First code in jshell

```
:\Users\DELL>jshell
|  Welcome to JShell -- Version 17.0.3.1
|  For an introduction type: /help intro

jshell> System.out.println("Hello World")
Hello World

jshell> System.out.println("Navin Reddy, Telusko")
Navin Reddy, Telusko

jshell> 2+4
$3 ==> 6

jshell> 9-6
$4 ==> 3

jshell>
```

How java works

```
public class hello {
	public static void main(String[] args) {
		System.out.print("Hello World");
```

```
    }
}
```

Variables

```
public class hello {
    public static void main(String[] args) {
        System.out.print(3+5);
        System.out.print(8+7);
        //int num1=3;
        //int num2=5;
        //System.out.println(num1+num2);
        int num1=3;
        int num2=5;
        int result=num1+num2;
        System.out.println(result);
    }
}
```

Data types

```
public class hello {
    public static void main(String[] args) {
        int num1=9;
        byte by=127;
        short sh=558;
```

```
        long l=5854l;


        float f=5.8f;
        double d=5.8;

        char c='k';

        boolean b=true;
    }
}
```

Literals

```
public class hello {
    public static void main(String[] args) {
    int num1=0b101;
    System.out.println(num1);

    int num2=0x7E;
    System.out.println(num2);

    int num3=10_00_00_000;
    System.out.println(num3);

    float num4=56;
```

```
        System.out.println(num4);

        double num5=56;
        System.out.println(num5);

        double num6=12e10;
        System.out.println(num6);

        boolean num7= 1;
        System.out.println(num7);

        char c='a';
        System.out.println(c);
        c++;

        char c1='a';
        c1++;
        System.out.println(c1);
        }
}
```

Type_conversion

```
public class hello {
        public static void main(String[] args) {
                byte b=127;
```

```
        int a=b;
        System.out.println(b);
        System.out.println(a);

        byte b1=125;
        int a0=b1;
        System.out.println(b1);
        System.out.println(a0);

        int aa=257;
        byte k=(byte)aa;

        float f=5.6f;
        int t=(int)f;

        int a2=2567;
        byte b2=(byte)a;
        System.out.println(k);

        byte a3=10;
        byte b4=20;
        int t=a*b;
        System.out.println(t);
}
```

}

Assignment operators

```
public class hello {
	public static void main(String[] args) {
/**		int num1=7;
		int num2=5;
		int result=num1+num2;
		System.out.println(result);
**/
/**
		int num1=7;
		int num2=5;
		int result=num1-num2;
		System.out.println(result);
**/
/**
		int num1=7;
		int num2=5;
		int result=num1*num2;
		System.out.println(result);
**/
/**
```

```
        int num1=7;
        int num2=5;
        int result=num1/num2;
        System.out.println(result);
**/
/**
        int num1=7;
        int num2=5;
        int result=num1%num2;
        System.out.println(result);
**/
        int num=7;
//      num=num+2;
//      num+=2;
//      num*=2;

        num++;    //post increment
        ++num;    //pre increment
        num--;    //post decrement
        --num;      //pre decrement
        System.out.println(num);

        int result=num++; //fetch the value and then increment
```

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

Relational operators

```
public class hello {
    public static void main(String[] args) {
        int x=6;
        int y=5;
//      boolean result= x<y;
//      boolean result= x>y;
//      boolean result= x>=y;
//      boolean result= x<=y;
//      boolean result= x!=y;
        boolean result= x==y;
        System.out.println(result);

        double a=8.8;
        double b=9.8;
//      boolean res = a<=b;
        boolean res = a>=b;

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

```
}
```

Logical operators

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

//		boolean result= x>y && a<b ;
//		boolean result= x>y || a<b ;
//		boolean result= x>y && a>b ;
//		boolean result= x>y || a>b ;
//		boolean result= x<y && a<b ;
//		boolean result= x<y || a<b ;
//		boolean result= x<y || a<b || a>1 ;

//		System.out.println(result);

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

If else

```
public class hello {
	public static void main(String[] args) {
//		int x=8;
//		System.out.println("Hello");
//		System.out.println("Bye");

//		int x=18;
//		if(x>10) {
//			System.out.println("Hello");
//		}

//		if(true) {
//			System.out.println("Hello");
//		}

//		int x=28;
//		if(x>10 && x<=20) {	//11-20
//			System.out.println("Hello");
//		}
//		System.out.println("Bye");

//		int x=28;
//		if(x>10 && x<=20) {	//11-20
```

```
//                  System.out.println("Hello");
//          }
//          else
//          System.out.println("Bye");

            int x=8;
            int y=7;
            if(x>y) {
                  System.out.println(x);
                  System.out.println("Thankyou");
            }
            else
                  System.out.println(y);
      }
}
```

If else if

```
public class hello {
      public static void main(String[] args) {
//          int x=8;
//          int y=7;
//          int z=8;
//
```

```
//		if(x>y && x>z)  //false
//		{
//			System.out.println(x);
//		}
//		else
//			System.out.println(y);

//		int x=8;
//		int y=7;
//		int z=9;
//		if(x>y && x>z)		//false
//			System.out.println(x);
//		else if(y>x && y>z)
//			System.out.println(y);
//		else
//			System.out.println(z);

		int x=8;
		int y=7;
		int z=9;
		if(x>y && x>z)		//false
			System.out.println(x);
		else if(y>z)
```

```
            System.out.println(y);
        else
            System.out.println(z);



    }
}
```

Ternary

```
public class hello {
    public static void main(String[] args) {
//          int n=4;
//          int result=0;
//          if(n%2==0)
//              result=10;
//
//          else
//              result =20;
//          System.out.println(result);

            int n=5;
            int result=0;
            result = n%2==0 ? 10 : 20;
            System.out.println(result);
```

```
	}
}
```

Switch statement

```
public class hello {
	public static void main(String[] args) {
//		int n=1;
//		if(n==1)
//			System.out.println("Monday");
//		else if(n==2)
//			System.out.println("Tuesday");
//		else if(n==3)
//		System.out.println("Wednesday");
//		else if(n==4)
//		System.out.println("Thursday");
//		else if(n==5)
//		System.out.println("Friday");
//		else if(n==6)
//		System.out.println("Saturday");
//		else
//		System.out.println("Sunday");

		int n=8;
		switch(n) {
```

```
case 1:
        System.out.println("Monday");
        break;
case 2:
        System.out.println("Tuesday");
        break;
case 3:
System.out.println("Wednesday");
        break;
case 4:
System.out.println("Thursday");
        break;
case 5:
System.out.println("Friday");
        break;
case 6:
System.out.println("Saturday");
        break;
case 7:
System.out.println("Sunday");
        break;
default:
```

```
                System.out.println("Enter a valid number");
            }
        }
}
```

Need for loop

```
public class hello {
        public static void main(String[] args) {
            //repeat this statement 4 times
            //loop -while, do while, for

            //100 - condition

            System.out.println("Hi");
        }
}
```

While loop

```
public class hello {
        public static void main(String[] args) {
            int i=1;

//          while(true)
//          {
//              System.out.println("Hi"+ i);
```

```
//                  i++;
//          }

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

            while(i<=4)
            {
                    System.out.println("Hi"+ i);
                    int j=1;
                    while(j<=3) {
                            System.out.println("Hello"+j);
                            j++;
                    }
                    i++;
            }
            System.out.println("Bye"+i);
    }
}
```

Do while loop

```
public class hello {
	public static void main(String[] args) {
		int i=1;
		do
		{
			System.out.println("Hi"+i);
			i++;
		}
		while(i<=4);
	}
}
```

For loop

```
public class hello {
	public static void main(String[] args) {
		for(int i=0;i<=4;i++)
		{
			System.out.println("Hi"+i);
		}

		for(int i=1;i<=7;i++)
		{
			System.out.println("Day"+i);
```

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

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

While loop to use

For loop:-

If you know how many iterations you want to go for loop

(intiliazing vaue, condition, increment or decrement)

While loop:-

When you need to read the file go for while loop, if the number of iterations is not known.

- For loop can also be used as a while.

Do While Loop:-

If you condition get false but you want to exceute the code at least once.

Class and object theory

```
class Demo
{
	public static void main(String[] args)
	{
	}
}
// Object Oriented programming
// Object - Properties and Beahaviors
//Class
```

Class and object practical

```
class Calculator{
	public int add(int n1, int n2)
	{
		//int a;
		//System.out.println("in add");
		//return 0;

		//int r=num1+num2;
```

```
            //return r;


            int r=n1+n2;
            return r;
        }
}

class Demo
{
        public static void main(String[] args)
        {
            int num1=4;
            int num2=5;
            Calculator calc= new Calculator();
            int result=calc.add(4,5);
            //calc.add();
            //int result=calc.add();
            //int result=num1+num2;
            System.out.println(result);
        }
}
// Object Oriented programming
```

// Object - Properties and Beahaviors

//Class

JDK JRE JVM

JDK- Java Development Kit

JVM- Java Virtual Machine

JRE- Java Runtime Environment

Methods

```
package classDemo;
/**
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[] args) {
		int num1=4;
```

```
        int num2=5;
        Calculator calc= new Calculator();
        int result = calc.add(num1,num2);
        System.out.println(result);
    }
}
    **/
class Computer
{
    public void playMusic()
    {
        System.out.println("Music Playing...");
    }
    public String getMeAPen(int cost)
    {
        if(cost>=10)
            return "Pen";
        else
            return "Nothing";
    }
}
```

```
public class Demo {
	public static void main(String[] args) {
		Computer obj=new Computer();
		obj.playMusic();
		String src=obj.getMeAPen(10);
		System.out.println(src);
	}
}
```

Method overloading

```
class Calculator
{
	public int add(int n1, int n2, int n3)
	{
		return n1+n2+n3;
	}
	public int add(int n1, int n2)
	{
		return n1+n2;
	}
	public double add(double n1, int n2)
	{
		return n1+n2;
```

```
    }
}
```

```
class Calculator
{
    int num=5;
    public int add(int n1, int n2)
    {
        System.out.println(num);
        return n1+n2;
    }
}
```

Stack and heap

```
public class Demo {
    public static void main(String[] args) {
        int data=10;
        Calculator obj=new Calculator();
        Calculator obj1=new Calculator();
        int r1=obj.add(3,4);
        //System.out.println(r1);
        obj.num=8;

        System.out.println(obj.num);
        System.out.println(obj1.num);
```

```
	}
}
```

```
public class Demo {
	public static void main(String[] args) {
		Calculator obj=new Calculator();
		int r1=obj.add(3,4);
		System.out.println(r1);
	}
}
```

Need of an array

```
int i=5;
int j=6;
int k=7;
int num[]= {5,6,7};
int num[]=new int[4];
```

Creation of array

```
public class Demo {
	public static void main(String[] args) {
//	 int nums[]= {3,7,2,4};
//		nums[1]=6;
```

```
//		System.out.println(nums[1]);

		int nums[]=new int[4];
		nums[0]=4;
		nums[1]=8;
		nums[2]=3;
		nums[3]=9;

//		System.out.println(nums[0]);
//		System.out.println(nums[1]);
//		System.out.println(nums[2]);
//		System.out.println(nums[3]);

		for (int i=0;i<4;i++) {
			System.out.println(nums[i]);
		}
	}
}
```

Multi dimensional array

```
public class Demo {
	public static void main(String[] args) {
		int nums[][]=new int [3][4];
		//inr random=0;
```

```
//int random=(int)Math.random()*100;

for(int i=0;i<3;i++)
{
	for(int j=0;j<4;j++)
	{
		nums[i][j]=(int)Math.random()*100;
		System.out.println(nums[i][j]);
	}
	System.out.println();
}

for(int i=0;i<3;i++)
{
	for(int j=0;j<4;j++)
	{
		System.out.println(nums[i][j]+" ");
	}
	System.out.println();
}

for(int n[]:nums)
{
```

```
                for(int m:n)
                {
                        System.out.println(m+" ");
                }
                System.out.println();
        }
    }
}
```

Jagged and 3D array

```
public class Demo {
    public static void main(String[] args)
    {
//          int nums[][]=new int [3][]; //jagged
//          nums[0]=new int [3];
//          nums[1]=new int [4];
//          nums[2]=new int [2];

            int nums[][]=new int [3][4];
//          int nums[][]=new int [3][4][5]; // three dimensional

            for(int i=0;i<nums.length;i++)
            {
                for(int j=0;j<nums[i].length;j++)
```

```
                {
                        nums[i][j]=(int)(Math.random()*10);
                }
        }
    }
}
```

Drawbacks of array

```
public class Demo {
    public static void main(String[] args)
    {
        int nums[]=new int[4];
    }
}
```

Drawbacks:-

- The memory allocation is contiguous.
- The size of an array is fixed. Array size cannot be expand.
- Searching takes time.
- Array can store values of only same type. It can store homogeneous type value only.

Array of objects

```
class Student
{
    int rollno;
```

```
    String name;
    int marks;
}

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

        System.out.println(s1.name + ":"+ s1.marks);
```

```
            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+":"+students[i].marks);
            }
//          int nums[] = new int[6];
//          nums[0]=4;
//          nums[1]=8;
//          nums[2]=3;
//          nums[3]=9;
//
//          for(int i=0;i<nums.length;i++)
//          {
//                  System.out.println(nums[i]);
//          }
      }
}
```

Enhanched for loop

class Student

```
{
	int rollno;
	String name;
	int marks;
}

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

```
        System.out.println(s1.name + ":"+ s1.marks);

        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+":"+students[i].marks);
//      }

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

        int nums[]=new int[4];
        nums[0]=4;
        nums[1]=8;
        nums[2]=3;
        nums[3]=9;
```

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

		for(int n: nums)
		{
			System.out.println(n);
		}
	}
}
```

What is string

```
public class Demo {
	public static void main(String[] args)
	{
		String name=new String();
		System.out.println(name);
		System.out.println(name.hashCode());
		System.out.println("hello "+name);
		System.out.println(name.concat("reddy"));
//	String name="Navin";
	}
```

```
}
```

Mutable vs immutable string

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

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

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

StringBuffer and StringBuilder

```
class Demo {
	public static void main(String[] args)
	{
		StringBuffer sb= new StringBuffer("Navin");
//		System.out.println(sb.length());
//		System.out.println(sb.capacity());
		sb.append("Reddy");
```

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

//      String str=sb.toString();

//      sb.deleteCharAt(2);
//      sb.insert(0,"Java");
//      sb.insert(6,"java");
//      sb.setLength(30);
        sb.ensureCapacity(100);

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

Static variable

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

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

```
}

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

		Mobile obj2=new Mobile();
		obj2.brand="Samsung";
		obj2.price=1700;
		//obj2.name="SmartPhone";
		Mobile.name="SmartPhone";

		//obj1.name="Phone";
		Mobile.name="SmartPhone";

		obj1.show();
		obj2.show();

		//System.out.println(obj1.brand);
```

```
	}
}
```

Static method

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

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

	public static void show1(Mobile obj)
	{
//		System.out.println("in static method");
		System.out.println(obj.brand+" : "+ obj.price +" : "+obj.name);
	}
}

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

```
    {
        Mobile obj1=new Mobile();
        obj1.brand="Apple";
        obj1.price=1500;
        //obj1.name="SmartPhone";
        Mobile.name="SmartPhone";

        Mobile obj2=new Mobile();
        obj2.brand="Samsung";
        obj2.price=1700;
        //obj2.name="SmartPhone";
        Mobile.name="SmartPhone";

        //obj1.name="Phone";
        Mobile.name="SmartPhone";

        obj1.show();
        obj2.show();

        Mobile.show1(obj1);

        //System.out.println(obj1.brand);
    }
}
```

Static block

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

	static {
		name="Phone";
		System.out.println("in static block");
	}

	public Mobile() {
		brand="";
		price=200;
//		name="Phone";
		System.out.println("in constructor");
	}

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

```
public class Demo {
	public static void main(String[] args) throws ClassNotFoundException
	{
		Class.forName("Mobile");

//		Mobile obj1=new Mobile();
//		obj1.brand="Apple";
//		obj1.price=1500;
//		Mobile.name="SmartPhone";
//		Mobile obj2=new Mobile();
	}
}
```

Encapsulation

```
class Human
{
	//int age;
	//private int age=11;
	private int age;
	//String name;
	//private String name="Navin";
	private String name;
```

```
    public int getAge()
    {
        return age;
    }
    public void SetAge(int a)
    {
        age=a;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String n)
    {
        name=n;
    }
}
public class Demo {
    public static void main(String[] args) throws ClassNotFoundException
    {
        Human obj=new Human();
        obj.SetAge(30);
```

```
            obj.setName("Reddy");
//          obj.age=11;
//          obj.name="Navin";
            System.out.println(obj.getName()+" : "+obj.getAge());

        }
}
```

Getters and setters

```
class Human
{
        //int age;
        //private int age=11;
        private int age;
        //String name;
        //private String name="Navin";
        private String name;

        public int getAge()
        {
            return age;
        }
        public void SetAge(int age)
        {
```

```
        this.age=age;
    }
//  public void SetAge(int a)
//  {
//      age=a;
//  }
//  public int abc()
//  {
//      return age;
//  }
//  public void xyz(int a)
//  {
//      age=a;
//  }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name=name;
    }
```

```
//	public void setName(String n)
//	{
//		name=n;
//	}
}
public class Demo {
	public static void main(String[] args) throws ClassNotFoundException
	{
		Human obj=new Human();
//		obj.xyz(30);
		obj.SetAge(30);
		obj.setName("Reddy");
//		obj.age=11;
//		obj.name="Navin";

//		System.out.println(obj.abc()+" : "+obj.getAge());
		System.out.println(obj.getName()+" : "+obj.getAge());
	}
}
```

This keyword

```
class Human
{
	private int age;
```

```
    private String name;

    public int getAge()
    {
        return age;
    }
    public void SetAge(int age, Human obj)
    {
        //Human obj1=new Human();
        Human obj1=obj;
        obj1.age=age;
        //this.age=age;
    }

//  public void SetAge(int a)
//  {
//      age=a;
//  }

    public String getName()
    {
        return name;
    }
```

```
	public void setName(String name)
	{
		this.name=name;
	}

//	public void setName(String n)
//	{
//		name=n;
//	}
}
public class Demo {
	public static void main(String[] args) throws ClassNotFoundException
	{
		Human obj=new Human();

		obj.SetAge(30,obj);
//		obj.SetAge(30);
		obj.setName("Reddy");

//		System.out.println(obj.abc()+" : "+obj.getAge());
		System.out.println(obj.getName()+" : "+obj.getAge());
	}
}
```

Constructor

```
class Human
{
	private int age;
	private String name;

	public Human()
	{
		age=12;
		name="John";
		//System.out.println("in constructor");
	}
	public int getAge(){
		return age;
	}
	public void SetAge(int age)
	{
		this.age=age;
	}

	public String getName()
	{
		return name;
```

```
	}
	public void setName(String name)
	{
		this.name=name;
	}
}
public class Demo {
	public static void main(String[] args) throws ClassNotFoundException
	{
		Human obj=new Human();
		Human obj1=new Human();
		System.out.println(obj.getName()+" : "+obj.getAge());

		obj.SetAge(30);
		obj.setName("Reddy");

		//System.out.println(obj.getName()+" : "+obj.getAge());
	}
}
```

Default vs parameterized constructor

```
class Human
{
	private int age;
```

```
    private String name;

    public Human()
    {
        age=12;
        name="John";
    }
    public Human(String name) {
        this.age=age;
        this.name=name;
    }
    public Human(int age, String name)
    {
        this.age=age;
        this.name=name;
    }

//  public Human()    //default constructor
//  {
//      age=12;
//      name="John";
//      //System.out.println("in constructor");
//  }
```

```
//    public Human(int a, String n)  //Parameterized constructor
//    {
//        age=a;
//        name=n;
//    }

    public int getAge(){
        return age;
    }
    public void SetAge(int age)
    {
        this.age=age;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name=name;
    }
}
```

```
public class Demo {
	public static void main(String[] args) throws ClassNotFoundException
	{
		Human obj=new Human();
//		Human obj1=new Human(18, "Navin");
		System.out.println(obj.getName()+" : "+obj.getAge());
//		System.out.println(obj1.getName()+" : "+obj1.getAge());

//		obj.SetAge(30);
//		obj.setName("Reddy");

		//System.out.println(obj.getName()+" : "+obj.getAge());
	}
}
```

This and super method

```
/**
// super()
class A
{
	public A()
	{
		super();
		System.out.println("in A");
```

```
	}
	public A(int n)
	{
		super();
		System.out.println("in A int");
	}
}
class B extends A
{
	public B()
	{
//		super();
		super(5);
		System.out.println("in B");
	}
	public B(int n)
	{
//		super();   //call default constructor of super class
		super(n);
		System.out.println("in B int");
	}
}
```

```
public class Demo {
	public static void main(String[] args)
	{
		B obj=new B();
//		B obj=new B(5);
	}
}
**/
//this()
class A
{
	public A()
	{
		super();
		System.out.println("in A");
	}
	public A(int n)
	{
		super();
		System.out.println("in A int");
	}
}
```

```
class B extends A
{
	public B()
	{
		super();
		System.out.println("in B");
	}
	public B(int n)
	{
		this();  //call constructor of same class
		System.out.println("in B int");
	}
}
public class Demo {
	public static void main(String[] args)
	{

//		B obj=new B();
		B obj=new B(5);
	}
}
```

This and super keyword

```
class A extends Object
{
	int num= 1;
}

class B extends A
{
	int num=2;
	public int getValue()
	{
		int num=3;
//		return this.num;
		return super.num;
	}
}

public class Demo
{
	public static void main(String a[])
	{
		B obj=new B();
//		System.out.println(obj.num);
```

```
            System.out.println(obj.getValue());
        }
}
```

Naming convention

```
public class Demo
{
        public static void main(String a[])
        {
        }
}
// Camel casing

// class and interface - Calc       ( first letter capital)
//variable and method - marks, show()    (small letters)
// constants - PIE, BRAND       (all capital letters)

//showMyMarks()     (first letter of each word is capital from second word)
//show_my_marks()   (words join by underscore)

// MyData

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

Anonymous object

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


public class Demo
{
	public static void main(String a[])
	{
		int marks;
		marks=99;

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

```
//              A obj=new A();
                A obj;
                obj=new A();

                obj.show();
        }
}
```

Need of inheritance

Inheritance:-

is, has

is is used in inheritance

```
Class Calc     //Parent class , Super, Base
{
        add()
        sub()
        multi()
        div()
}

Class AdvCalc     //Child class, Sub, Derived
{
        {
```

```
    }
}
```

What is inheritance

```
class Calc
{
    public int add(int n1, int n2)
    {
        return n1+n2;
    }
    public int sub(int n1, int n2)
    {
        return n1-n2;
    }
}

public class AdvCalc extends Calc
{
    public int multi(int n1, int n2)
    {
        return n1*n2;
    }
    public int div(int n1, int n2)
```

```
	{
		return n1/n2;
	}
}

public class Demo
{
	public static void main(String a[])
	{
//		Calc obj=new Calc();
		AdvCalc obj=new AdvCalc();
		int r1=obj.add(4, 5);
		int r2=obj.sub(7,3);
		int r3=obj.multi(5,3);
		int r4=obj.div(15,4);

		System.out.println(r1+" "+r2);

	}
}
```

Single and multilevel inheritance

```
class Calc
{
```

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

class AdvCalc extends Calc
{
    public int multi(int n1, int n2)
    {
        return n1*n2;
    }
    public int div(int n1, int n2)
    {
        return n1/n2;
    }
}
```

```
//class VeryAdvCalc extends Calc
class VeryAdvCalc extends AdvCalc
{
	public double power(int n1,int n2)
	{
		return Math.pow(n1, n2);
	}
}

public class Demo
{
	public static void main(String a[])
	{
//		Calc obj=new Calc();
//		AdvCalc obj=new AdvCalc();
		VeryAdvCalc obj=new VeryAdvCalc();

		int r1=obj.add(4, 5);
		int r2=obj.sub(7,3);
		int r3=obj.multi(5,3);
		int r4=obj.div(15,4);
		double r5=obj.power(4,2);
```

```
        System.out.println(r1+" "+r2+" "+r3+" "+r4+" "+r5);

    }
}
```

Multiple inheritance

```
class A{

}
class B extends A{

}
//class C extends A,B
// Multiple inheritance does not supported by Java
// Ambiguity issue
class C extends B{

}

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

    }
}
```

Method overriding

```
//class A
//{
//	public void show()
//	{
//		System.out.println("in A show");
//	}
//	public void config()
//	{
//		System.out.println("in A config");
//	}
//}
//class B extends A
//{
//	public void show()
//	{
//		System.out.println("in B show");
//	}
//}

class Calc
{
```

```
	public int add(int n1, int n2)
	{
		return n1+n2;
	}
}
class AdvCalc extends Calc
{
	public int add(int n1, int n2)
	{
		return n1+n2+1;
	}
}

public class Demo{
	public static void main(String args[])
	{
//		B obj=new B();
//		obj.show();
//		obj.config();

		AdvCalc obj=new AdvCalc();
		int r1=obj.add(3, 4);
		System.out.println(r1);
```

```
    }
}
```

Packages

```
package other.tools;
//package tools;

public class Calc
{
    public int add(int n1, int n2)
    {
        return n1+n2;
    }
    public int sub(int n1, int n2)
    {
        return n1-n2;
    }
}


package tools;

public class AdvCalc extends Calc
```

```
{
	public int multi(int n1, int n2)
	{
		return n1*n2;
	}
	public int div(int n1, int n2)
	{
		return n1/n2;
	}
}
```

```
package other;

public class A
{

}
```

```
package com.google.Calculation;

//import tools.Calc;
//import tools.AdvCalc;
```

```
//import tools.*;
import other.tools.*;
//import.other.*;

//import java.util.ArrayList;
import java.lang.*;

public class Demo{
	public static void main(String args[])
	{
//		ArrayLis list=new ArrayList();

		Calc obj=new Calc();
		AdvCalc obj1=new AdvCalc();

		A obj2=new A();

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

Access modifiers

```
class Launch
{
```

```
        A obj=new A();
        System.out.println(obj.marks);
}

package other;
public class A{
        int marks=6;
//      protected marks=6;
        public void show()
        {

        }
}

public class B
{
//      private int marks;
        int marks;

}

class C extends A
{
```

```
    public void abc()
    {
        System.out.println(marks);
    }
}

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

    A obj=new A();
    System.out.println(obj.marks);
    obj.show();

    B obj1=new B();
    System.out.println(obj.marks);

    }
}
```

Polymorphism

Polymorphism:

- Many behaviour (same object or reference has differnet behaviours)

1. Compile time polymorhism -- Overloading

add(int, int)

add(int, int, int)

2. Run time polymorphism -- Overriding

A

add(int,int)

B

add(int, int)

Dynamic method dispatch

```
// class Computer
// {
//
// }
// class Laptop extends Computer
// {
//
// }

class A
{
	public void show()
	{
```

```
        System.out.println("in A show");
    }
}

class B extends A
{
    public void show()
    {
        System.out.println("in B show");
    }
}

class C extends A
{
    public void show()
    {
        System.out.println("in C show");
    }
}

class D
{
```

```
}

public class Demo{
	public static void main(String[] args) {
//		A obj=new B();
//		obj.show();

		A obj=new A();
		obj.show();

		obj=new B();
		obj.show();

		obj=new C();
		obj.show();

//		obj=new D();

//		Laptop obj1=new Laptop();
//		Computer obj1=new Laptop();

	}
}
```

Final keyword

```
//final - variable, method, class


//final class Calc
//{
//	public void show()
//	{
//		System.out.println("in Calc show");
//	}
//	public void add(int a, int b)
//	{
//		System.out.println(a+b);
//	}
//}



class Calc
{
	public final void show()
	{
		System.out.println("By Navin");
	}
	public void add(int a, int b)
```

```
	{
		System.out.println(a+b);
	}
}

class AdvCalc extends Calc
{
	public void show()
	{
		System.out.println("By John");

}

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

//	final int num=8;
//	num=9;
//	System.out.println(num);

//	Calc obj= new Calc();
//	obj.show();
//	obj.add(4, 5);
```

```
        AdvCalc obj= new AdvCalc();
        obj.show();
        obj.add(4, 5);
    }
}
```

Object class equals tostring hashcode

```
class Laptop
{
        String model;
        int price;
//      String serial;

        public String toString()
        {
//              return "Hey";
                return model+ " : "+price;
        }
        public boolean equals(Laptop that)
        {
//              if(this.model.equals(that.model) && this.price==that.price)
//                      return true;
//              else
//                      return false;
```

```
            return this.model.equals(that.model) &&
this.price==that.price ;
        }
}

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

        Laptop obj=new Laptop();
        obj.model="Lenevo Yoga";
        obj.price=1000;

        Laptop obj2=new Laptop();
        obj2.model="Lenevo Yoga";
 //     obj2.model="Lenevo Yoga1";
        obj2.price=1000;

        boolean result = obj.equals(obj2);

        System.out.println(obj.toString());
//      System.out.println(obj);

    }
```

```
}
```

Upcasting and downcasting

```
class A
{
	public void show1()
	{
		System.out.println("in A show");
	}
}
class B extends A
{
	public void show2()
	{
		System.out.println("in show B");
	}
}

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

//	double d=4.5;
//	int i=(int)d;
```

```
//
//      System.out.println(i);

//      A obj= new A();
//      A obj=(A) new B();   //upcasting
//      obj.show1();

        A obj=new B();
        obj.show1();

        B obj1=(B)obj;
        obj1.show2();
    }
}
```

Abstarct keyword

```
abstract class Car
{
//      public void drive()
//      {
//
//      }

        public abstract void drive();
        public abstract void fly();
```

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

abstract class WagnoR extends Car
{
//	public void fly()
//	{
//		System.out.println("Flying...");
//	}
	public void drive()
	{
		System.out.println("Driving...");
	}
}

class UpdateWagnoR extends WagnoR   //concrete class
{
	public void fly()
	{
```

```
            System.out.println("flying...");
        }
}

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

 //     Car obj=new Car();
 //     Car obj=new WagnoR();
        Car obj=new UpdateWagnoR();
        obj.drive();
        obj.playMusic();
    }
}
```

Inner class

```
class A
{
        int age;

        public void show()
        {
                System.out.println("in show");
        }
```

```
//     class B
//     {
//             public void config()
//             {
//                     System.out.println("in config");
//             }
//     }

        static class B
        {
                public void config()
                {
                        System.out.println("in config");
                }
        }
}


public class  Demo{
    public static void main(String[] args) {
        A obj=new A();
        obj.show();
```

```
//      A.B obj1=obj.new B();
//      obj1.config();

        A.B obj1=new A.B();
        obj1.config();


    }
}
```

Anonymous inner class

```
class A
{
        public void show()
        {
                System.out.println("in A show");
        }
}
//class B extends A
//{
//      public void show()
//      {
//              System.out.println("in B Show");
//      }
```

```
//}

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

       //A obj=new B();

       A obj=new A()
       {
              public void show()
              {
                     System.out.println("in new show");
              }
       };
       obj.show();
   }
}
```

Abstract and anonymous inner class

```
abstract class A
{
       public abstract void show();
       public abstract void config();
```

```
}

//class B extends A
//{
//	public void show()
//	{
//		System.out.println("in B show");
//	}
//}

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

//	A obj=new B();

	A obj=new A()
	{
		public void show()
		{
			System.out.println("in new show");
		}
	};
	obj.show();
```

```
    }
}
```

Need of interface

```
/*
abstract class Computer
{
//      public void code()
//      {
//
//      }
        public abstract void code();
}


class Laptop extends Computer
{
        public void code()
        {
                System.out.println("code, compile, run");
        }


}
```

```
class Desktop extends Computer
{
	public void code()
	{
		System.out.println("code, compile, faster");
	}
}
class Developer
{
//	public void devApp(Laptop lap)
	public void devApp(Computer lap)
	{
		lap.code();
	}
}

public class Demo {
   public static void main(String[] args) {
//		Laptop lap=new Laptop();
//		Desktop desk=new Desktop();

		Computer lap=new Laptop();
		Computer desk=new Desktop();
```

```
        Developer navin=new Developer();
        navin.devApp(lap);

    }
}
*/
interface Computer
{
        void code();
}

class Laptop implements Computer
{
        public void code()
        {
                System.out.println("code, compile, run");
        }

}

class Desktop implements Computer
{
```

```
        public void code()
        {
                System.out.println("code, compile, faster");
        }
}
class Developer
{
//      public void devApp(Laptop lap)
        public void devApp(Computer lap)
        {
                lap.code();
        }
}

public class Demo {
   public static void main(String[] args) {
//        Laptop lap=new Laptop();
//        Desktop desk=new Desktop();

        Computer lap=new Laptop();
        Computer desk=new Desktop();

        Developer navin=new Developer();
```

```
        navin.devApp(lap);

    }
}
```

What is interface

```
interface A
{
//      public abstract void show();
//      public abstract void config();
        int age=44;             // final and static
        String area="Mumbai";

        void show();
        void config();
}

class B implements A
{
        public void show()
        {
                System.out.println("in show");
        }
        public void config()
```

```
	{
		System.out.println("in cofing");
	}
}

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

	A obj;
	obj=new B();

	obj.show();
	obj.config();

//	A.area="Hyderabad";

	System.out.println(A.area);

   }
}
```

More on interface

// class - class -> extends

// class - interface -> implements

// interface - interface -> extends

```
interface A
{
//      public abstract void show();
//      public abstract void config();
        int age=44;             // final and static
        String area="Mumbai";

        void show();
        void config();
}

interface X
{
        void run();
}
interface Y extends X
{

}

class B implements A,Y
{
```

```
    public void show()
    {
        System.out.println("in show");
    }
    public void config()
    {
        System.out.println("in cofing");
    }
    public void run()
    {
        System.out.println("running...");
    }
}

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

    A obj;
    obj=new B();

    obj.show();
    obj.config();
```

```
        X obj1=new B();
        obj1.run();

//      A.area="Hyderabad";

        System.out.println(A.area);

    }
}
```

What is enum

```
enum Status{
        Running, Failed, Pending, Success;
}

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

        int i=5;
//      Status s= Status.Running;
//      Status s= Status.Failed;
//      Status s= Status.NoIdea;
//      Status s= Status.Success;

//      System.out.println(s);
```

```
//      System.out.println(s.ordinal());

        Status[] ss=Status.values();
        System.out.println(ss);

        for(Status s:ss)
        {
                System.out.println(s);
                System.out.println(s+" : "+s.ordinal());
        }
    }
}
```

Enum if and switch

```
enum Status{
        Running, Failed, Pending, Success;
}

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

        Status s=Status.Pending;

        switch(s)
```

```
{
        case Running:
                System.out.println("All Good");
                break;

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

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

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

if(s==Status.Running)
        System.out.println("All Good");
else if(s==Status.Failed)
        System.out.println("Try Again");
else if ( s==Status.Pending)
        System.out.println("Please Wait");
```

```
        else
                System.out.println("Done");

    }
}
```

Enum class

```
enum Laptop{
//      Mackbook(2000), XPS(2200), Surface(1500), ThinkPad(1800);
        Mackbook(2000), XPS(2200), Surface, ThinkPad(1800);

        private int price;

        private Laptop()
        {
                price=500;
        }

        private Laptop(int price)
        {
                this.price=price;
        }

        public int getPrice()
        {
```

```
            return price;
        }
        public void setPrice(int price)
        {
            this.price=price;
            System.out.println("in Laptop" + this.name());
        }
}

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

//      Laptop lap=Laptop.Mackbook;
//      System.out.println(lap+ " : "+lap.getPrice());

        for(Laptop lap : Laptop.values())
        {
            System.out.println(lap+" : "+lap.getPrice());
        }
    }
}
```

What is Annotations

@Deprecated

```
class A
{
	public void showTheDataWhichBelongsToThisClass()
	{
		System.out.println("in show A");
	}
}

class B extends A
{
	@Override
//	public void showTheDataWhichBelongToThisClass()
	public void showTheDataWhichBelongsToThisClass()

	{
		System.out.println("in show B");
	}
}

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

	B obj=new B();
```

```
        obj.showTheDataWhichBelongsToThisClass();

    }
}
```

Functional interface new

```
@FunctionalInterface
interface A
{
        void show();
//      void run();
}
//class B implements A
//{
//      public void show()
//      {
//              System.out.println("in Show");
//      }
//}

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

        A obj=new A()
```

```
        {
                public void show()
                {
                        System.out.println("in Show");
                }
        };
 //     A obj=new A();
 //     A obj=new B();
        obj.show();
    }
}
```

Lambda expression

```
@FunctionalInterface
interface A
{
//      void show();
        void show(int i);
//      void show(int i,int j);



}

public class Demo {
```

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

//    A obj=() -> System.out.println("in Show");
//    obj.show();

//    A obj=new A()
//    {
//          public void show(int i)
//          {
//                System.out.println("in show"+i);
//          }
//    };
//    obj.show(5);

//    A obj=(int i) ->System.out.println("in show "+i);
//    obj.show(5);

//    A obj=(int i,int j) ->System.out.println("in show "+i);
//    obj.show(5,8);

      A obj=i -> System.out.println("in show "+i);
      obj.show(5);
   }
}
```

Lambda expression with return statement

```
FunctionalInterface
interface A
{
	int add(int i, int j);
}

public class Demo {
   public static void main(String[] args) {
//
//	A obj=new A()
//	{
//		public int add(int i, int j)
//		{
//			return i+j;
//		}
//	};

	A obj=(i,j) -> i+j;
	int result=obj.add(5, 4);
	System.out.println(result);
   }
}
```

Types of interface

Types of Interface:-

1. Normal interface
    - an interface having two or more methods
2. Functional interface (SAM)
    - SAM => Single Abstract Method interface
3. Marker interface
    - an interface that as no methods (blank interface)

What is exception

Types of error:-

1. Complile - time error
2. Runtime error -> Exception handling
3. Logical error

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

//     System.out.Println();
       System.out.println(2+2);
   }
}
```

Exception handling with try catch

```
public class Demo {
```

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

    int i=0;
    int j=0;

    try
    {
        j=18/i;
    }
    catch(Exception e)
    {
        System.out.println("Something went wrong");
    }
    System.out.println(j);

    System.out.println("Bye");
    }
}
```

Try with multiple catch block

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

        int i=2;
  //    int i=0;
```

```
        int j=0;

        int nums[]=new int[5];
        String str=null;

        try
        {
                j=18/i;
                System.out.println(str.length());
                System.out.println(nums[1]);
                System.out.println(nums[5]);
        }
//      catch(Exception e)
//      {
//
//              System.out.println("Something went wrong."+e);
//      }
        catch(ArithmeticException e)
        {
                System.out.println("Cannot divide by zero");
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
```

```
            System.out.println("Stay in your limit.");
        }
        catch(Exception e)
        {

            System.out.println("Something went wrong."+e);
        }
        System.out.println(j);
        System.out.println("Bye");
    }
}
```

Exception hierarchy

Error and Exception extends Throwable class.

Throwable class is extended by Object class.

Error is divided into :-

1. Thread Death
2. Virtual Machine error (Out of memory)
3. IO Error

Exception is divided into:-

1. Runtime Exception (Unchecked Exception)
    - Arithmetic

- ArrayIndexOutOfBound
- Null POinter

- It is your choose to handle or not

2. SQL Exception (Checked Exception)

- It is necessary to handle

3. IO Exception (Checked Exception)

- It is necessary to handle

Exception throw keyword

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

//     int i=2;
       int i=0;
       int j=0;


       try
       {
            j=18/i;
            if(j==0)
                 throw new ArithmeticException("I don't want to do print
zero");
```

```
        }

        catch(ArithmeticException e)
        {
                j=18/i;
                System.out.println("that is default output"+e);

//              System.out.println("Cannot divide by zero");
        }

        catch(Exception e)
        {

                System.out.println("Something went wrong."+e);
        }
        System.out.println(j);
        System.out.println("Bye");
    }
}
```

Custom exception

```
class NavinException extends Exception
{
        public NavinException(String string)
```

```
	{
		super(string);
	}
}
public class Demo {
   public static void main(String[] args) {

//	int i=2;
//	int i=0;
	int i=20;
	int j=0;


	try
	{
		j=18/i;
		if(j==0)
//			throw new Exception("I don't want to do print zero");
			throw new NavinException("I don't want to do print
zero");
	}


	catch(ArithmeticException e)
	{
```

```
            j=18/i;
            System.out.println("that is default output"+e);

//          System.out.println("Cannot divide by zero");
        }

        catch(Exception e)
        {

            System.out.println("Something went wrong."+e);
        }
        System.out.println(j);
        System.out.println("Bye");
    }
}
```

Ducking exception using throws

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

```
class A
{
	public void show() throws ClassNotFoundException
	{
//	try
//	{
//		Class.forName("Calc");
//	}
//	catch(ClassNotFoundException e)
//	{
//		System.out.println("Not able to find theh class");
//	}

		Class.forName("Calc");
	}
}

public class Demo {

	static {
		System.out.println("Class Loader");
	}
```

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

//      try
//      {
//              Class.forName("Class");
//      }
//      catch(ClassNotFoundException e)
//      {
//              System.out.println("Not able to find theh class");
//      }

        A obj=new A();
        try {
                obj.show();
        }catch(ClassNotFoundException e)
        {
                e.printStackTrace();
        }
    }
}
```

User input using bufferReader and scanner

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

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

//     System.out.println("Enter a number");
//     int num=System.in.read();
//
//     System.out.println(num);
//     System.out.println(num-48);

       System.out.println("Enter a number");

//     InputStreamReader in=new InputStreamReader(System.in);
//     BufferedReader bf=new BufferedReader(in);

//     int num=Integer.parseInt(bf.readLine());
//     System.out.println(num);
  //   BufferedReader bf=new BufferedReader(null);
  //   System.out.println(num-48);

       Scanner sc=new Scanner(System.in);
```

```
        int num=sc.nextInt();
        System.out.println(num);
    }
}
```

Try with resources

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

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

        int i=0;
        int j=0;
        try
        {
                j=18/i;

        }
        catch(Exception e)
        {
                System.out.println("Someting went wrong.");
                System.out.println("Bye");
```

```
        }
        finally
        {
                System.out.println("Bye");
        }

        int num=0;
        //BufferedReader br=null;
        try(BufferedReader br=new BufferedReader(new
InputStreamReader(System.in)))
        {
//              InputStreamReader in =new InputStreamReader(System.in);
//              BufferedReader br=new BufferedReader(in);
                num=Integer.parseInt(br.readLine());
                System.out.println(num);
        }
        finally
        {
                //br.close();
        }
    }
}
```

Threads

Threads:-

Multiple threads run at same time in a code.

This is known as Multithreading.

- A thread is a smallest unit to work with. (individual task)
- They can run parallely.
- Multiple threads can share resources.

Multiple threads

```
class A extends Thread
{
	public void run()
	{
		for(int i=1;i<=100;i++)
		{
			System.out.println("Hi");
		}
	}
}

class B extends Thread
{
	public void run()
```

```
        {
                for(int i=1;i<=100;i++)
                {
                        System.out.println("Hello");
                }
        }
}
public class Demo {
   public static void main(String[] args) throws NumberFormatException {

        A obj1=new A();
        B obj2=new B();

//      obj1.show();
//      obj2.show();

        obj1.start();
        obj2.start();
   }

}
```

Thread prority and sleep

```
class A extends Thread
{
```

```
	public void run()
	{
		for(int i=1;i<=100;i++)
		{
			System.out.println("Hi");
			try {
				Thread.sleep(10);
			}catch(InterruptedException e) {
				e.printStackTrace();
			}
		}
	}
}

class B extends Thread
{
	public void run()
	{
		for(int i=1;i<=100;i++)
		{
			System.out.println("Hello");
			try {
```

```
                Thread.sleep(10);
            }catch(InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

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

        A obj1=new A();
        B obj2=new B();

//      obj1.show();
//      obj2.show();

        obj2.setPriority(Thread.MAX_PRIORITY);
        System.out.println(obj1.getPriority());

        obj1.start();
        try {
                Thread.sleep(2);
            }catch(InterruptedException e) {
```

```
                e.printStackTrace();
            }
        obj2.start();
    }
}
```

Runnable vs throwable

```
class Z
{

}

class A implements Runnable
{
        public void run()
        {
                for(int i=1;i<=5;i++)
                {
                        System.out.println("Hi");
                        try {
                                Thread.sleep(10);
                        }catch(InterruptedException e) {
                                e.printStackTrace();
                        }
```

```
            }
        }
}

class B implements Runnable
{
        public void run()
        {
                for(int i=1;i<=5;i++)
                {
                        System.out.println("Hello");
                        try {
                                Thread.sleep(10);
                        }catch(InterruptedException e) {
                                e.printStackTrace();
                        }
                }
        }
}

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

```
//     A obj1=new A();
//     B obj2=new B();
//     Runnable obj1=new A();
//     Runnable obj2=new B();

//     Runnable obj1=new Runnable()
//     {
//           public void run()
//           {
//                 for(int i=1;i<=5;i++)
//                 {
//                       System.out.println("Hello");
//                       try {
//                             Thread.sleep(10);
//                       }catch(InterruptedException e) {
//                             e.printStackTrace();
//                       }
//                 }
//           }
//           };

      Runnable obj1=()->
      {
```

```
            for(int i=1;i<=5;i++)
            {
                System.out.println("Hi");
                try {Thread.sleep(10);}catch(InterruptedException e)
{e.printStackTrace();}
            }
        };

        Runnable obj2=()->
        {
            for(int i=1;i<=5;i++)
            {
                System.out.println("Hello");
                try {Thread.sleep(10);}catch(InterruptedException e)
{e.printStackTrace();}
            }
        };

        Thread t1=new Thread(obj1);
        Thread t2=new Thread(obj2);

        t1.start();
        t2.start();
    }
```

```
}
```

Race condition

```
class Counter
{
	int count;
//	public void increment()
	public synchronized void increment()
	{
		count++;
	}
}

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

	Counter c=new Counter();

	Runnable obj1=()->
	{
//		for(int i=1;i<=1000;i++)
		for(int i=1;i<=10000;i++)
		{
			c.increment();
```

```
            }
        };

        Runnable obj2=()->
        {
//          for(int i=1;i<=1000;i++)
            for(int i=1;i<=10000;i++)
            {
                c.increment();
            }
        };
        Thread t1=new Thread(obj1);
        Thread t2=new Thread(obj2);
        t1.start();
        t2.start();

        t1.join();
        t2.join();

        System.out.println(c.count);
    }
}
```

Thread states

- New State
- Runnable State -> start() method
- Running State -> a thead is running with run() method
- Waiting State -> sleep(), wait() method
- Dead State

Through notify(), you will go to waiting state to runnable state.

From Running, Runnable state to dead state through stop() method.

Collection API

Collection API -> concept

Collection -> Interface

Collections -> classe with multiple methods

different type of data structures

Arraylist

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

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

//     Collection<Integer> nums= new ArrayList<Integer>();
```

```
//      Collection nums=new ArrayList();
        List<Integer> nums=new ArrayList<Integer>();
        nums.add(6);
        nums.add(5);
        nums.add(8);
        nums.add(2);
        //nums.add("5");

        System.out.println(nums.get(2));

        System.out.println(nums.indexOf(2));

//      for(int n:nums)
//      {
//              System.out.println(nums);
//      }
        for(Object n:nums)
        {
                int num=(Integer)n;
                System.out.println(nums);
        }
    }
}
```

Sets

```
import java.util.Set;
import java.util.HashSet;
import java.util.TreeSet;
import java.util.Collection;
import java.util.Iterator;

/*
import java.util.List;
import java.util.ArrayList;

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

      List<Integer> nums=new ArrayList<Integer>();
      nums.add(6);
      nums.add(5);
      nums.add(8);
      nums.add(2);
      nums.add(6);
      //nums.add("5");

      for(Object n:nums)
```

```
        {
            int num=(Integer)n;
            System.out.println(num+2);
        }
    }
}
*/
public class Demo {
    public static void main(String[] args){

//      Set<Integer> nums=new HashSet<Integer>();
//      Set<Integer> nums=new TreeSet<Integer>();
        Collection<Integer> nums=new TreeSet<Integer>();
        nums.add(62);
        nums.add(54);
        nums.add(82);
        nums.add(21);

        //nums.add("5");

        Iterator<Integer> values = nums.iterator();

        while(values.hasNext())
            System.out.println(values.next());
```

```
//     for(int n:nums)
//     {
//             System.out.println(n);
//     }
   }
}
```

Map

```
import java.util.HashMap;
import java.util.Map;
import java.util.Hashtable;

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

//     Map<String, Integer> students=new HashMap<>();
       Map<String, Integer> students=new Hashtable<>();

       students.put("Navin",56);
       students.put("Harsh",23);
       students.put("Sushil",67);
       students.put("Kiran",92);
       students.put("Harsh",45);
```

```
        System.out.println(students.keySet());

        for(String key : students.keySet())
        {
                System.out.println(key + ":" + students.get(key));
        }
    }
}
```

Comparator vs comparable

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Comparator;



//class Student implements Comparable<Student>
class Student
{
        int age;
        String name;

        public Student(int age, String name)
        {
```

```
		this.age=age;
		this.name=name;
	}

	public String toString() {
		return "Student [age=" + age + ", name=" +name +"]";
	}

//	public int CompareTo(Student that)
//	{
////		return 0;
//		if(this.age >that.age)
//			return 1;
//		else
//			return -1;
//	}
}

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

//	Comparator<Integer> com=new Comparator<Integer>()
//	{
```

```
//          public int compare(Integer i,Integer j)
//          {
//                  if(i%10 >j%10)
//                          return 1;
//                  else
//                          return -1;
//          }
//    };

//    List<Integer> nums= new ArrayList<>();
//    nums.add(43);
//    nums.add(31);
//    nums.add(72);
//    nums.add(29);

//    Comparator<Student> com=new Comparator<Student>()
//    {
//          public int compare(Student i,Student j)
//          {
//                  if(i.age >j.age)
//                          return 1;
//                  else
//                          return -1;
```

```
//            }
//    };

        Comparator<Student> com=(i,j) -> i.age > j.age?1:-1;

        List<Student> studs= new ArrayList<>();
        studs.add(new Student(21,"Navin"));
        studs.add(new Student(12,"John"));
        studs.add(new Student(18,"Parul"));
        studs.add(new Student(20,"Kiran"));

//      Collections.sort(nums);
//      System.out.println(nums);

        for(Student s:studs)
                System.out.println();

        Collections.sort(studs);
        for(Student s: studs)
                System.out.println(s);
    }
}
```

Need of stream API

```
import java.util.Arrays;
```

```
import java.util.List;
import java.util.stream.Stream;

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

       List<Integer> nums= Arrays.asList(4,5,7,3,2,6);

//     for(int i=0;i<nums.size();i++)
//     {
//           System.out.println(nums.get(i));
//     }


//     for(int n: nums)
//     {
//           System.out.println(n);
//     }

       nums.forEach(n -> System.out.println(n));


       int sum=0;
       for(int n:nums)
```

```
        {
            if(n%2==0)
            {
                n=n*2;
                sum=sum+n;
            }
        }

//      System.out.println(nums);
        System.out.println(sum);

    }
}
```

Foreach method

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Stream;
import java.util.function.Consumer;

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

        List<Integer> nums= Arrays.asList(4,5,7,3,2,6);
```

```
//     Consumer<Integer> con=new Consumer<Integer>() {
//
//             public void accept(Integer n)
//             {
//                     System.out.println(n);
//             }
//     };

        Consumer<Integer> con= n -> System.out.println(n);

        nums.forEach(n -> System.out.println(n));

 //     nums.forEach(null);
 //     nums.forEach(con);


        //nums.forEach(n -> System.out.println(n));

    }
}
```

Stream API

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Stream;
```

```
import java.util.function.Consumer;

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

       List<Integer> nums=Arrays.asList(4,5,7,3,2,6);

//     Stream<Integer> s1=nums.stream();
//     Stream<Integer> s2= s1.filter(n ->n%2==0);
//     Stream<Integer> s3= s2.map(n->n*2);
//     int result=s3.reduce(0,(c,e)->c+e);
//
//     s2.forEach(n -> System.out.println(n));
//     s3.forEach(n -> System.out.println(n));
//
//     s1.forEach(n-> System.out.println(n));
//     s1.forEach(n-> System.out.println(n));

       int result=nums.stream()
                              .filter(n-> n%2==0)
                              .map(n->n*2)
                              .reduce(0, (c,e)-> c+e);
       System.out.println(result);
```

```
    }
}
```

Map filter reduce sorted

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.stream.Stream;

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

        List<Integer> nums=Arrays.asList(4,5,7,3,2,6);

//      Predicate<Integer> p= new Predicate<Integer>() {
//              public boolean test(Integer n) {
//                      return n%2==0;
//                      if(n%2==0)
//                              return true;
//                      else
//                              return false;
//              }
```

```
//	};

//	Predicate<Integer> p= n-> n%2==0;


//	Function<Integer, Integer> fun= new Function<Integer,Integer>() {
//		public Integer apply(Integer n) {
//			return n*2;
//		}
//	};

//	Function<Integer,Integer> fun= n-> n*2;


//	int result=nums.stream()
//				.filter(n-> n%2==0)
//				.map(n->n*2)
//				.reduce(0, (c,e)-> c+e);
//	System.out.println(result);

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

	Stream<Integer> sortedValues = nums.parallelStream()
```

```
                .filter(n-> n%2==0)
                .sorted();

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

Wrapper class

Wrapper Classes

int -> Integer

char -> Character

double -> Double

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

        int num=7;
//      Integer num1=new Integer(8);
//      Integer num1=8;

//      Integer num1=new Integer(num);     //boxing
```

```
        Integer num1=num;               // autoboxing

        int num2=num1.intValue();       // unboxing

//      System.out.println(num1);
        System.out.println(num2);

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

        System.out.println( num3+2);
    }
}
```