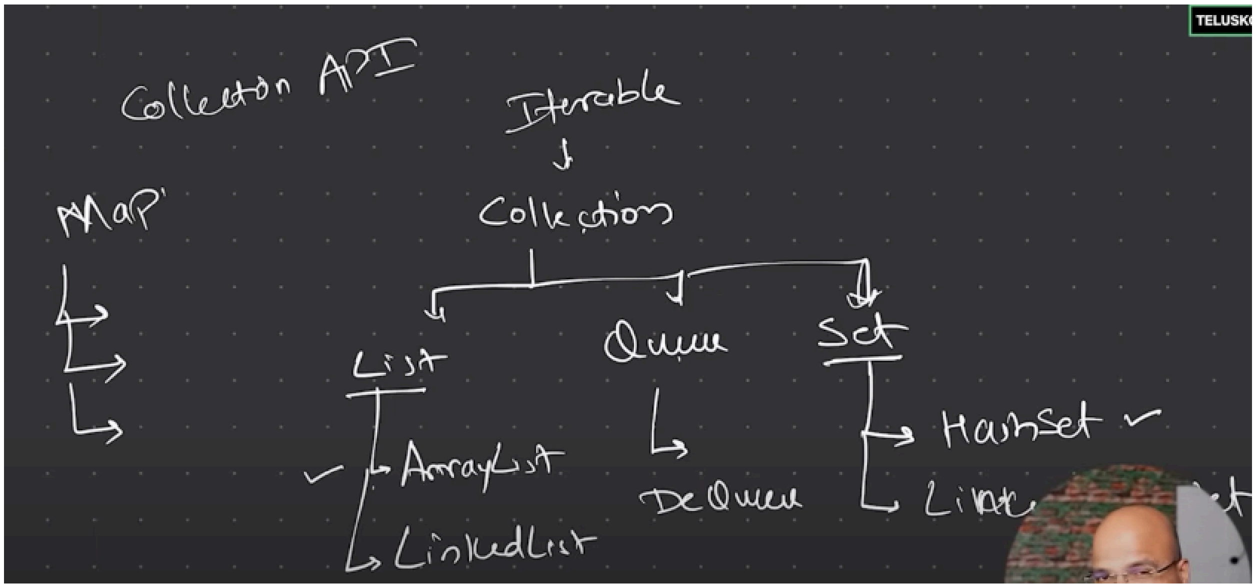
```
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        System.out.println(c.count);
    }
}
```

Thread States



Collections API

- Collection works with objects not with primitive values.
- Collection : Interface
- Collections : Class

- Inbuilt classes (below)
    - LIst
        - ArrayList
        - Linked list
    - Queue
        - Dequeue
    - Set
        - Hashset
        - Linked Hashset
- Generics
    - *Syntax* : **Collection<Integer> c=new Arraylist <Integer>();**
        - So here we can change the *type of Collection* as you want like float,Integer,String and so on…… .
- To work with index value then use **List**.
    - **List<Integer> c=new Arraylist <Integer>();**
- ***List*** supports ***Duplicate*** values while *set* support only *unique* values.
- Treeset is use to sorted values
    - **Set<Integer> c=new TreeSet<Integer>();**
- 

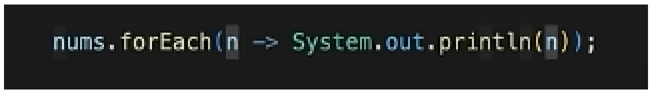
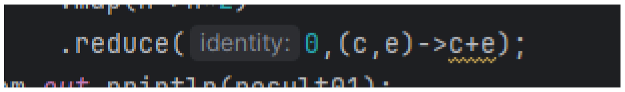
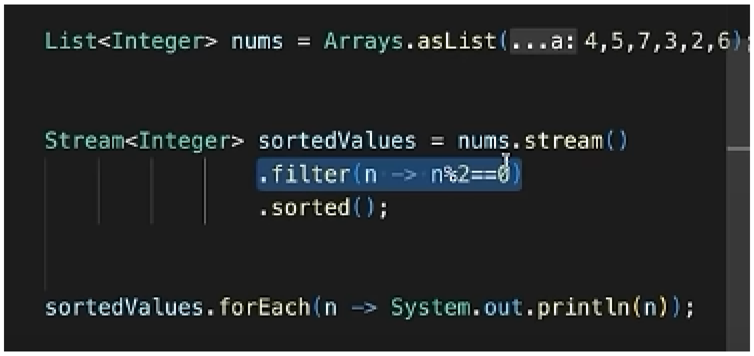
MAP

- MAP is a collection key and value pair.

Comparator

- It is a Concept Or Interface to specify your own logic of sorting.
- We can sort without using a comparator just implement Comparable then declare comparable conditions inside the class. Then it will work as same as other Comparator
- If you want to give to class itself then called comparable

Need of Stream API

- Foreach
  - 
    ```
    nums.forEach(n -> System.out.println(n));
    ```
- 
  ```
  .reduce( identity: 0,(c,e)->c+e);
  ```
  - C-> Carry
  - E is element
- 
  ```
  List<Integer> nums = Arrays.asList(...a: 4,5,7,3,2,6);


  Stream<Integer> sortedValues = nums.stream()
                  .filter(n -> n%2==0)
                  .sorted();


  sortedValues.forEach(n -> System.out.println(n));
  ```

NEW FEATURES of JAVA

JAVA is a easy and structured language

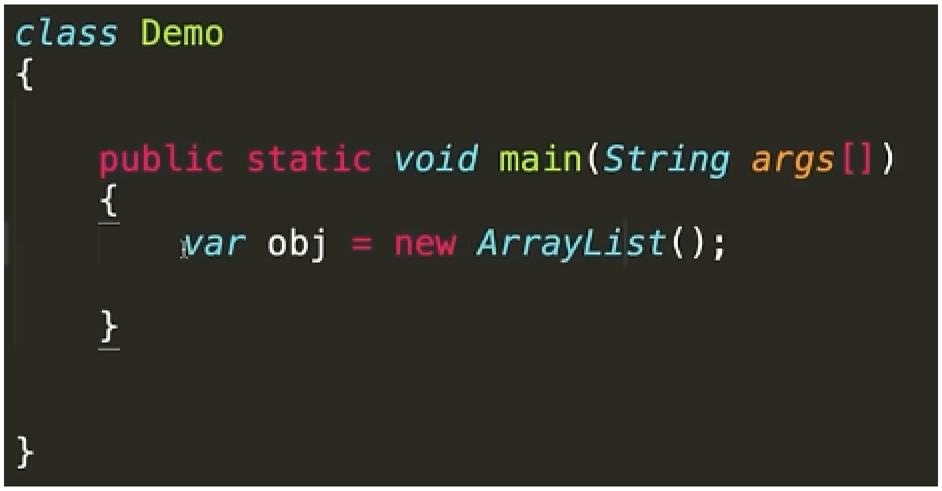
For learning purpose dev makes

Void main()
{
System.out.println("hello");
}

Above code will work on JDK version 21 onwards

LVTI - Local Variable Type Inference

1. In JAVA 10 no need to declare data type just write it as I=8;
   a. Now changes happen and we can use **var** keyword

b. 

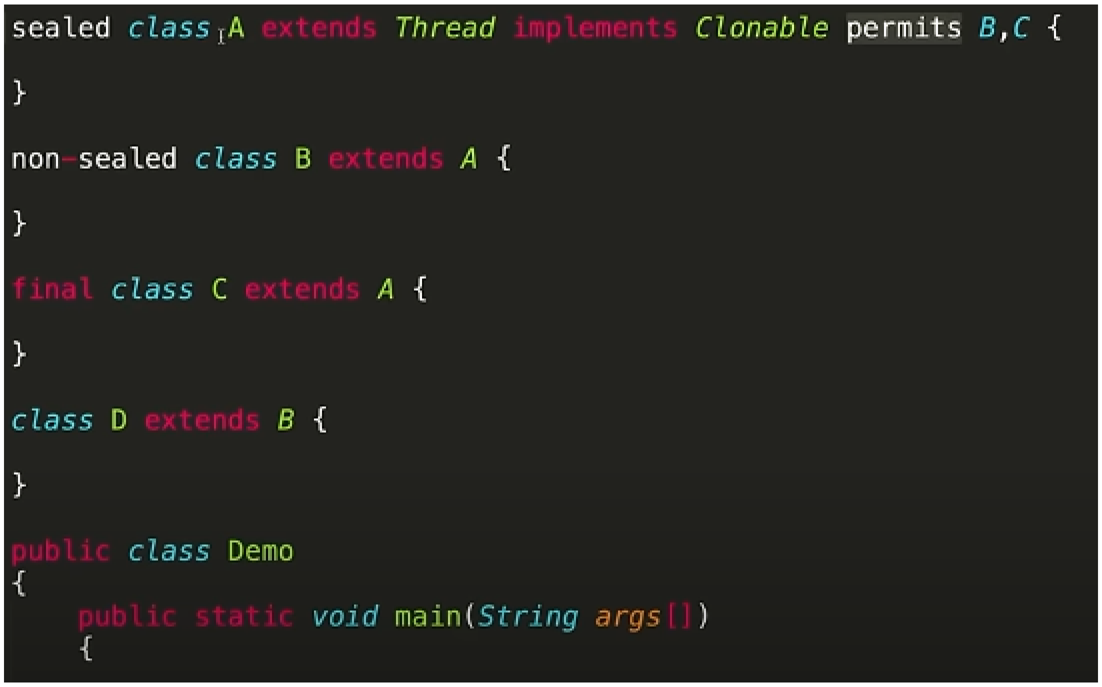
```java
class Demo
{

    public static void main(String args[])
    {
        var obj = new ArrayList();

    }


}
```

c. This applicable for local variable
d. **var=int** in JAVA 10
e. You can use as variable name LIKE : String **var**=”Abhishek”
f. **var** keyword cannot used as class name.
g. var cannot use as reference variable LIKE : **var[] nums=new int[6];**
    i. You can use LIKE: **var nums=new int[6];**
h. Abhishek obj=new Abhishek();
    i. **var obj=new Abhishek();**

SEALED CLASSES

To be continued…….. Complete Java, Spring, and Microservices course

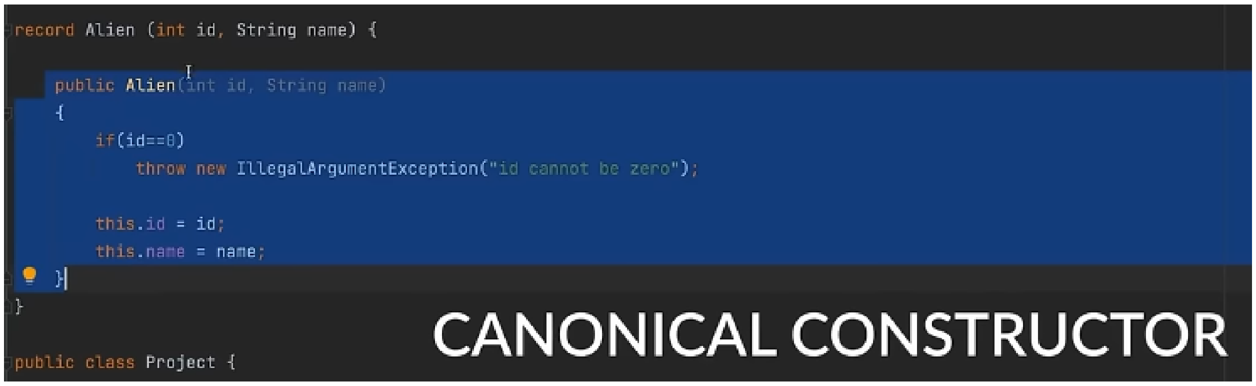
- To Inherit specific class and Interface you can use **sealed** Keyword

- 

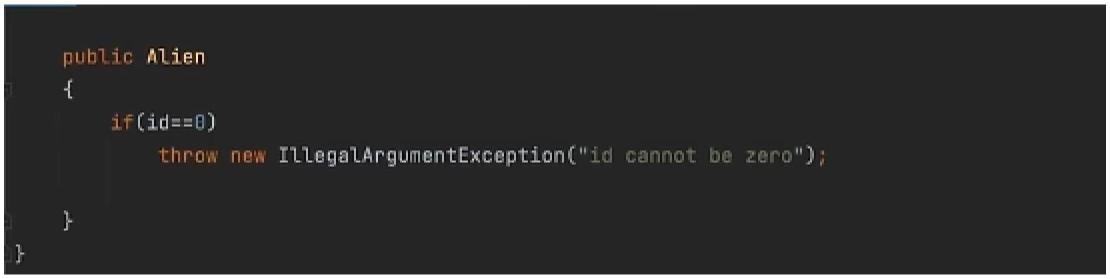
```
sealed class A extends Thread implements Clonable permits B,C {

}

non-sealed class B extends A {

}

final class C extends A {

}

class D extends B {

}

public class Demo
{
    public static void main(String args[])
    {
```

- Interface can not be final So you cannot use final Keyword
    - Use only **sealed** and **non-sealed**

Data Carrier Classes

- By default every field and All the variable **Private** and **final** because it s **immutable** data
- **Record** is a class and this class can extend any other class because it extends a record class but you can implement interfaces as many as you want, also you can create multiple methods (static or normal methods) as well as.
- Instance variable you can define only in brackets (int age…..) not inside the class.
- Static variable you can create inside No issue 👏
- Object is immutable (we can't change it)
- Object is only used for data storage
- Default Constructor not exist in this So you can create by your self
- 

```
record Alien (int id, String name) {

    public Alien(int id, String name)
    {
        if(id==0)
            throw new IllegalArgumentException("id cannot be zero");

        this.id = id;
        this.name = name;
    }
}

public class Project {
```

- 

```
    public Alien
    {
        if(id==0)
            throw new IllegalArgumentException("id cannot be zero");

    }
}
```

    - In above you can remove argument there no need to assign because already available.
-