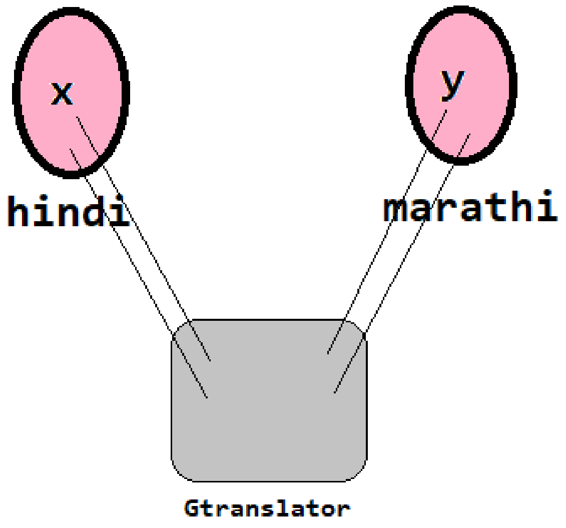
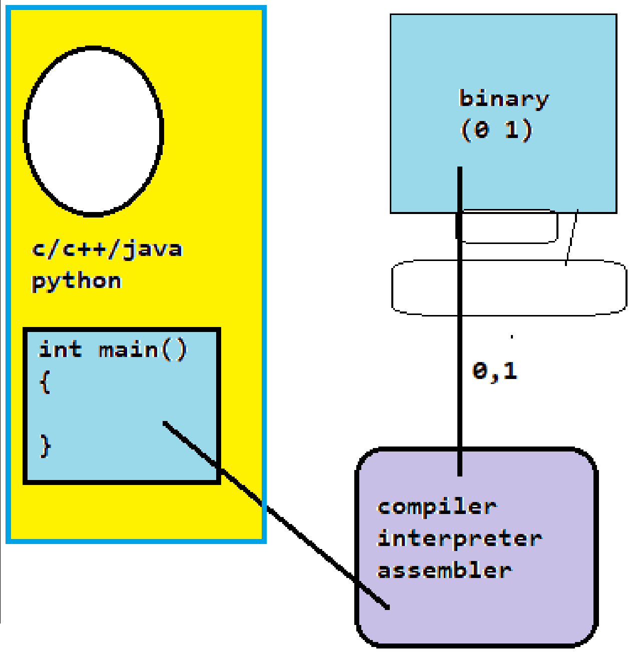
Core java

- Java to standard edition(J2SE) : core Java
- Java standard edition(JSE): core Java
- Java to enterprise edition(J2EE) : adv Java
- Java to enterprise edition(JEE) : adv Java
- Java is a Programming language
- What is a programming language?





```
int main()
{

}
```

Why computer understands only 0 1?

Coz of electric current(ON-1 , OFF-0)

Types of Programming language?

- **Low level**

- Machine language(0,1)
    - Ada lovelace
    - Charles babbage(Father of Computer)

Ex. int a=4;

- 1010101 101010101 1010101 100 : ML
- mov(a,100)
- int a = 4 : high/middle

MNEMONICS : human readable commands

Add,mul,sub,mov,div..etc

- Machine language: No translator
- Assembly ----->**Assembler**------>binary

- Assembly language

➢High level

It introduce English like statement

(A-Z,a-z,0-9,#$^%..etc)

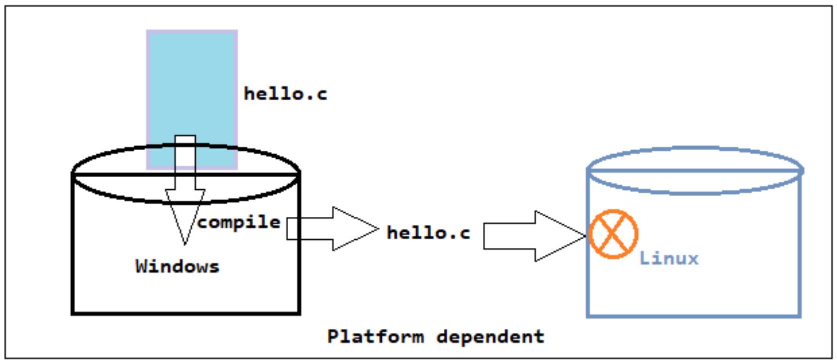
- Java,python,sql..etc

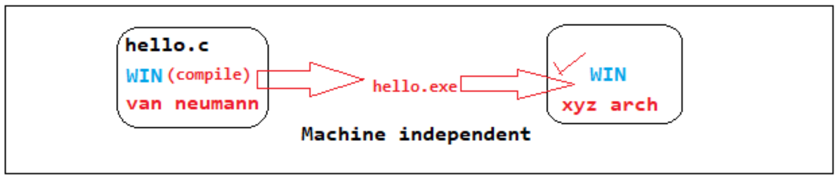
➢Middle level

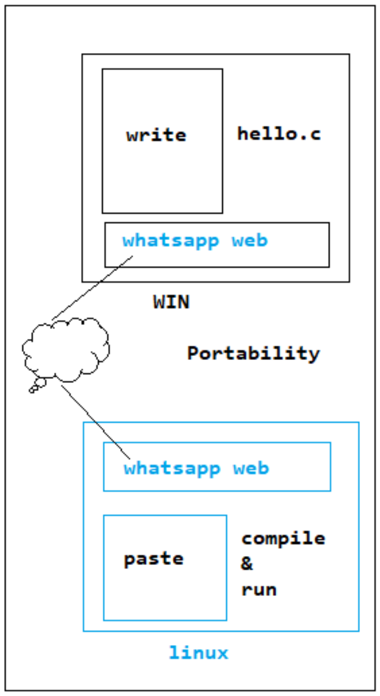
Language which can directly communicate with microcontroller.

- C and C++

C and C++







1.platform dependency
2.pointer (disadvantage for internet application)

Java

How to check whether java is installed or not

- ❖ Open cmd
- ❖ Java --version

1. Java development kit(JDK-19)

https://download.oracle.com/java/19/latest/jdk-19_windows-x64_bin.exe (sha256)

2. IDE: Eclipse(photon)

https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/photon/R/eclipse-jee-photon-R-win32-x86_64.zip

How to create first java project in eclipse

❖Select workspace

❖Change perspective to java (by default: javaEE)

❖Create java project(file->new->java project)

Give your project name and click on finish->don't create

❖Create one class with main method(right click on src->new->class)

```
package demo;
public class Launch {
     public static void main(String[] args) {
          // single line comment

          /*
             multi line comment
          */

          System.out.println("hello");//sysout ctrl space
     }
}
```

//public : access modifier

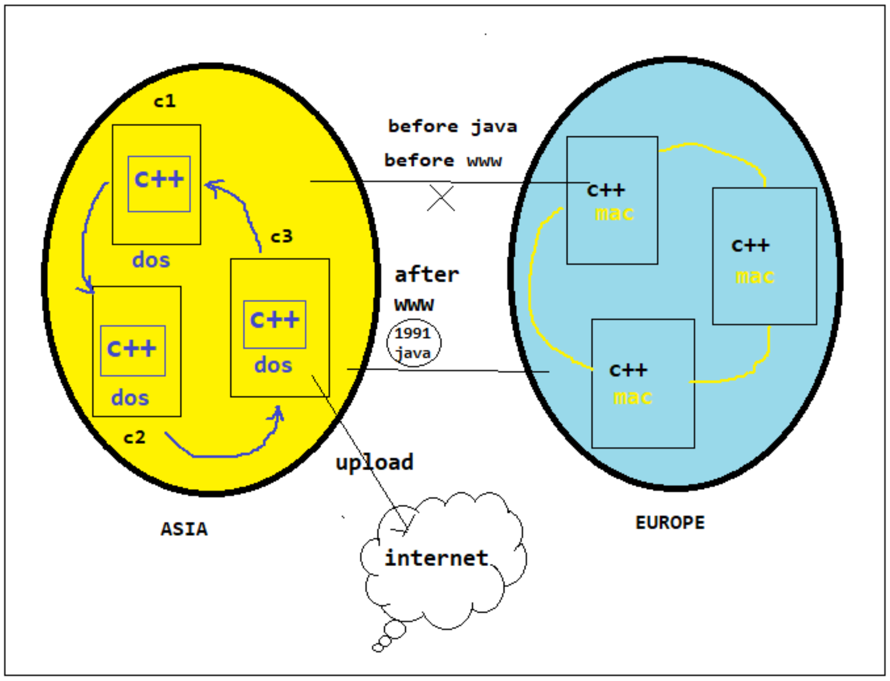
//static : non access modifier

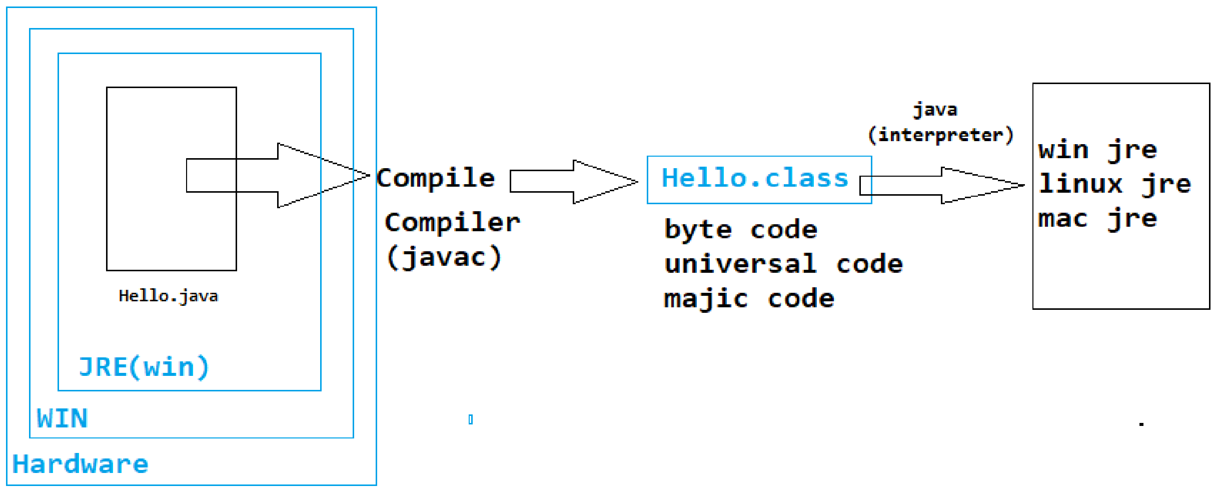
//String[] args : array

//System : java.io class

Java History

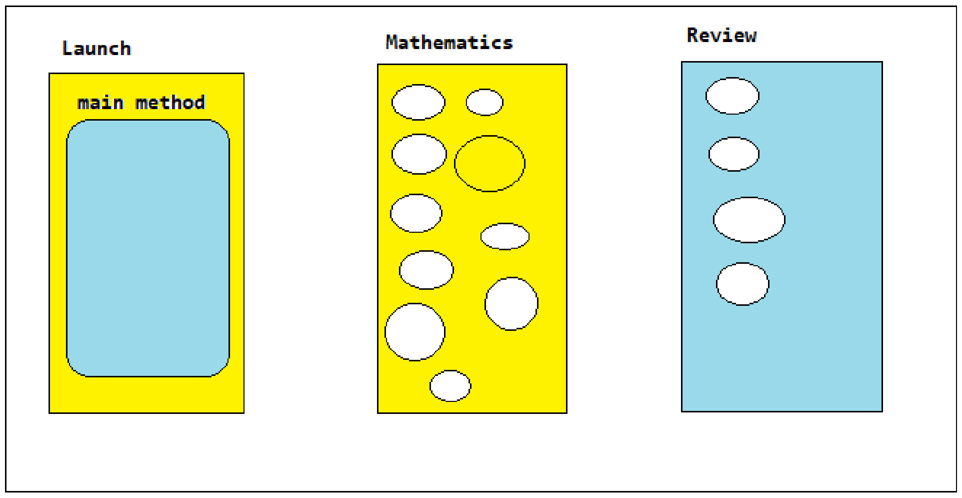
- Sunmicrosystem
- Green Project(Java): Green Team(James goslin,mike sheredon,Patrick naughton..and mamy more)
- Set of box
- Java --->C+
- Popularity: www





What is class?

- class is a keyword in java which is used to create class
- class is a basic concept of OOP
- All the concepts of OOP taken from realworld
- It is used to categorize problems

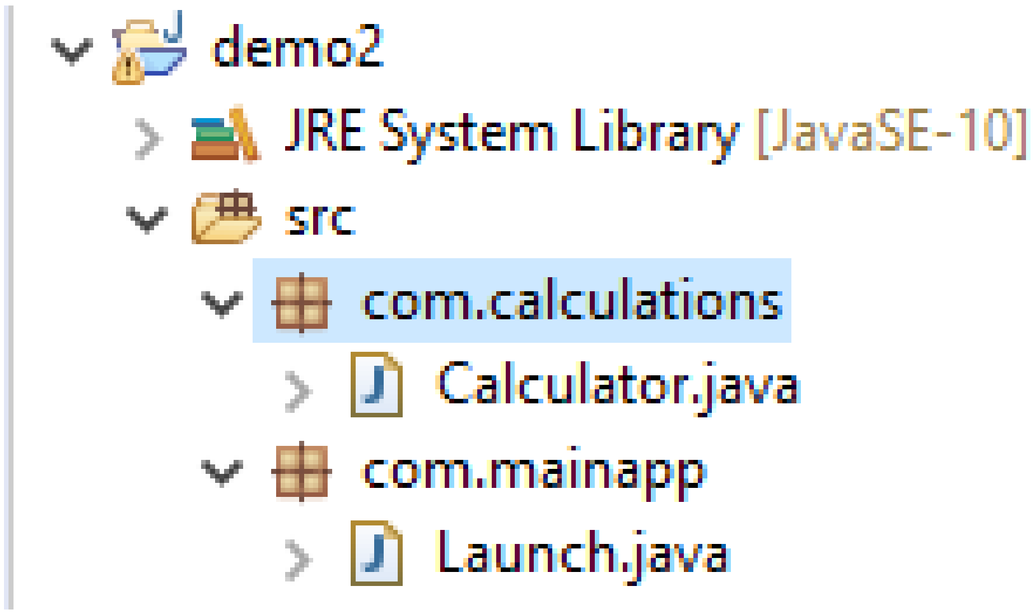


→Library : Java ,python

→Packages

→Classes and interfaces

→methods



Data types

1.primitive DT

2.non primitive DT

- byte : -128 +127 **size 1 byte**
- short
- int
- long
- **float**
- **double**
- **char**
- **boolean**

```
package com.mainapp;
public class Launch {
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        byte b=100;
        short st=1000;
        int i=10000;
        long l=100000;

        float f=10.55F;  //f or F
        double d=10.45;
        char c='%';

        boolean bool1=true;
        //1 //case sensitive  1 bit
        boolean bool2=false;
        //0 //case sensitive
    }
}
```

byte: -128 to +127 why? (size=1byte)

```
127 1
63  1
31  1
15  1
7   1
3   1
1

127: 1 1 1 1 1 1 1
left most bit 0->positive num
left most bit 1->negative num

 0 1 1 1 1 1 1 1   8bit
```

```
128 0
64  0
32  0
16  0
8   0
4   0
2   0
1

128: 1 0 0 0 0 0 0 0
+128 0 1 0 0 0 0 0 0 0  9bit
```

```
129 1
64  0
32  0
16  0
8   0
4   0
2   0
1

        129 : 1 0 0 0 0 0 0 1
  one's com : 0 1 1 1 1 1 1 0
                            1
  two's com : _______________
              0 1 1 1 1 1 1 1
      -129: 1 0 1 1 1 1 1 1 1  9bit
```

Conditional Statement

- if
- if else
- nested if else
- else if ladder/clause

```
package com.mainapp;
public class Launch {

	public static void main(String[] args) {

		//int age; //Declaration
		//age=50;  //Initialization
		int age=50;
		if(age<40)
		{
			System.out.println("eligible");
		}
		System.out.println("abcd");
	}
}
```

Program2:

```
package com.mainapp;
public class Launch {

	public static void main(String[] args) {

		//int age; //Declaration
		//age=50;  //Initialization

		int age=50;
		if(age<40)
		{
```

```
            System.out.println("eligible");

        }
        if(age>40)
        {
            System.out.println("not eligible");
        }
    }
}
```

Project3:

```
package com.mainapp;
public class Launch {

    public static void main(String[] args) {

        //int age; //Declaration
        //age=50;  //Initialization

        int age=50;
        if(age<40)
        {
            System.out.println("eligible");
        }
        else
        {
            System.out.println("not eligible");
        }
    }
}
```

Nested if-else

```
package com.mainapp;
public class Launch {
	public static void main(String[] args) {

		//age>18 , pincode=1111 , per>40

		int age=100;
		int pincode=111;
		int per=300;
		int support=1;
		if(age>18)
		{
			if(pincode==1111)
			{
				if(per>40)
				{
					System.out.println("Eligible");
				}
				else
				{
					if(support==1)
					{
						System.out.println("eligible");
					}
					else
					{
		System.out.println("not eligible : PERCENTAGE");
					}
				}
			}
			else
			{
		System.out.println("not eligible : PINCODE");
			}
		}
		else
		{
```

```
                System.out.println("not eligible : AGE");
            }
        }
    }
```

else if ladder

```
package elseif;

public class Launch {

    public static void main(String[] args) {

        int age=8;

        if(age==10)
        {
            System.out.println("ten");
        }
        else if(age==11)
        {
            System.out.println("eleven");
        }
        else if(age<10)
        {
            System.out.println("one");
        }
        else
        {
            System.out.println("out of range");
        }


    }

}
```

Logical operator

&& and

|| or

! not

&& operator

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int age=100;
		int pincode=1111;
		int per=300;

		if(age>18 && pincode==1111 && per>40)
		{
			System.out.println("eligible");
		}
		else
		{
			System.out.println("not eligible");
		}

	}

}
```

|| operator

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int age=100;
		int per=30;

		if(age>18 || per>40)
		{
			System.out.println("eligible");
		}
		else
		{
			System.out.println("not eligible");
		}

	}

}
```

! operator

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int age=100;
		int per=40;

		if(age>180 || per!=30)
		{
```

```
                System.out.println("eligible");
            }
            else
            {
                System.out.println("not eligible");
            }
        }
}
```

Loops

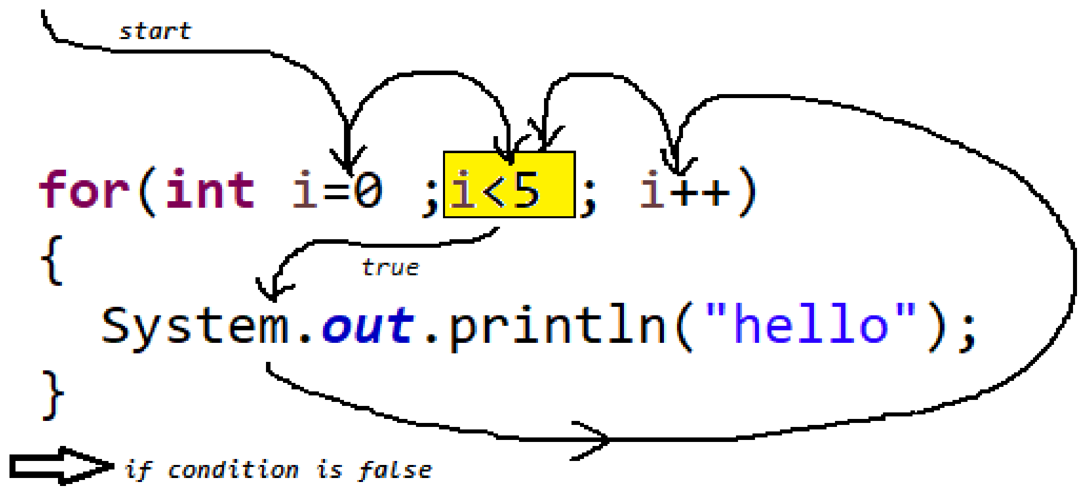
Used for code repetition

1.for

2.while

3.do-while

4.for-each loop



```
for(int i=0 ;i<5 ; i++)
{
  System.out.println("hello");
}
```

```java
package elseif;
public class Launch {

	public static void main(String[] args) {
		int i=0;
		for(  ;i<5 ; )
		{
		  i++;
		  System.out.println("hello");
		}
	}
}
```

```java
package elseif;
public class Launch {

	public static void main(String[] args) {
		int i=0;
		for(  ;i<5 ; )  //0 2 4
		{
		  System.out.println("hello");
		  i=i+2;
		}
	}
}
```

```java
package elseif;
public class Launch {

	public static void main(String[] args) {
		int i=0;
		for(  ;i<5;i++ )  //0 2 4
		{
		  System.out.println("hello");
		  i++;
		}
	}
}


package elseif;
public class Launch {

	public static void main(String[] args) {
		int i=10;
		for(  ;i>5;i--)  //10-6
		{
		  System.out.println("hello");

		}
	}
}


package elseif;
public class Launch {
	public static void main(String[] args) {

		for(int i=0;i<5; ++i)
		{
		  System.out.println("hello");
		}
	}
}


package elseif;
public class Launch {
	public static void main(String[] args) {


		int a=10;
		System.out.println(a++); //10(p)--11
		System.out.println(++a); //11-12(p)
```

```
     }
}
package elseif;
public class Launch {

     public static void main(String[] args) {

          //NESTED LOOP
          for(int i=0;i<5;i++)
          {
               for(int j=0;j<5;j++)
               {
                 System.out.println(i+"hello"+j);
               }
          }
     }
}


package elseif;
public class Launch {

     public static void main(String[] args) {

          //NESTED LOOP
          for(int i=0;i<5;i++)
          {
               for(int j=0;j<i;j++)
               {
                 System.out.println(i+"hello"+j);
               }
          }
     }
}
```

Task1

int a=10;

int b=14;

output:

10+11+12+13+14=60

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int a=10;
		int b=11;
		int sum=0;
		if(b<=a)
		{
		System.out.println("b should be greater than a");
		}
		else
		{
			for(int i=a;i<=b;i++)
			{
				sum=sum+i;
				System.out.print(i);
				if(i!=b)
				{
				System.out.print("+");
				}
			}
			System.out.println("="+sum);
		}
	}
}
```

Task1

int a=10;

int b=20;

output:

12+14+16+18=60

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int a=10;
		int b=20;
		int sum=0;
		if(b<=a)
		{
		System.out.println("b should be greater than a");
		}
		else
		{
			for(int i=a;i<=b;i++)
			{
				if(i!=a && i!=b) //sum
				{
					if(i%2==0)//even
					{
						sum=sum+i;
						System.out.print(i);
						int k;
					       if(b%2==0)
					       {
```

```
                              k=b-2;
                             }
                              else
                              {
                                  k=b-1;
                              }
                          if(i!=k)
                          {
                          System.out.print("+");
                          }
                          }
                      }
                  }
                  System.out.println("="+sum);
              }
          }
      }
```

While loop

```
package elseif;
public class Launch {

    public static void main(String[] args) {

        int a=1;
        while(a<10)
        {
            System.out.println("hello");
            a++;
        }
    }
}
```

do-While loop

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int a=1;
		do
		{
			System.out.println("hello");
			a++;
		}
		while(a<10);
	}
}
```

Switch Statement

It used to create Menu

```
package elseif;
public class Launch {

	public static void main(String[] args) {

		int choice=10;
		switch (choice)
		{

		    case 1: System.out.println("one");
```

```
                 break;

                case 2: System.out.println("two");
                 break;

                case 3: System.out.println("three");
                 break;

                case 4: System.out.println("four");
                 break;

                case 5: System.out.println("five");
                 break;

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

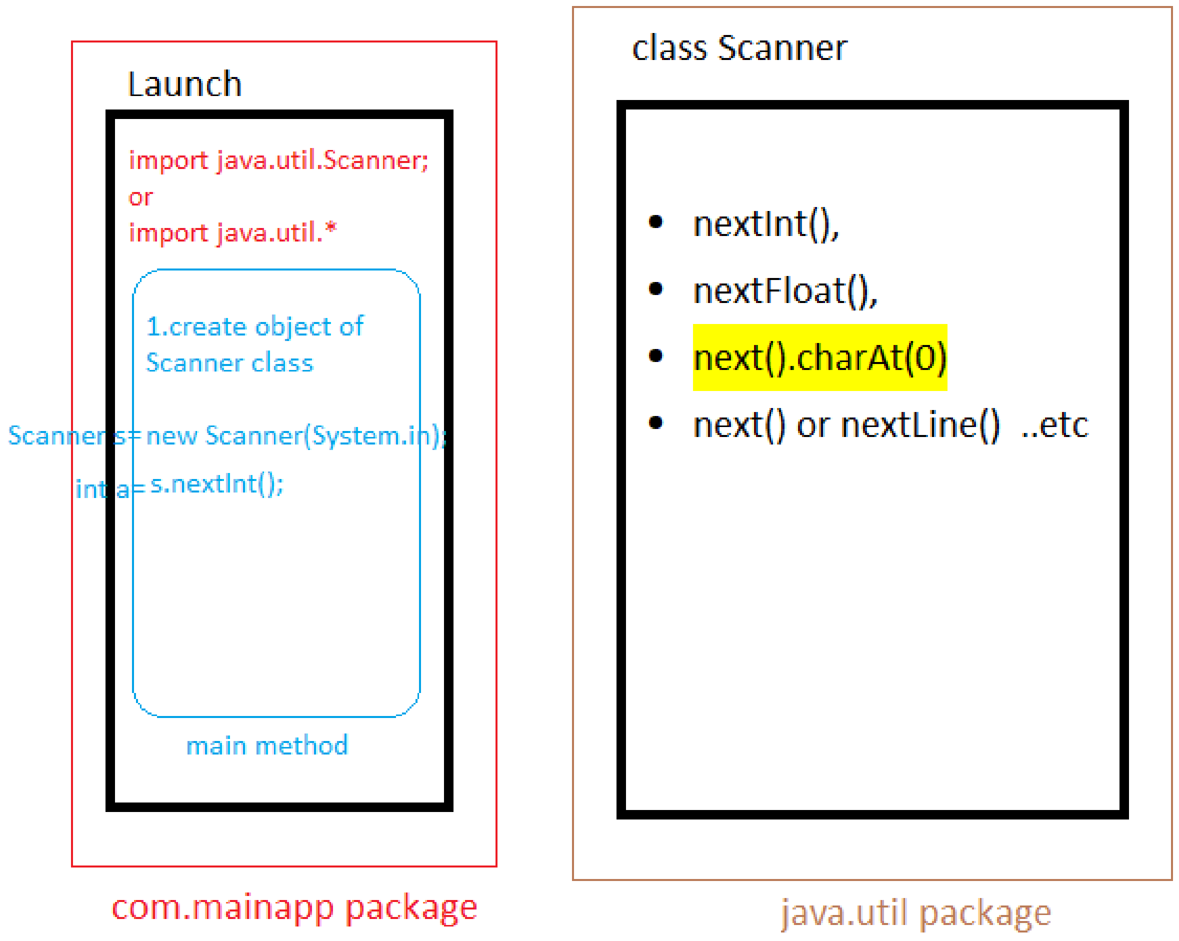
USER INPUT

C prog: scanf()->Function

Java:

- nextInt(),
- nextFloat(),
- next().charAt(0)
- next() or nextLine() ..etc

```java
package com.mainapp;

import java.util.Scanner;

public class Launch {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner s = new Scanner(System.in);
		int a=s.nextInt();
		System.out.println(a);

	}

}
```



Switch Case

```
package switchcase;

public class Launch {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		int a=20;
		switch (a)
		{
		  case 10: System.out.println("hello");
		  break;
		  case 2: System.out.println("hii");
		  break;
		  default:  System.out.println("default");
		  break;
		}
	}
}
```

Task

Welcome to my calculator

Enter first digit: user input (Ex. 10)

Enter second digit: user input (Ex. 20)

Press 1: add

Press 2: sub

Press 3: div

Press 4: mul

Enter choice: user input (Ex. 1)

Addition is : 30

Do you want to continue Y/N

```java
package switchcase;
import java.util.Scanner;
public class Launch {
     public static void main(String[] args) {

          Scanner sc = new Scanner(System.in);
          while(true)
          {

          System.out.print("Enter first digit: ");
          float a = sc.nextFloat();
          System.out.print("Enter second digit: ");
          float b = sc.nextFloat();

          System.out.print("Press 1: add");
          System.out.print("Press 2: sub");
          System.out.print("Press 3: div");
          System.out.println("Press 4: mul");
          System.out.print("Enter choice: ");
          int c = sc.nextInt();

          switch (c) {

          case 1: System.out.println("addition is : "+(a+b));
          break;
          case 2: System.out.println("substraction is : "+(a-b));
          break;
          case 3: System.out.println("division is : "+(a/b));
          break;
          case 4: System.out.println("multiplication is : "+(a*b));
          break;

          default: System.out.println("unknown entry");
          break;
          }
          System.out.println("Do you want to continue: Y/N");
          char cc = sc.next().charAt(0);
          if(cc=='N' || cc=='n')
          {
               break;
          }

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

}

Method

What is Method

method is nothing but a block in which we can write reusable logic

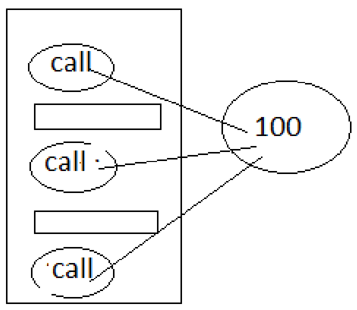
->Inbuil method/predefined method

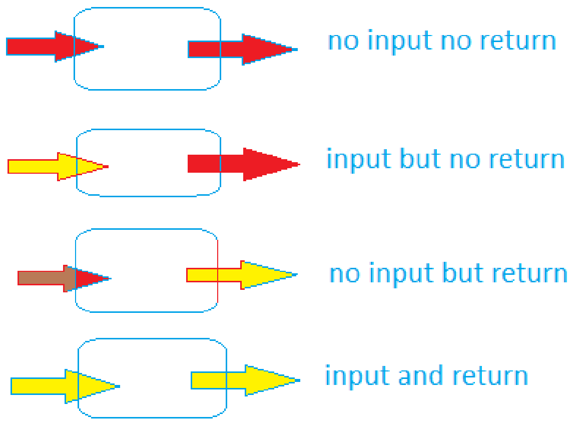
nextInt();

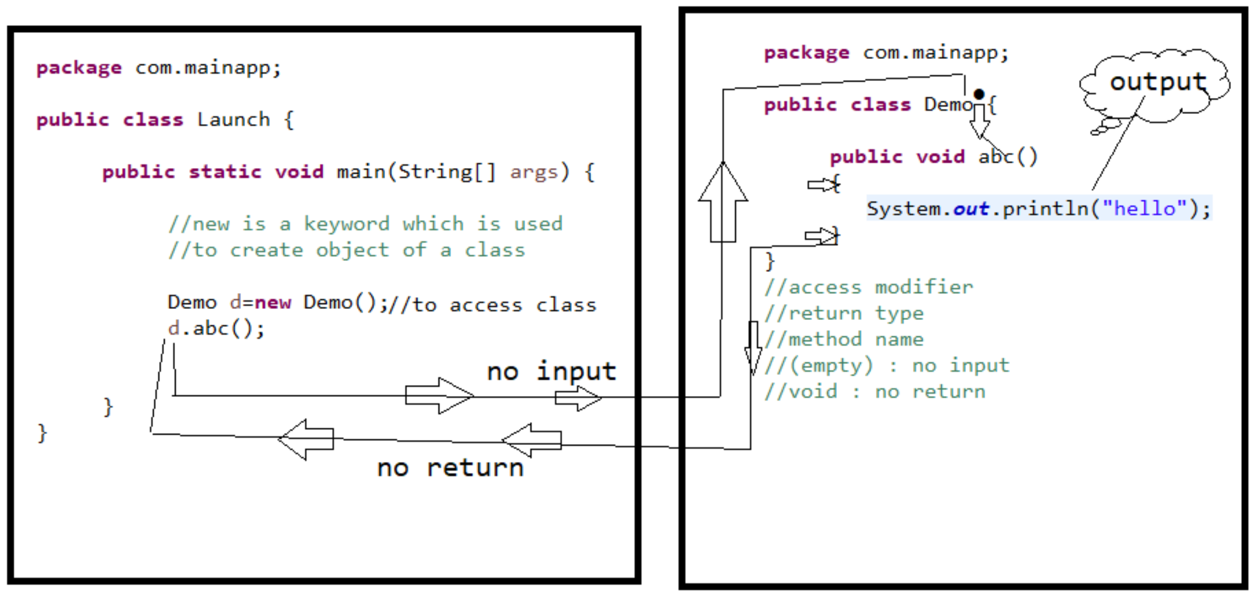
->User defined method

raju() {}

main method

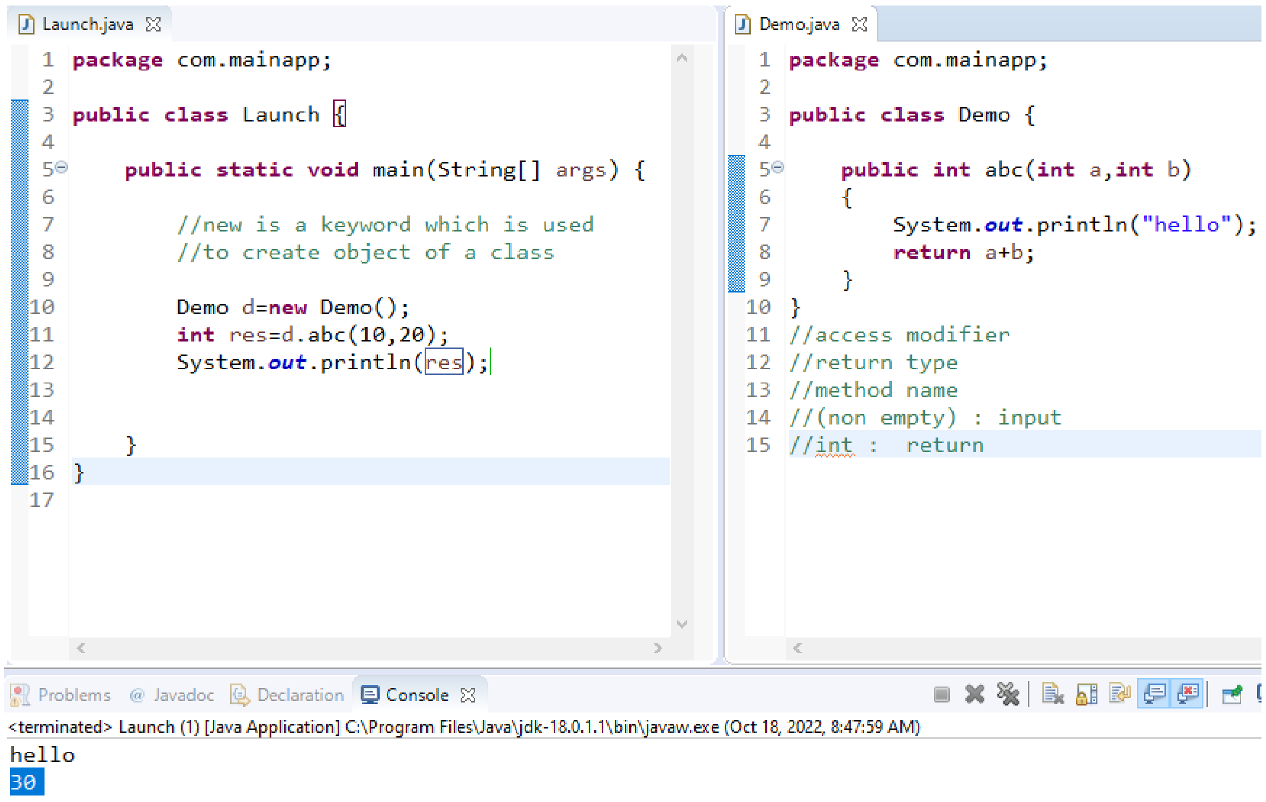






```
package com.mainapp;

public class Launch {

	public static void main(String[] args) {

		//new is a keyword which is used
		//to create object of a class

		Demo d=new Demo();//to access class
		d.abc();

	}
}
```

```
package com.mainapp;

public class Demo {

	public void abc()
	{
		System.out.println("hello");
	}
}
//access modifier
//return type
//method name
//(empty) : no input
//void : no return
```


Launch.java

```java
package com.mainapp;

public class Launch {

    public static void main(String[] args) {

        //new is a keyword which is used
        //to create object of a class

        Demo d=new Demo();
        int res=d.abc(10,20);
        System.out.println(res);


    }
}

```

Demo.java

```java
package com.mainapp;

public class Demo {

    public int abc(int a,int b)
    {
        System.out.println("hello");
        return a+b;
    }
}
//access modifier
//return type
//method name
//(non empty) : input
//int :  return
```

```
<terminated> Launch (1) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Oct 18, 2022, 8:47:59 AM)
hello
30
```

Task

Welcome to my calculator

Enter first digit: user input (Ex. 10)

Enter second digit: user input (Ex. 20)

Press 1: int res=add(d1,d2);
sop(res);

Press 2: sub(d1,d2)

Press 3: div (d1,d2)

Press 4: mul(d1,d2)

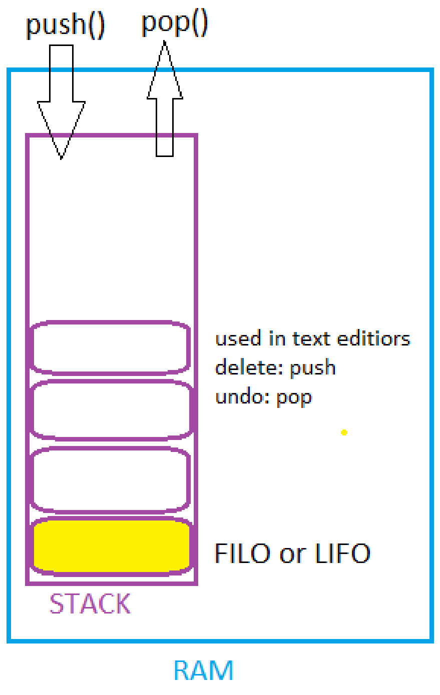
Enter choice: user input (Ex. 1)

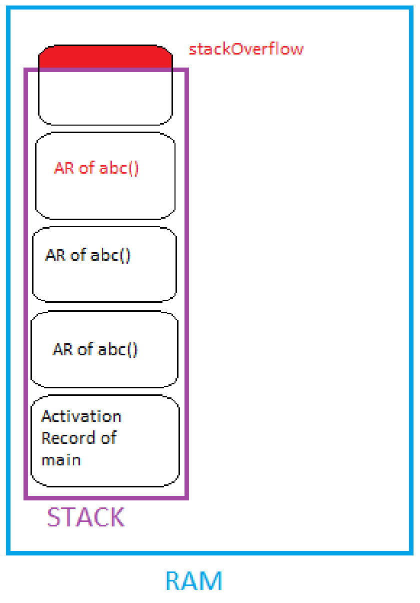
Addition is : 30

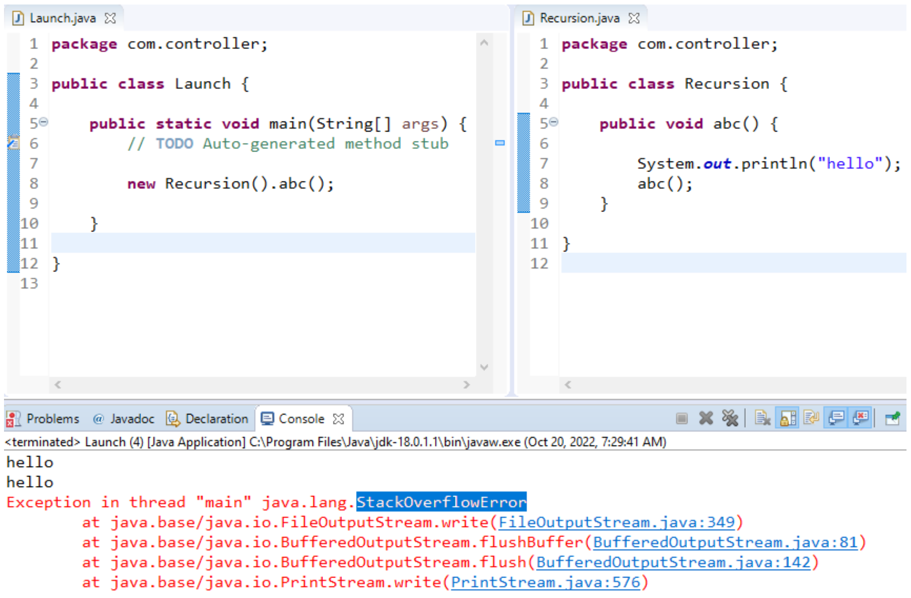
Do you want to continue Y/N

RECURSION

When a method call itself called recursion







Launch.java

```java
package com.controller;

public class Launch {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        new Recursion().abc();

    }

}
```

Recursion.java

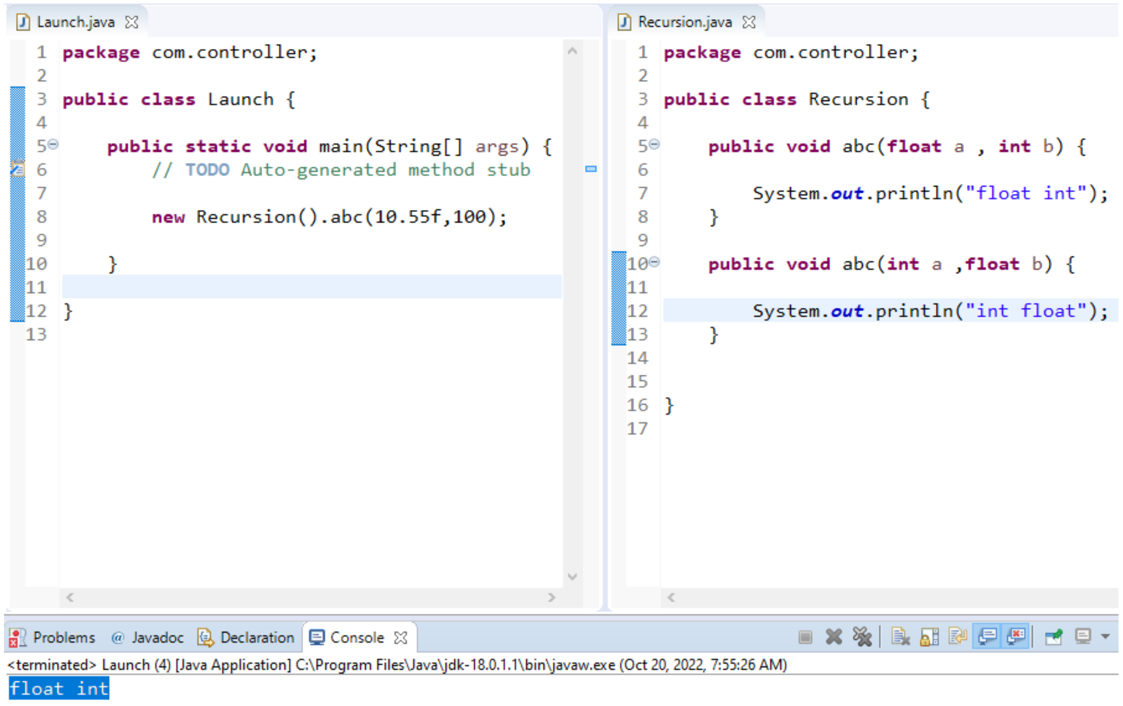
```java
package com.controller;

public class Recursion {

    public void abc() {

        System.out.println("hello");
        abc();
    }

}
```

Console

```
<terminated> Launch (4) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Oct 20, 2022, 7:29:41 AM)
hello
hello
Exception in thread "main" java.lang.StackOverflowError
	at java.base/java.io.FileOutputStream.write(FileOutputStream.java:349)
	at java.base/java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:81)
	at java.base/java.io.BufferedOutputStream.flush(BufferedOutputStream.java:142)
	at java.base/java.io.PrintStream.write(PrintStream.java:576)
```

Method Overloading

- What is the diff between method overloading and overriding
- Multiple method with same inside same class
    - ➢ No of Parameter
    - ➢ Diff data type
    - ➢ Diff Sequence



Launch.java

```java
package com.controller;

public class Launch {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        new Recursion().abc(10.55f,100);

    }

}
```

Recursion.java

```java
package com.controller;

public class Recursion {

    public void abc(float a , int b) {

        System.out.println("float int");
    }

    public void abc(int a ,float b) {

        System.out.println("int float");
    }


}
```

Console

```
<terminated> Launch (4) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Oct 20, 2022, 7:55:26 AM)
float int
```

Types of variable

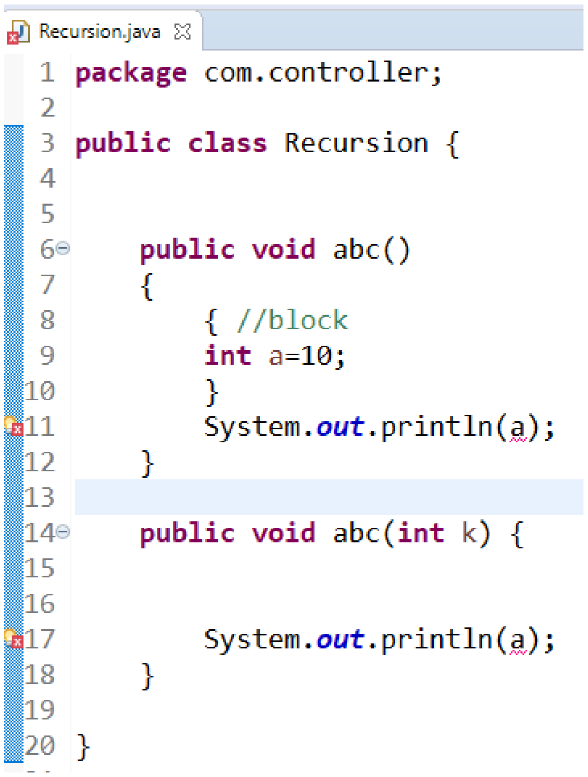
1.local variable

- Inside the method
- Scope: within the block
- We can't use Local variable before initialization

2.instance variable

- Inside the class but outside the method
- Scope: depends on Access modifier(public,private,protected,default)

Ex: Local variable(Blocked Scope)



Recursion.java

```
package com.controller;

public class Recursion {


    public void abc()
    {
        { //block
        int a=10;
        }
        System.out.println(a);
    }

    public void abc(int k) {


        System.out.println(a);
    }

}
```

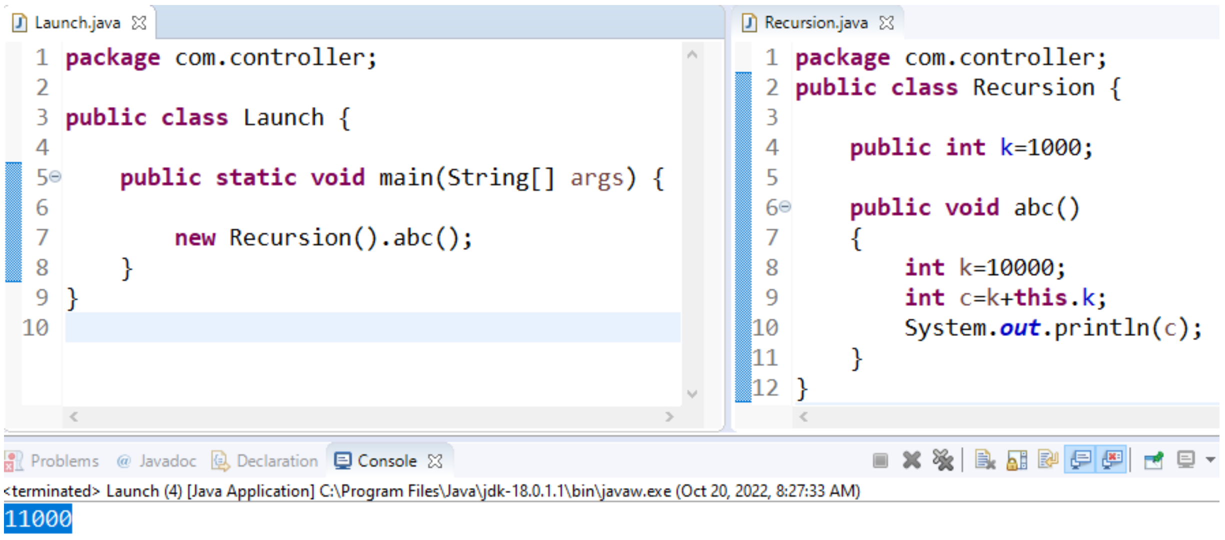
Value of Default instance variable

- We can use instance variable before initialization and in such case it will provide default value

1. int	0
2. float	0.0
3. char	whitespace
4. boolean	false

- In a same class we can create instance var and local var with same name
- To diff between local var and instance var we can use “this” keyword.
- this→it provides current class object

Note: what is the diff between this this() and super super()



Launch.java

```java
package com.controller;

public class Launch {

    public static void main(String[] args) {

        new Recursion().abc();
    }
}
```

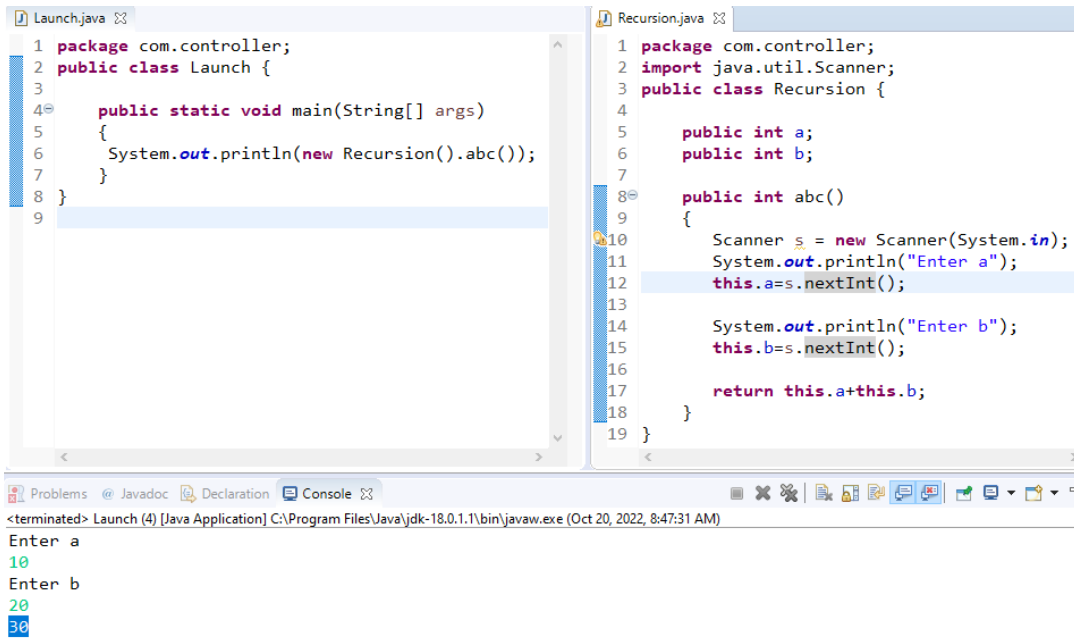
Recursion.java

```java
package com.controller;
public class Recursion {

    public int k=1000;

    public void abc()
    {
        int k=10000;
        int c=k+this.k;
        System.out.println(c);
    }
}
```

Console

```
<terminated> Launch (4) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Oct 20, 2022, 8:27:33 AM)
11000
```

TASK: WAP to insert value inside instance var



Launch.java

```java
package com.controller;
public class Launch {

    public static void main(String[] args)
    {
     System.out.println(new Recursion().abc());
    }
}
```

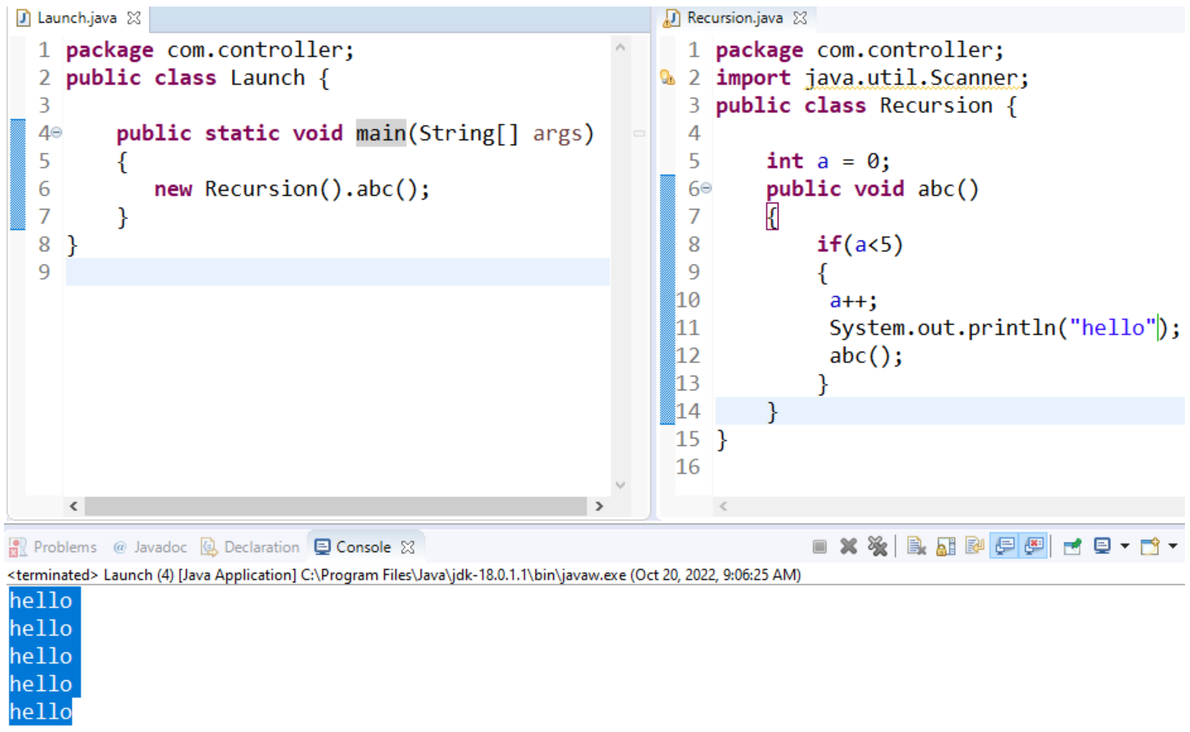
Recursion.java

```java
package com.controller;
import java.util.Scanner;
public class Recursion {

    public int a;
    public int b;

    public int abc()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a");
        this.a=s.nextInt();

        System.out.println("Enter b");
        this.b=s.nextInt();

        return this.a+this.b;
    }
}
```

Console

```
<terminated> Launch (4) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Oct 20, 2022, 8:47:31 AM)
Enter a
10
Enter b
20
30
```

Task : WAP to put some limit on infinite method calling without using loops



Launch.java

```java
package com.controller;
public class Launch {

    public static void main(String[] args)
    {
        new Recursion().abc();
    }
}
```

Recursion.java

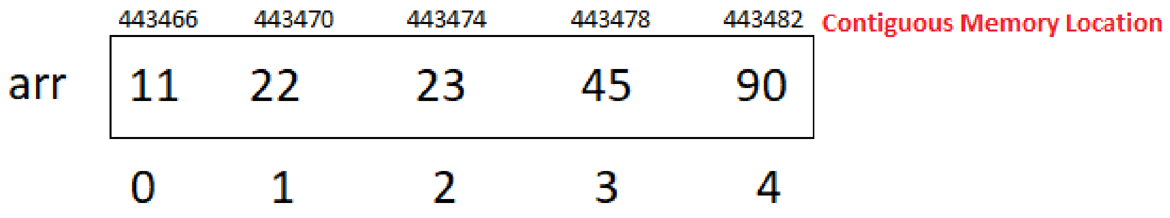
```java
package com.controller;
import java.util.Scanner;
public class Recursion {

    int a = 0;
    public void abc()
    {
        if(a<5)
        {
         a++;
         System.out.println("hello");
         abc();
        }
    }
}
```

Console

```
<terminated> Launch (4) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Oct 20, 2022, 9:06:25 AM)
hello
hello
hello
hello
hello
```

Array in Java

- Array is a basic data structure(**WILD ANIMAL**)
- Stack , ArrayList, Queue….etc (**DOMESTIC ANIMAL**)
- Array is used to store multiple data in a single variable
- Array uses contiguous memory location



- We can't change size of Array at Runtime
- We can't store **heterogeneous**(diff data type) Data

We can create Array in two ways

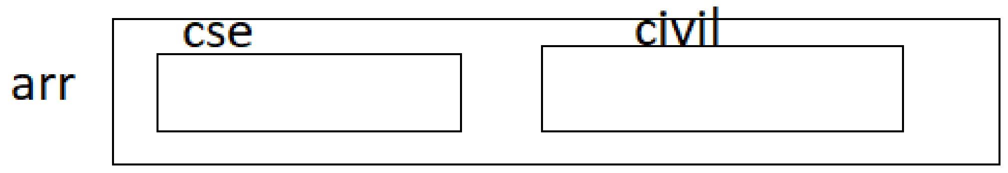
1.with new keyword (Dynamic array : user)

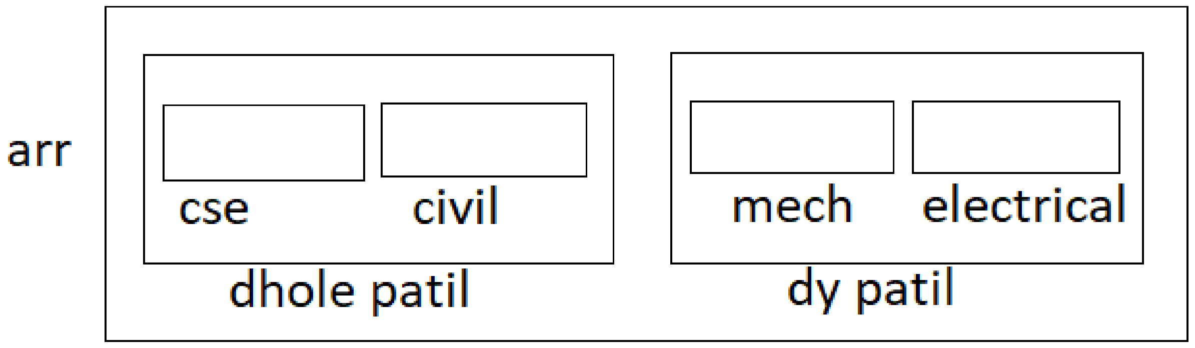
2.without new keyword(static array: programmer)

- Single Dimension -1D
- Multi Dimension-2D,3D,4D,5D…etc

arr 50 std-> 1D -> array [std1,std2,std3…std50]

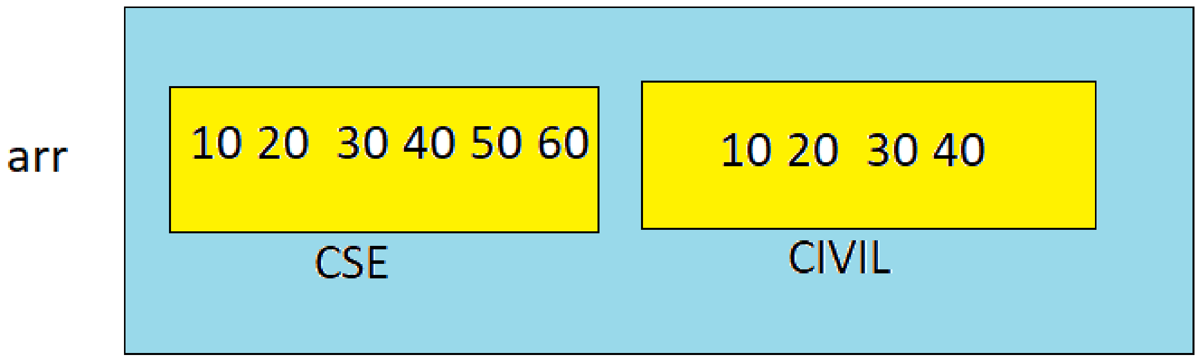
2 dep : cse(50std) , civil(50std)





Jagged Array

Jagged Array (MultiDimensional)



We can add element from user in static array but it is not preferred approach coz it takes many variables which is against the Array

```
package array;
import java.util.Scanner;

public class Launch {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter array element");
		int a=sc.nextInt();

		int arr[]= {a}; //empty array, static array
		//a is not size of array but it is zeroth element
		System.out.println(arr[0]);

	}
}
```

Print Array using for loop

```
package array;
import java.util.Scanner;
public class Launch {

	public static void main(String[] args) {


		int arr[]= {11,22,33,44,55,66};
		System.out.print("[");
		for(int i=0; i<6 ; i++)
```

```
        {
            System.out.print(arr[i]);
            if(i!=5)
                System.out.print(" ");
        }
        System.out.print("]");
//      System.out.println(arr[0]);
//      System.out.println(arr[1]);
//      System.out.println(arr[2]);
//      System.out.println(arr[3]);
//      System.out.println(arr[4]);
//      System.out.println(arr[5]);


    }
}
```

Iteration: going through each elements

Length of an ARRAY

```
package array;
import java.util.Scanner;
public class Launch {
    public static void main(String[] args) {


        int arr[]= {11,22,33,44,55,66};
        //last index=length-1
        System.out.println(arr.length);
        for(int i=0;i<arr.length;i++)
        {
            System.out.println(arr[i]);
        }

    }
}
```

TASK

WAP to sum up all the elements array given below

```
int arr[]= {11,22,33,44,55,66};
```

```
package array;
public class Launch {

	public static void main(String[] args) {

		int arr[]= {11,22,33,44,55,66};
		int sum=0;
		for (int i = 0; i < arr.length; i++) {
			sum=sum+arr[i];
		}
		System.out.println(sum);
		}
	}
}
```

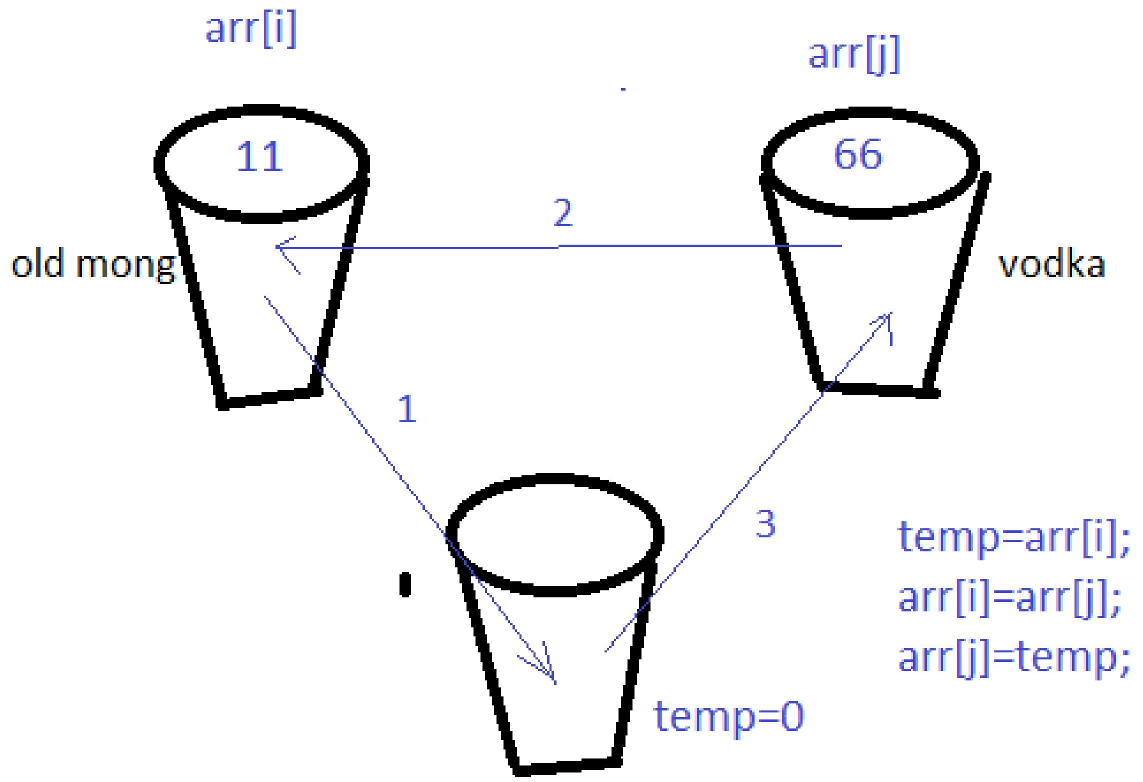
TASK

WAP to reverse array

```
int arr[]= {11,22,33,44,55,66};
```

output: 66 55 44 33 22 11

```
package array;
public class Launch {
	public static void main(String[] args) {

		int arr[]= {11,22,33,44,55,66};
		//your code
		int temp=0;
for(int i=0;i<arr.length/2;i++) // 0 1
{
     for(int j=arr.length-1-i;j>=arr.length/2;j--) //4 3 2
     {
          temp=arr[i];
          arr[i]=arr[j];
          arr[j]=temp;
          break;
     }
}

          for (int i = 0; i < arr.length; i++) {
               System.out.print(arr[i]+" ");
          }
     }
}
```



temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;

Dynamic 1D array(with new keyword)

```
package array;
public class Launch {
	public static void main(String[] args) {

		//Dynamic 1D array(new keyword)


		int arr[]=new int[5];
		arr[0]=10;
		arr[1]=20;
		arr[2]=40;
		arr[3]=50;
		arr[4]=60;

		System.out.println(arr[3]);

	}
}
```

Note:

Rename all same variable at once: alt shift R

```
package array;

import java.util.Scanner;

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

		//Dynamic 1D array(new keyword)
		Scanner s = new Scanner(System.in);
		System.out.println("Enter array size");
		int size = s.nextInt();//alt shift L
		int arr[]=new int[size];

		//user input
		for (int i = 0; i < arr.length; i++) {
			System.out.print("arr["+i+"]=");
			arr[i]=s.nextInt();
		}
		//print
		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i]+" ");

		}
	}
}
```

American Standard Code for Information Interchange

Types casting

```
package array;
import java.util.Scanner;
public class Launch {
	public static void main(String[] args) {

		//ASCII value
		//A-65 Z-90
		//a-97 z-122
		int a=99;
		//char c=(char)a;
		System.out.println((char)a);
	}
}
```

TASK

WAP to store ASCII value of alphabets (A-Z,a-z) inside an array

[65-90 97-122]

```java
package array;
import java.util.Scanner;
public class Launch {
    public static void main(String[] args) {

        //ASCII value  65 90 {91-96} 97-122
        byte b[]=new byte[52];
        int k=0;
        for(byte i=65 ;i<=122 ; i++)
        {
            if(i>=91 && i<=96)
            {
                continue;
            }
            b[k]=i;
            k++;
        }

        for (int i = 0; i < b.length; i++) {
            System.out.print(b[i] +" ");
        }
    }
}
```

2D static array

```
package array;
public class Launch {
    public static void main(String[] args) {

        //2D static array : set of 1d array
        int arr[][]= {

                {11,22,33,44},  //dep 0
                {33,44,55,66},  //dep 1
                {55,66,77,88}   //dep 2
        };
        System.out.println(arr[1][2]);
    }
}
```

For 2D array, 1-d array is an element

2D static jagged array

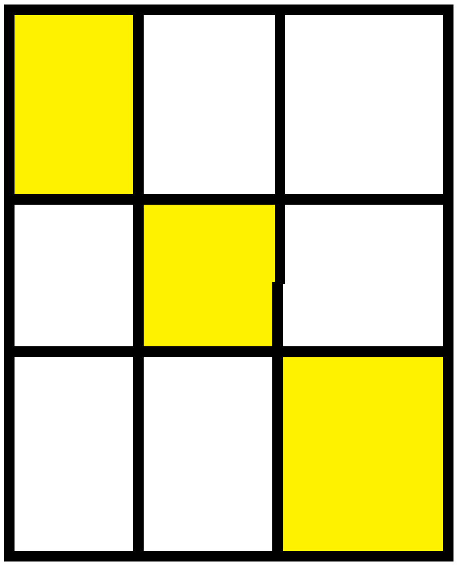
```
package array;

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

        //2D static jagged array : set of 1d array
        int arr[][]= {

                {11,22,33,44,54,56},  //dep 0
                {33,44,55,66},  //dep 1
                {55,66,77}   //dep 2
        };
        System.out.println(arr.length);
        System.out.println(arr[2].length);

    }
}
```

Print using for loop

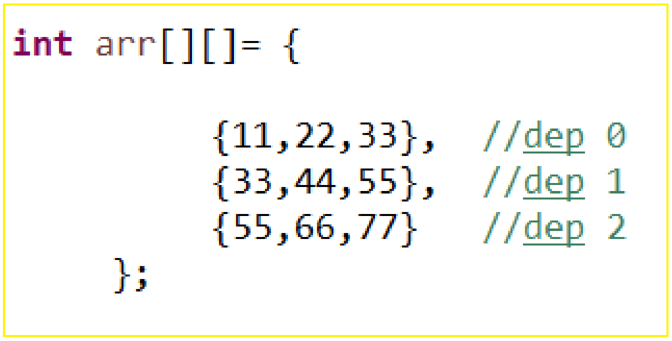
```
package array;

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

        //2D static jagged array : set of 1d array
        int arr[][]= {

                {11,22,33,44,54,56},  //dep 0
                {33,44,55,66},  //dep 1
                {55,66,77}   //dep 2
        };

        System.out.println(arr.length);
        System.out.println(arr[2].length);

        //print using for loop
        for(int i=0;i<arr.length;i++)
        {
            for(int j=0; j<arr[i].length ; j++)
            {
                System.out.print(arr[i][j]+" ");
            }
            System.out.println();
        }

    }
}
```

TASK

WAP to sum up all the elements lies on diagonals



3x3



```
int arr[][]= {

            {11,22,33},   //dep 0
            {33,44,55},   //dep 1
            {55,66,77}    //dep 2
      };
```

□

2D dynamic array

- We can create dynamic array by using new keyword

```
package array;
import java.util.Scanner;

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

          Scanner s = new Scanner(System.in);
          int arr[][]=new int[2][3];  //2 dep //3 std


          for(int i=0;i<arr.length;i++)
          {
               for(int j=0; j<arr[i].length ; j++)
               {
          System.out.print("enter a["+i+"]["+j+"]=");
                    arr[i][j]=s.nextInt();
               }

          }

          //print using for loop
          for(int i=0;i<arr.length;i++)
          {
               for(int j=0; j<arr[i].length ; j++)
               {
                    System.out.print(arr[i][j]+"  ");
               }
               System.out.println();
          }
     }
}
```

2D dynamic jagged array

```java
package array;
import java.util.Scanner;
public class Launch {
    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);
        int arr[][]=new int[2][];    //2 dep
        arr[0]=new int[5];//0
        arr[1]=new int[2];//1


         for(int i=0;i<arr.length;i++)
         {
             for(int j=0; j<arr[i].length ; j++)
             {
             System.out.print("enter a["+i+"]["+j+"]=");
                 arr[i][j]=s.nextInt();
             }
         }

         //print using for loop
         for(int i=0;i<arr.length;i++)
         {
             for(int j=0; j<arr[i].length ; j++)
             {
                 System.out.print(arr[i][j]+" ");
             }
             System.out.println();
         }
    }
}
```

3D static array

Group of 2d array

```java
package array;

import java.util.Scanner;

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

        int arr[][][]= {

                {
                    {23,45,67,11},
                    {67,54,34,56},
                    {98,65,17,54}
                },
                {
                    {73,45,64,18},
                    {67,44,34,56},
                    {38,61,17,58}
                }
        };

        System.out.println(arr[0][2][2]);
    }
}
```

3D static jagged array

```java
package array;
import java.util.Scanner;

public class Launch {
	public static void main(String[] args) {
		
		int arr[][][]= {
				
				{
					{23,45,67,11,54},
					{67,54,},
					{98,65,17,54}
				},
				{
					{73,45,64},
					{67,44,34,56,64,42},
					{38,61}
				}
			
		};
		
		for(int i=0;i<arr.length;i++)//3d
		{
			for(int j=0;j<arr[i].length;j++)//2d
			{
				for(int k=0;k<arr[i][j].length;k++)
				{
					System.out.print(arr[i][j][k]+" ");
				}
				System.out.println();
			}
			System.out.println();System.out.println();
		}
		
	}
}
```

3D dynamic array

```java
package array;
import java.util.Scanner;
public class Launch {
    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);
        int arr[][][]=new int[2][3][4];//2clg //3dep //4std

        for(int i=0;i<arr.length;i++)//3d
        {
            for(int j=0;j<arr[i].length;j++)//2d
            {
                for(int k=0;k<arr[i][j].length;k++)
                {
    System.out.print("Enter arr["+i+"]["+j+"]["+k+"]=");
                    arr[i][j][k]=s.nextInt();
                }
                System.out.println();
            }
            System.out.println();System.out.println();
        }
        for(int i=0;i<arr.length;i++)//3d
        {
            for(int j=0;j<arr[i].length;j++)//2d
            {
                for(int k=0;k<arr[i][j].length;k++)
                {
                    System.out.print(arr[i][j][k]+"  ");
                }
                System.out.println();
            }
            System.out.println();System.out.println();
        }
    }
}
```

3D dynamic jagged array

```
package array;
import java.util.Scanner;
public class Launch {
     public static void main(String[] args) {

          Scanner s = new Scanner(System.in);
          System.out.println("enter clg");
          int clg=s.nextInt();
          int arr[][][]=new int[clg][][];//2clg

          arr[0]=new int[3][]; //0th clg  3dep
          arr[1]=new int[2][]; //1st clg  2dep

          arr[0][0]=new int[5];//0th clg 0th dep
          arr[0][1]=new int[4];//0th clg 1st dep
          arr[0][2]=new int[3];//0th clg 2nd dep

          arr[1][0]=new int[4];//1st clg 0th dep
          arr[1][1]=new int[3];//1st clg 1st dep


          for(int i=0;i<arr.length;i++)//3d
          {
               for(int j=0;j<arr[i].length;j++)//2d
               {
                    for(int k=0;k<arr[i][j].length;k++)
                    {
         System.out.print("Enter arr["+i+"]["+j+"]["+k+"]=");
                         arr[i][j][k]=s.nextInt();
                    }
                    System.out.println();
               }
               System.out.println();System.out.println();
          }
          for(int i=0;i<arr.length;i++)//3d
          {
```

```
				for(int j=0;j<arr[i].length;j++)//2d
				{
					for(int k=0;k<arr[i][j].length;k++)
					{
						System.out.print(arr[i][j][k]+" ");
					}
					System.out.println();
				}
				System.out.println();System.out.println();
			}
		}
}
```

3D pure dynamic jagged array

```
package array;
import java.util.Scanner;
public class Launch {
	public static void main(String[] args) {

		Scanner s = new Scanner(System.in);
		System.out.println("enter clg");
		int clg=s.nextInt();

//		int arr[][][];//declaration
//		arr=new int[clg][][];//initialization

		int arr[][][]=new int[clg][][];

		for(int i=0;i<clg;i++)
		{
			System.out.println("Enter dep in clg "+i);
			int dep=s.nextInt();
			arr[i]=new int[dep][];
		}

		for(int i=0;i<clg;i++)
		{
```

```
                for(int j=0;j<arr[i].length;j++)
                {
        System.out.println("Enter std in clg "+i+" dep "+j);
                    int std=s.nextInt();
                    arr[i][j]=new int[std];
                }
            }

            for(int i=0;i<arr.length;i++)//3d
            {
                for(int j=0;j<arr[i].length;j++)//2d
                {
                    for(int k=0;k<arr[i][j].length;k++)
                    {
          System.out.print("Enter arr["+i+"]["+j+"]["+k+"]=");
                        arr[i][j][k]=s.nextInt();
                    }
                    System.out.println();
                }
                System.out.println();System.out.println();
            }
            for(int i=0;i<arr.length;i++)//3d
            {
                for(int j=0;j<arr[i].length;j++)//2d
                {
                    for(int k=0;k<arr[i][j].length;k++)
                    {
                        System.out.print(arr[i][j][k]+" ");
                    }
                    System.out.println();
                }
                System.out.println();System.out.println();
            }
        }
    }
```

How to pass array to a method

```
package array;
public class Launch {
      public static void main(String[] args) {

      int arr[]= {11,22,33,44,55};

      Demo d = new Demo();
      int newarr[]=d.abc(arr);
      for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i]+" ");
      }
   }
}
```

```
package array;

public class Demo {

      public int[] abc(int arr[])
      {
            for (int i = 0; i < arr.length; i++) {
                  System.out.print(arr[i]+" ");
            }

            //add 5 with each element
            for (int i = 0; i < arr.length; i++) {
                  arr[i]=arr[i]+5;
            }
            System.out.println();
            return arr;

      }

}
```

Anonymous Array

```
package array;

public class Demo {

	public int[] abc(int arr[])
	{
		return new int[5];
		//array without name
	}
}
```

String in Java

- String->Character enclose within double quotes

String is of two types

1.mutable string (We can change it) ex. address,phone..etc

- StringBuffer : class (java.lang)
- StringBuilder : class (java.lang)

2.immutable string(We cant change it) ex. dob , AdhaarNo..etc

- With new keyword
- Without new keyword

Immutable String

```
package array;
public class Launch {
     public static void main(String[] args) {

    //Immutable

    //Without new keyword
     String s1=""; //empty string
     String s2=" "; //not empty string
     String s3="abcd...upto ram size";
     System.out.println(s3);
     //String is not primitive datatype like int float char
     //String is predefined class in java found in java.lang
package

  }
}
```

String method

1.concat()

```
package array;
public class Launch {
     public static void main(String[] args) {
```

```
		//immutable

		String s="hello";
		//String concatenation

		String s2 = s.concat("abcd");
		System.out.println(s2);

		System.out.println(s);


	}
}
```

Note: we cannot change immutable string but we can completely override

```
package array;
public class Launch {
	public static void main(String[] args) {

		//immutable

		String s="hello";
		//String concatenation

		s = s.concat("abcd");

		System.out.println(s);


	}
}
```

2.length()

```
package array;
public class Launch {
    public static void main(String[] args) {

     //immutable
     String s="hello";  //index : 0 to 4
     int len=s.length();
     System.out.println(len);
     System.out.println(s.length());

  }
}
```

3.charAt()

```
package array;
public class Launch {
    public static void main(String[] args) {

     //immutable
     String s="hello";  //index : 0 to 4
     char c = s.charAt(1);
     System.out.println(c);

  }
}
```

4.equals()

```
package array;
public class Launch {
    public static void main(String[] args) {
         //immutable
        String s="hello";  //index : 0 to 4
        String s1="hello";

        boolean equals = s.equals(s1);
        System.out.println(s==s1);

    }
}
```

Task :WAP to check whether a string is palindrome or not

Ex. nitin , dad , malayalam

String s=”dad”;

Output : this is palindrome

String s=”abcd”;

Output : this is not palindrome

```
package array;
public class Launch {
     public static void main(String[] args) {

         //immutable
        String s="dad";  //index : 0 to 4
        //reverse : olleh
        //if sidha is eq to ulta : palindrome
        String k="";
        for(int i=s.length()-1;i>=0;i--)
        {
         k=k+s.charAt(i);
        }

        System.out.println(k);

        if(s.equals(k))
        {
         System.out.println("palindrome");
        }
        else
        {
         System.out.println("not palindrome");
        }

    }
}
```

5.getBytes() : it will return only ascii value of each character

```
package array;
public class Launch {
       public static void main(String[] args) {

        String s="ABCD";
        byte[] b = s.getBytes();
        for (int i = 0; i < b.length; i++) {

         System.out.println((char)b[i]+" : "+b[i]);
         }

    }
}
```

Task :

WAP to filter only character(UL)

String s=”1a2 b%c* dAB3 C4D”;

OUTPUT: abcdABCD

Hint: getBytes()

```
package array;
public class Launch {
     public static void main(String[] args) {
      String s="1a2 b%c* dAB3 C4D";
      byte[] b = s.getBytes();
      for (int i = 0; i < b.length; i++) {

       if((b[i]>=65 && b[i]<=90) || (b[i]>=91 && b[i]<=122))
       {
            System.out.print((char)b[i]);
       }

       }
    }
}
```

Task :

String s=”abcyz”; //97 98 99 121 122

int k=5;

OUTPUT: fghde

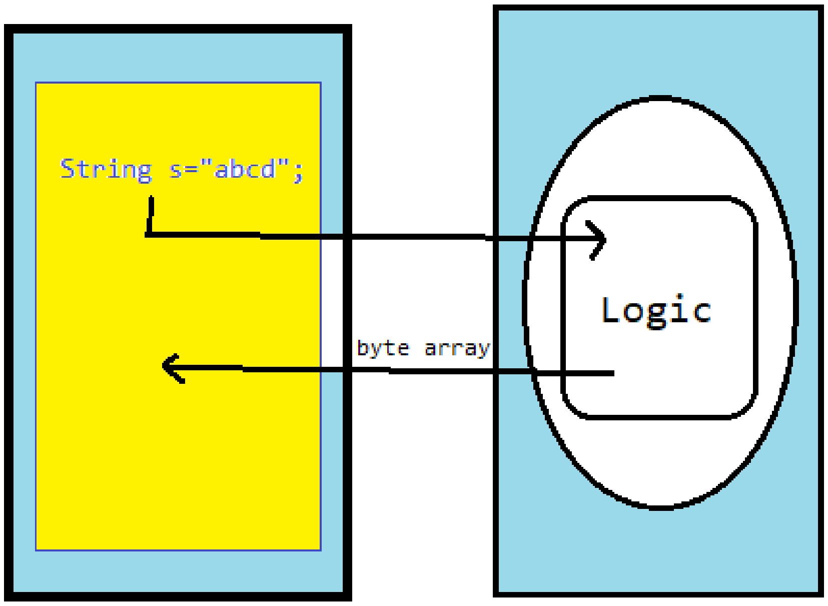
int k=2;

OUTPUT: cdeab

```
package array;
public class Launch {
    public static void main(String[] args) {


      String s="abcdzyABCZ";  //z: 122+3=125
      int shift=3;
      byte[] b = s.getBytes();
      for (int i = 0; i < b.length; i++) {

       //97 122
       int totalValue=b[i]+shift;  //122+3=125
       int forwardshift=totalValue-122;  //3

       if(totalValue>122)
       {
            System.out.print((char)(forwardshift+97-1));
       }
       else
       {
          System.out.print((char)totalValue);
       }
       //

       }
   }
}
```

Task

- Wap to convert all character of string into its ascii value in the form of byte array
- getBytes() is not allowed



```
hint: type casting
char c=(char)65
byte b=(byte)A
```

6.toCharArray()

```
package array;
public class Launch {
	public static void main(String[] args) {


	String s="ABCD";
	byte[] b = new Demo().abc(s);
	for (int i = 0; i < b.length; i++) {
	  System.out.println((char)b[i]+" : "+b[i]);
	  }
  }
}
```

```
package array;

public class Demo {

	public byte[] abc(String s)
	{
		byte b[]=new byte[s.length()];

		char[] c = s.toCharArray();
		//string to char array

		for (int i = 0; i < c.length; i++)
		{
			b[i]=(byte)c[i];
		}

		return b;
	}
}
```

7.toUpperCase()8.toLowerCase()

```
package array;
public class Launch {
	public static void main(String[] args) {


	 String s="ABCDabcd";

	 String s2 = s.toUpperCase();
	 System.out.println(s2);

	 String s3 = s.toLowerCase();
	 System.out.println(s3);
  }
}
```

9.startsWith() 10.endsWith()

```java
package array;
public class Launch {
    public static void main(String[] args) {

     String s="abcd@gmail.com";

     if(s.endsWith("@gmail.com"))
     {
      System.out.println("VALID");
     }
     else
     {
      System.out.println("INVALID");
     }
  }
}
```

```java
package array;
public class Launch {
    public static void main(String[] args) {

     String s="abcd@gmail.com";

     if(s.startsWith("abcd"))
     {
      System.out.println("VALID");
     }
     else
     {
      System.out.println("INVALID");
     }
  }
}
```

11.substring

```
package array;
public class Launch {
    public static void main(String[] args) {

     String s="abcd@gmail.com";
     String s1 = s.substring(3); //3 to last
     System.out.println(s1);

     String s2 = s.substring(0, 4); //0 to 3
     System.out.println(s2);
  }
}
```

12.split

```
String s="this is my car";
String[] split = s.split(" ");
System.out.println(split[1]);


Output : is
```

Task

Input string:- this is my car

Output string:- This Is My Car

Hint: substring

```
package array;
public class Launch {
    public static void main(String[] args) {

     //Input string:- this is my car
      //Output string:- This Is My Car

     String s="this is my car";
     String k="";
     String[] split = s.split(" ");

     for (int i = 0; i < split.length; i++) {

      String first = split[i].substring(0, 1).toUpperCase();
      String remaining=split[i].substring(1); //this-->his
      k=k+first+remaining+" ";

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

14.replace

```
package array;
public class Launch {
    public static void main(String[] args) {

      //Input string:- this is my car
      //Output string:- This Is My Car

     String s="this is my car car car";
     String replace = s.replace("car", "bike");
     System.out.println(replace);

   }
}
```

Task:

User input: this is my car

User input: 3

Output : this is my rac

```
package array;
public class Launch {
     public static void main(String[] args) {

      //Input string:- this is my car
     //Output string:- This Is My Car

      String s="this is my car";
      String rev="";
      int k=0;
      String[] split = s.split(" ");

      String s1 = split[k]; //car

      for(int i=s1.length()-1 ; i>=0 ; i--)
      {
       rev=rev+s1.charAt(i);
      }

      split[k]=rev;

      for (int i = 0; i < split.length; i++) {
            System.out.print(split[i]+" ");
     }
   }
}
```

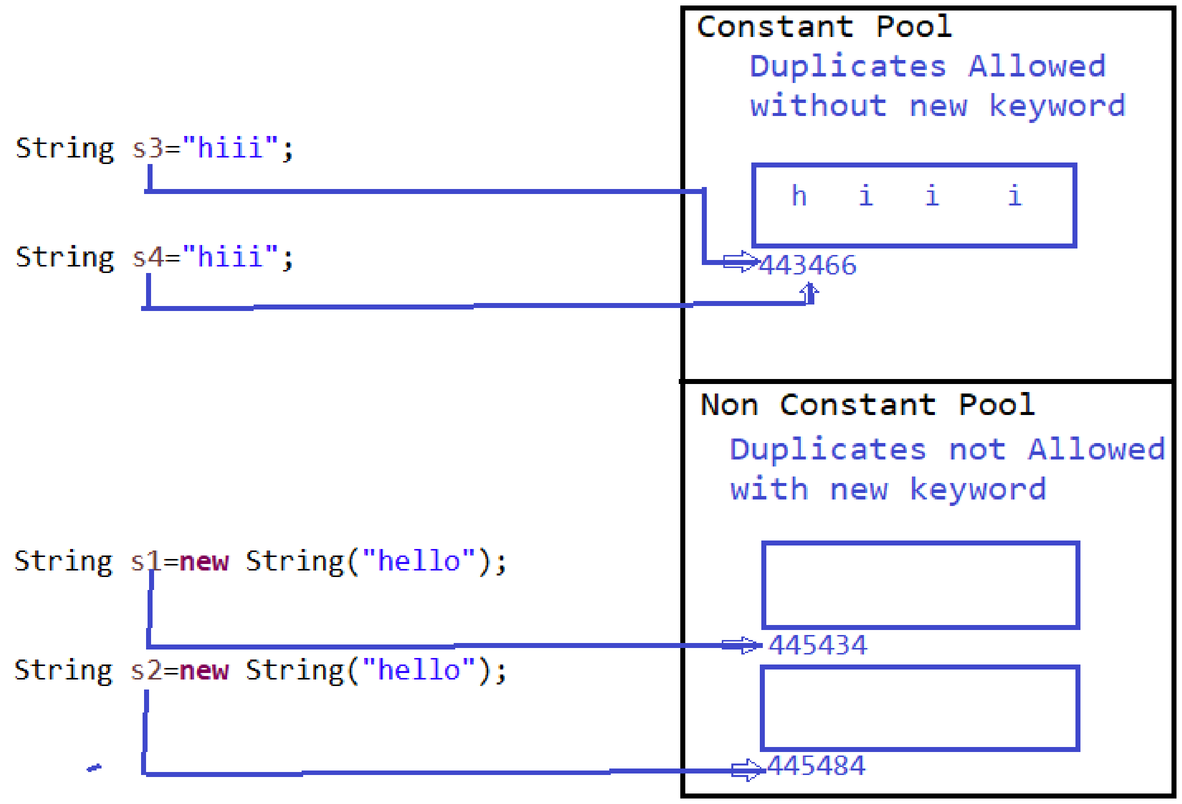
15.indexOf()

```
package array;
public class Launch {
     public static void main(String[] args) {

      String s="thist";
      int i = s.indexOf("th");
      System.out.println(i);

   }
}
```

Immutable String with new keyword

```
package array;
public class Launch {
     public static void main(String[] args) {

    //String by using new keyword

          String s1=new String("hello");
          String s2=new String("hello");
          System.out.println(s1==s2);     //false

          String s3="hiii";
          String s4="hiii";
          System.out.println(s3==s4);     //true

   }
}
```

String Constant and Non Constant Pool



Mutable String

1.StringBuffer : class : java.lang

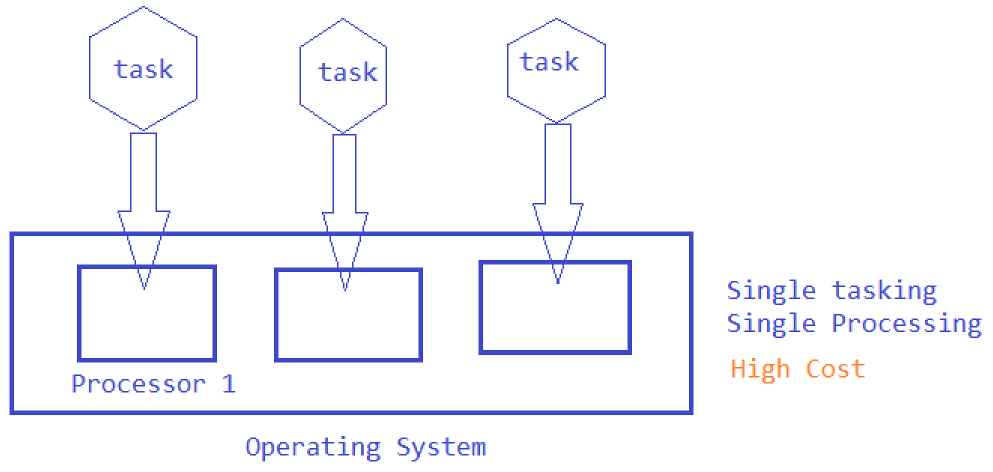
2.StringBuilder : class : java.lang

```
package array;
public class Launch {
	public static void main(String[] args) {

		String s1="hiii";
		s1.concat("abc");
		System.out.println(s1);
```

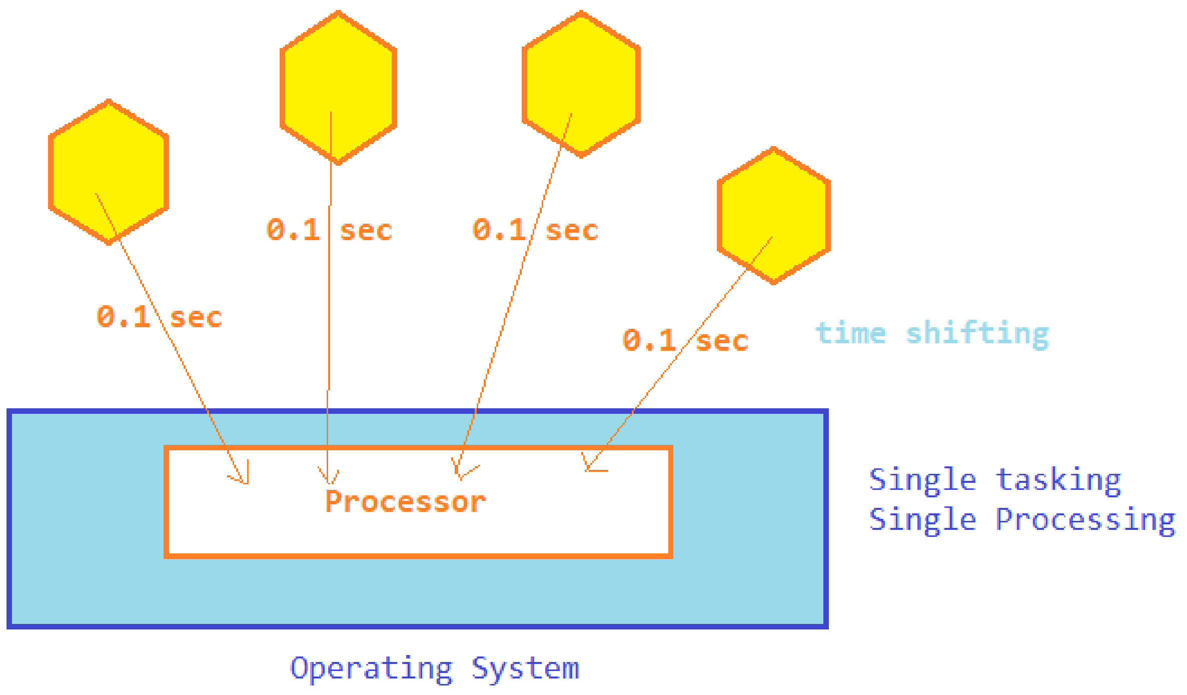
```
        //mutable

//      StringBuilder s2=new StringBuilder("hiii");
//      System.out.println(s2);
//      s2.append("abc");
//      System.out.println(s2);
//      s2.reverse();
//      System.out.println(s2);

        StringBuffer s2=new StringBuffer("hiii");
        System.out.println(s2);
        s2.append("abc");
        System.out.println(s2);
        s2.reverse();
        System.out.println(s2);

        //StringBuilder : non synchronized
        //StringBuffer  : synchronized
    }
}
```

What is synchronization?

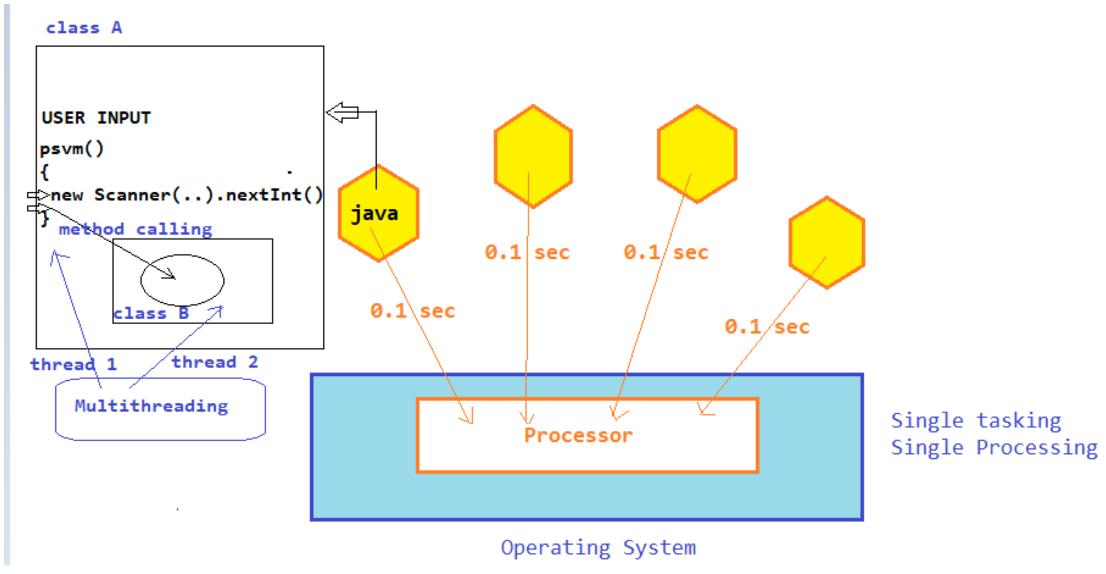
OLD MODEL



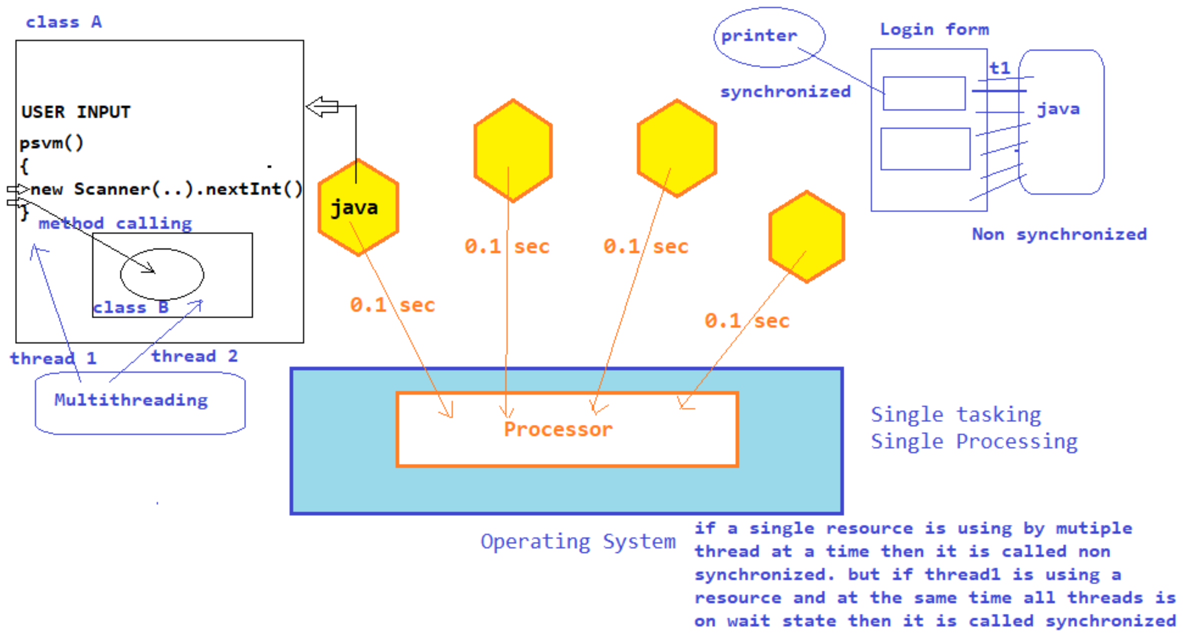
NEW MODEL



JAMES GOSLIN



Synchronization



```
psvm()
{
new Scanner(..).nextInt()
}
```

if a single resource is using by mutiple thread at a time then it is called non synchronized. but if thread1 is using a resource and at the same time all threads is on wait state then it is called synchronized

OOPS

Object Oriented Programming Concepts

->All the concepts of OOP is taken from Real-World

- class
- object
- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

What is Class

->class is a keyword in Java which is used to create class in Java

Ex.

```
class Demo
{
}
```

->Realworld: class is define a group of similar entity

->Main purpose of class is to achieve categorization

AMAZON

Product->add,delete,update,read

Login->Login Logout code

Payment->GPAY,UPI,CARDS..etc

Main purpose: categorization

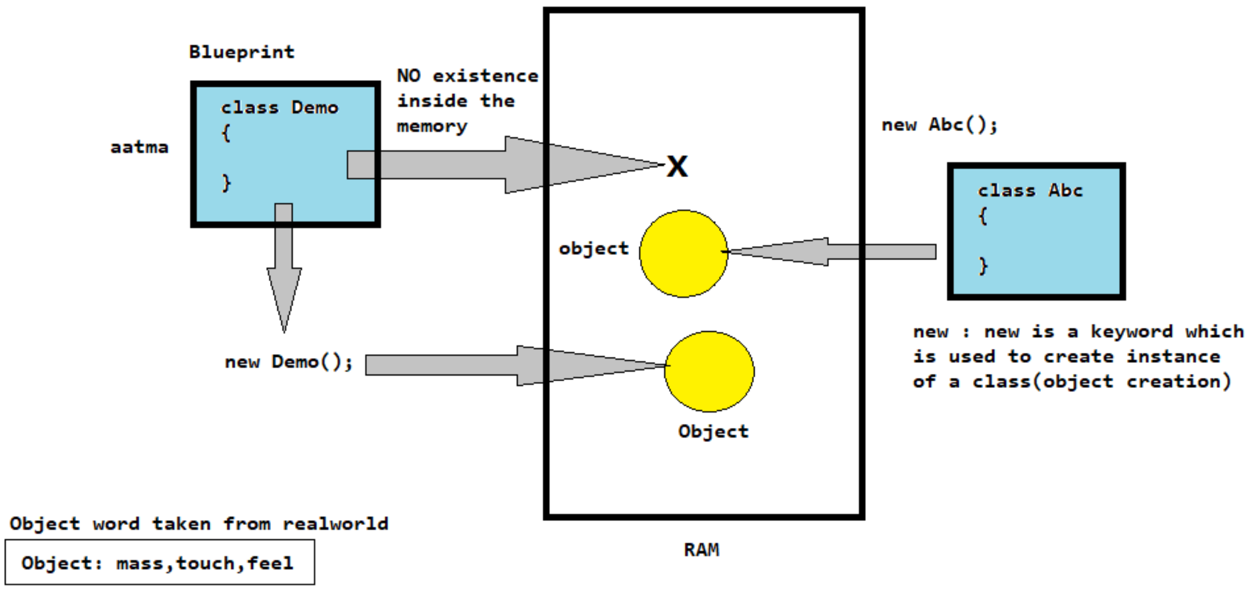
Scanner: input

String: text manipulation

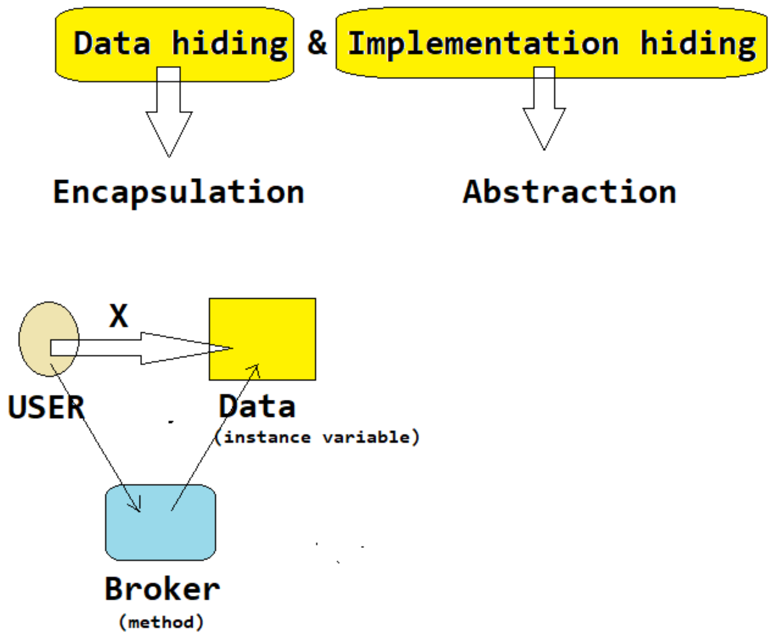
Package: used to categorize class

```
Library{
   Package->folder
   {
     class Add
     {
         //logic
     }
    class Sub
    {
    }
    class Div
    {
    }
    class Mul
    {
    }
  }
}
```

- Class has no physical existence, it is just a blueprint of your problem



ENCAPSULATION



Access Modifier :

variable(instance var),methods

- Public : scope: anywhere(8X)
- Private : scope: within the class(red dot)
- Default
- Protected

Encapsulation

- ➢ Stop Direct access
- ➢ Allow indirect access

```java
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{

		Employee emp = new Employee();
		emp.abc();//get

		//change password
		emp.abc1();//get

		emp.abc();//get
	}
}


package com.mainapp;
import java.util.Scanner;

public class Employee {
	private int password=1111;

	public void abc()
	{
		System.out.println("Enter pass");
		int key = new Scanner(System.in).nextInt();
		if(key==1234)
		System.out.println(this.password);
		else
		System.out.println("null");
	}

	//change password method

	public void abc1()
	{
		System.out.println("enter the new pasword");
		Scanner s = new Scanner(System.in);
		this.password= s.nextInt();
		System.out.println("password is changed");

	}

}
```

Output:

```
Enter pass
1234
1111
enter the new pasword
4444
password is changed
Enter pass
1234
4444
```

Setter and Getter

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{

		Employee emp = new Employee();

		//setter
		emp.setEid(19);
		emp.setEname("raju");
		emp.setEaddress("csk");
		emp.setEsalary(1000);

		//getter
		System.out.println(emp.getEid());
		System.out.println(emp.getEname());
		System.out.println(emp.getEaddress());
		System.out.println(emp.getEsalary());

		emp.setEid(190);
		emp.setEname("kaju");
		emp.setEaddress("mi");
		emp.setEsalary(2000);

		//getter
		System.out.println(emp.getEid());
		System.out.println(emp.getEname());
		System.out.println(emp.getEaddress());
		System.out.println(emp.getEsalary());


	}
}

package com.mainapp;
public class Employee {

	private int eid;//0
	private String ename;
	private String eaddress;
	private int esalary;
	public int getEid() {
		return eid;
	}

	public void setEid(int eid) {
		this.eid = eid;
	}

	public String getEname() {
		return ename;
	}

	public void setEname(String ename) {
		this.ename = ename;
	}

	public String getEaddress() {
		return eaddress;
	}

	public void setEaddress(String eaddress) {
		this.eaddress = eaddress;
	}

	public int getEsalary() {
		return esalary;
	}

	public void setEsalary(int esalary) {
		this.esalary = esalary;
	}
	//alt shift s
}
```

Task

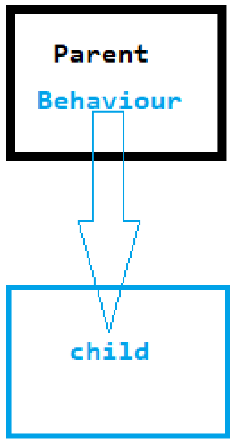
- Store 10 emp data by using encapsulation class
- Setter and getter
- Multiple Employee objects
- Array

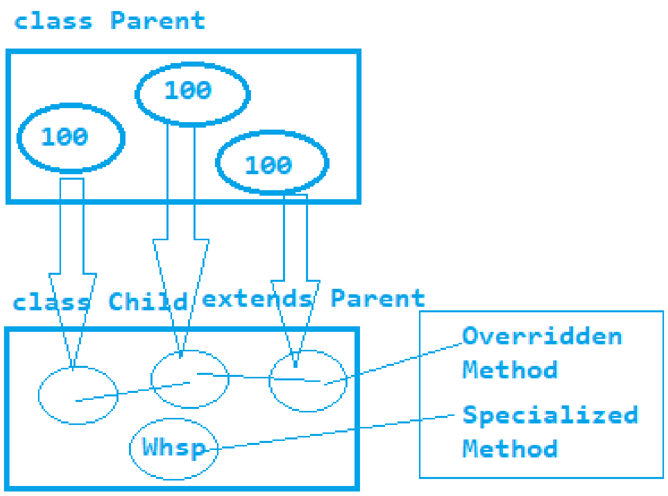
```
package com.mainapp;

import java.util.Scanner;
public class Launch {

	public static void main(String[] args)
	{

		Scanner s = new Scanner(System.in);
		System.out.println("Enter the count of emp");
		int count=s.nextInt();//5

		Employee e[]=new Employee[count];
		int i=0;
		while(i<count)
		{
			Employee emp = new Employee();

			System.out.println("Enter eid");
			int eid=s.nextInt();
			emp.setEid(eid);

			System.out.println("Enter ename");
			emp.setEname(s.next());

			System.out.println("Enter eaddress");
			emp.setEaddress(s.next());

			System.out.println("Enter esalary");
			emp.setEsalary(s.nextInt());
			e[i]=emp;
			i++;
		}

		//print
//		Employee e1=e[0];
//		System.out.println(e1.getEid());
//		System.out.println(e1.getEname());
//		System.out.println(e1.getEaddress());
//		System.out.println(e1.getEsalary());
//
//		Employee e2=e[1];
//		System.out.println(e2.getEid());
//		System.out.println(e2.getEname());
//		System.out.println(e2.getEaddress());
//		System.out.println(e2.getEsalary());
//		for (int j = 0; j < e.length; j++) {
//			Employee emp=e[j];
//			System.out.println(emp.getAlldata());
//		}
		
		//foreach loop
		for(Employee emp:e)
		{
			System.out.println(emp.getAlldata());
		}
	}
}


package com.mainapp;

public class Employee {

	private int eid;
	private String ename;
	private String eaddress;
	private int esalary;
	
	//alt shift s: generate setter & getter
	public int getEid() {
		return eid;
	}

	public void setEid(int eid) {
		this.eid = eid;
	}

	public String getEname() {
		return ename;
	}

	public void setEname(String ename) {
		this.ename = ename;
	}

	public String getEaddress() {
		return eaddress;
	}
	public void setEaddress(String eaddress) {
		this.eaddress = eaddress;
	}

	public int getEsalary() {
		return esalary;
	}

	public void setEsalary(int esalary) {
		this.esalary = esalary;
	}

	public String getAlldata()
	{
		return "{ ID: "+this.eid+" Name:"
			+ " "+this.ename+" Address: "+this.eaddress+" Salary: 
"+this.esalary+" }";
	}

}
```

Output:

```
Enter the count of emp
2
Enter eid
12
Enter ename
raju
Enter eaddress
csk
Enter esalary
1000
Enter eid
13
Enter ename
kaju
Enter eaddress
mi
Enter esalary
2000
{ ID: 12 Name: raju Address: csk Salary: 1000 }
{ ID: 13 Name: kaju Address: mi Salary: 2000 }
```

Inheritance





➡ extends: It is a keyword which is used to inherit the class

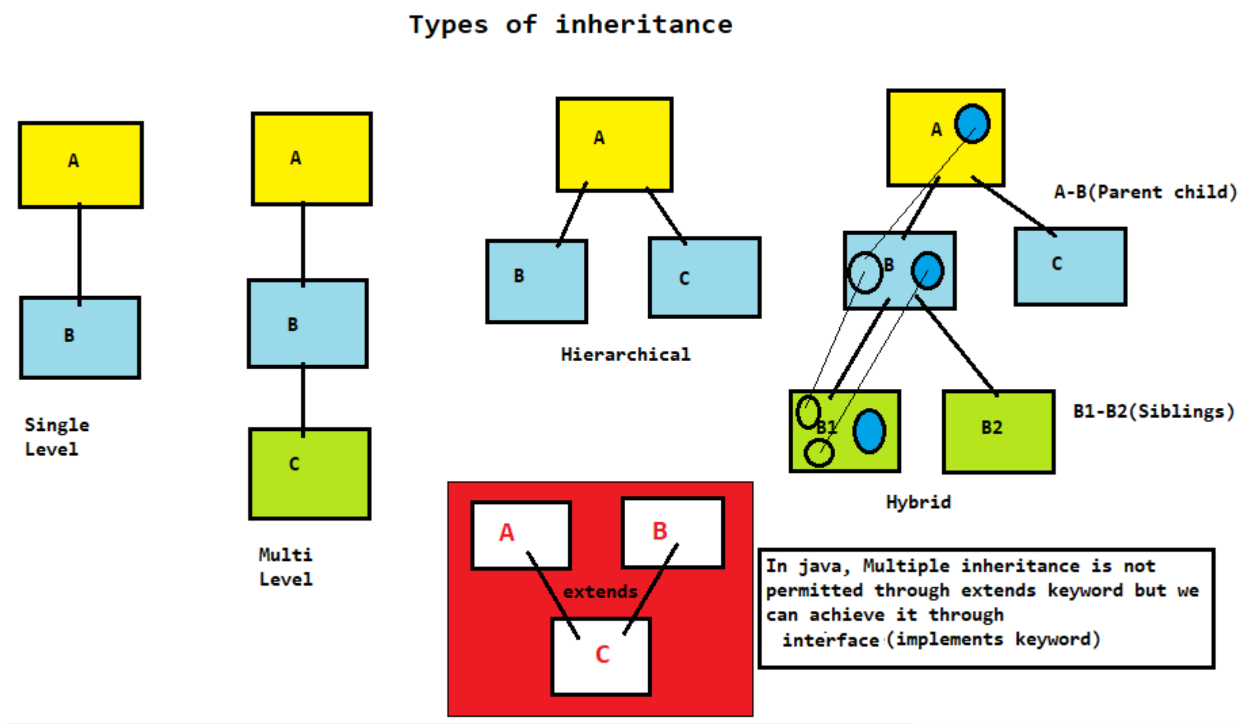
➡ Advantage:
reduce line of code
profitable for org

Program

```
package com.mainapp;
public class Launch {

	public static void main(String[] args) {
	Demo1 d1 = new Demo1();
	d1.m1();
	d1.m2();
	d1.m3();

	System.out.println();
	Demo2 d2 = new Demo2();
	d2.m1();
```

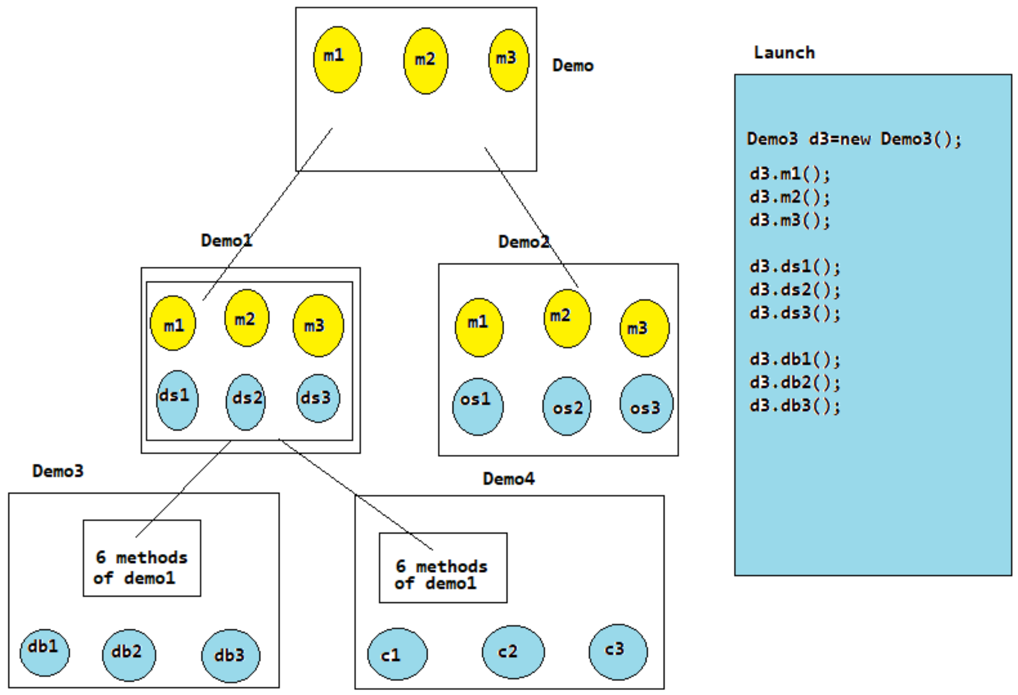
```
        d2.m2();
        d2.m3();
        d2.m4();

        }
}
package com.mainapp;
public class Demo1 {

        public void m1()
        {
                System.out.println("m1");
        }
        public void m2()
        {
                System.out.println("m2");
        }
        public void m3()
        {
                System.out.println("m3");
        }
}
package com.mainapp;
public class Demo2 extends Demo1 {
    @Override //Annoation
        public void m1()
        {
                System.out.println("my m1");
        }
    @Override
        public void m2()
        {
                System.out.println("my m2");
        }
    @Override
        public void m3()
        {
                System.out.println("my m3");
        }
    //specialized
    public void m4()
        {
                System.out.println("my m4");
        }
}
```

TYPES OF INHERITANCE



In java, Multiple inheritance is not permitted through extends keyword but we can achieve it through interface (implements keyword)

Hybrid example:



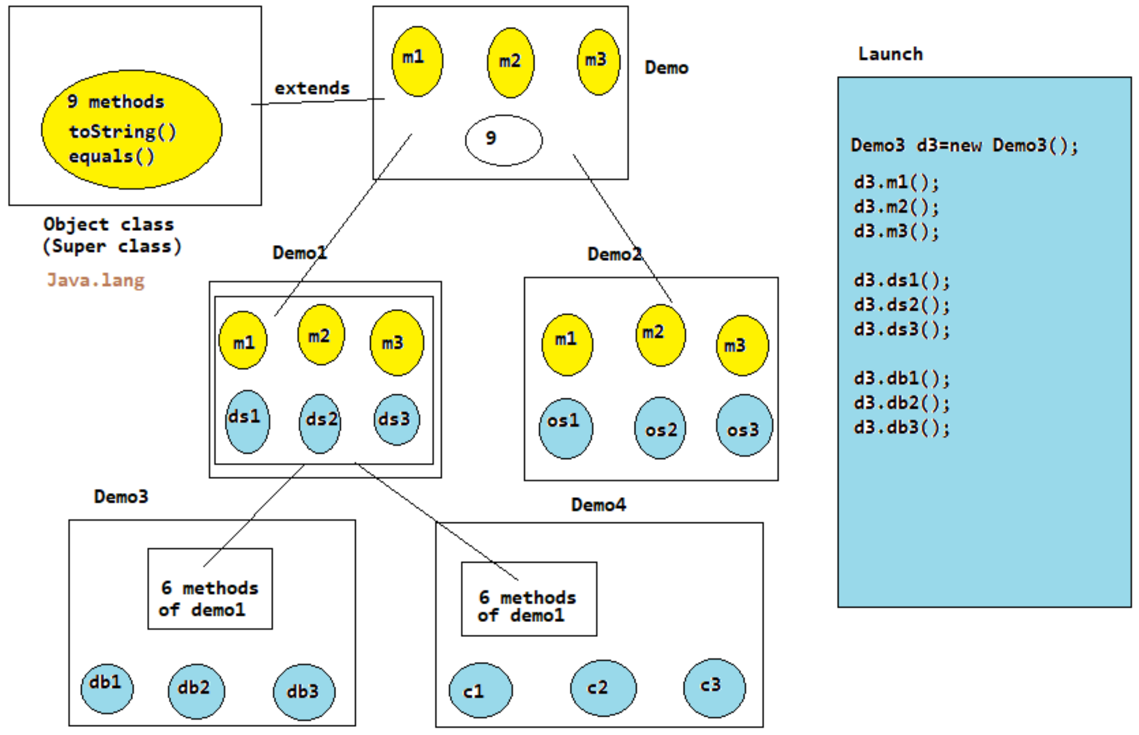
```
Demo3 d3=new Demo3();
d3.m1();
d3.m2();
d3.m3();

d3.ds1();
d3.ds2();
d3.ds3();

d3.db1();
d3.db2();
d3.db3();
```

Program

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{

	Demo3 d3 = new Demo3();
	d3.m1();
	d3.m2();
	d3.m3();

	d3.ds1();
	d3.ds2();
	d3.ds3();

	d3.db1();
	d3.db2();
	d3.db3();
  }
}

package com.mainapp;
public class Demo {

	public void m1()
	{
		System.out.println("m1");
	}
	public void m2()
	{
		System.out.println("m2");
	}
	public void m3()
	{
		System.out.println("m3");
	}
}

package com.mainapp;
public class Demo1 extends Demo {

	public void ds1()
	{
		System.out.println("ds1");
	}
	public void ds2()
	{
		System.out.println("ds2");
	}
	public void ds3()
	{
		System.out.println("ds3");
	}

}

package com.mainapp;
public class Demo2 extends Demo {

	public void os1() {
		System.out.println("os1");
	}

	public void os2() {
		System.out.println("os2");
	}

	public void os3() {
		System.out.println("os3");
	}

}

package com.mainapp;
public class Demo3 extends Demo1 {

	public void db1()
	{
		System.out.println("db1");
	}
	public void db2()
	{
		System.out.println("db2");
	}
	public void db3()
	{
		System.out.println("db3");
	}
```

```
}

package com.mainapp;
public class Demo4 extends Demo1 {

	public void c1()
	{
		System.out.println("c1");
	}
	public void c2()
	{
		System.out.println("c2");
	}
	public void c3()
	{
		System.out.println("c3");
	}
}
```

- Every class extends Object class internally except child class



```
Demo3 d3=new Demo3();
d3.m1();
d3.m2();
d3.m3();

d3.ds1();
d3.ds2();
d3.ds3();

d3.db1();
d3.db2();
d3.db3();
```

toString()

its internal implementation

```
public String toString()
{
  return getClass().getName() + "@" + Integer.toHexString(hashCode());
}


package com.mainapp;
public class Launch {

      public static void main(String[] args)
      {
            Demo d = new Demo();
            System.out.println(d);
      }
}

package com.mainapp;
public class Demo extends Object {

      public void m1()
      {
            System.out.println("m1");
      }
      public void m2()
      {
            System.out.println("m2");
      }
      public void m3()
      {
            System.out.println("m3");
      }
}
```

```
Output: com.mainapp.Demo@5a07e868
```

- We can override toString() method inside Demo class because it is inside Object class which is parent class of Demo class

After overriding:

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{

		Demo d = new Demo();
		System.out.println(d);

	}
}
package com.mainapp;
public class Demo extends Object {  //extends object is optional

	public void m1()
	{
		System.out.println("m1");
	}
	public void m2()
	{
		System.out.println("m2");
	}
	public void m3()
	{
		System.out.println("m3");
	}
	@Override
	public String toString() {

		return "raju";
	}
}
```

Output: raju

Constructor chaining

*Constructor is a special type of method
*It has No return type
*Constructor Name should be same as class Name
*Constructor should be public
*After object creation JVM calls constructor automatically
We can have multiple constructor in a same class

- this() : current class constructor
- this(): it should be first statement inside the constructor(we can use it only inside Constructor)

```
package com.mainapp;
public class Launch {

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

//this->instance var
//constructor calling-> this()


package com.mainapp;

public class Demo {
	public Demo()
	{
		this(10);
		System.out.println("ZPC");
	}
	public Demo(int a)
	{
		this("aabc",10);
		System.out.println("OPC");
	}
	public Demo(String s,int a)
	{
		System.out.println("TPC");
	}
}
```

- Recursion is not permitted in Java
- Constructor is used to initialize object(instance variable)

Constructor->initialize data

setter()-->update data

- If a class doesn't have any user define constructor then JVM will create one No-Arg(ZPC) constructor

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{
	   Demo d = new Demo(10,"raju","csk",1000);
	   System.out.println(d.toString());
```

```
        //update
        d.setEaddress("mi");
        d.setEsalary(2000);
        System.out.println(d.toString());

    }
}

package com.mainapp;
public class Demo {

    private int eid;
    private String ename;
    private String eaddress;
    private int esalary;
    //initialization
    public Demo(int eid, String ename, String eaddress, int esalary)
    {

        this.eid = eid;
        this.ename = ename;
        this.eaddress = eaddress;
        this.esalary = esalary;
    }
    public int getEid() {
        return eid;
    }
    public void setEid(int eid) {
        this.eid = eid;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public String getEaddress() {
        return eaddress;
    }
    public void setEaddress(String eaddress) {
        this.eaddress = eaddress;
    }
    public int getEsalary() {
        return esalary;
    }
    public void setEsalary(int esalary) {
```

```
		this.esalary = esalary;
	}

	@Override
	public String toString() {
		return "Demo [eid=" + eid + ", ename=" + ename + ", 
eaddress=" + eaddress + ", esalary=" + esalary + "]";
	}
}
```

- super() constructor call
- super(): it will Call Zero Parametrized Constructor from Parent class
- this(): it will Call Zero Parametrized Constructor from the same class

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{
	   new Demo1();
	}
}
```

```
package com.mainapp;

public class Demo {

    public Demo() {

		System.out.println("PC-ZPC");
	}
    public Demo(int a) {
		this();
		System.out.println("PC-OPC");
	}

}
```

```
package com.mainapp;

public class Demo1 extends Demo {

	public Demo1() {

		super(10);//internally
		System.out.println("CC-ZPC");
	}


}
```

Tight Coupling

```
package com.mainapp;
public class Demo {

	public Demo(int a) {

		System.out.println("PC-OPC");
	}

}
package com.mainapp;

public class Demo1 extends Demo {

	public Demo1() {
		super();   //ERROR
		System.out.println("CC-ZPC");
	}
}
```

We didn't change anything inside Demo1 still it is showing error coz of tight coupling between Demon and Demo1

Solution: Loose Coupling : **Through Interface**

this this() & super super()

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{
	   new Demo1();
	}
}
```

```
package com.mainapp;

public class Demo1 extends Demo {
	public int a=100;

	public Demo1() {
		super();
		System.out.println("CC-ZPC");
		abc();
	}

	public void abc()
	{
		int a=10;
		System.out.println(a);
		System.out.println(this.a);
		System.out.println(super.a);
	}

}
```

final keyword

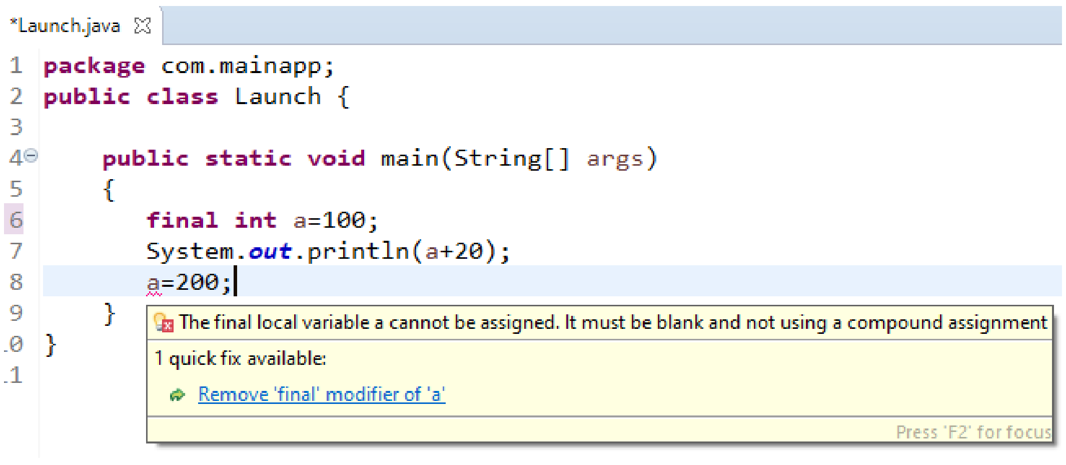
1.variable : we cant change its value

```
public final int i=100;
```

No one can change this if it is already initialized

```
package com.mainapp;
public class Demo
{

    public final int i;

    public Demo(int i) {

      this.i=i; //NO ERROR
    }

    public void setter()
    {
      this.i=200; //ERROR
    }
}
```

We can initialize final variable only through Constructor



*Launch.java

```
package com.mainapp;
public class Launch {

    public static void main(String[] args)
    {
        final int a=100;
        System.out.println(a+20);
        a=200;
    }
}
```

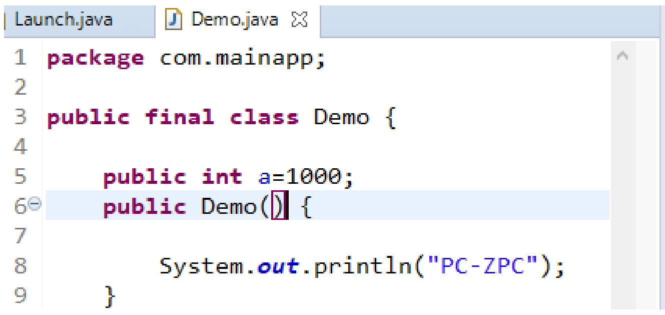
The final local variable a cannot be assigned. It must be blank and not using a compound assignment

1 quick fix available:

Remove 'final' modifier of 'a'

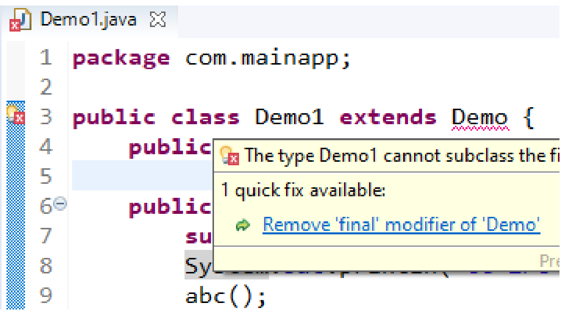
Press 'F2' for focus

2.class : we cant extends it



Launch.java   Demo.java

```java
package com.mainapp;

public final class Demo {

    public int a=1000;
    public Demo() {

        System.out.println("PC-ZPC");
    }
```

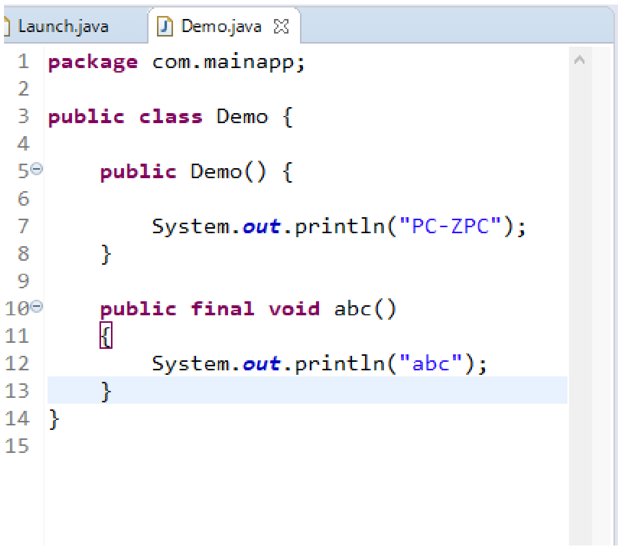


Demo1.java

```java
package com.mainapp;

public class Demo1 extends Demo {
    public

    public
        su
        Sy
        abc();
```

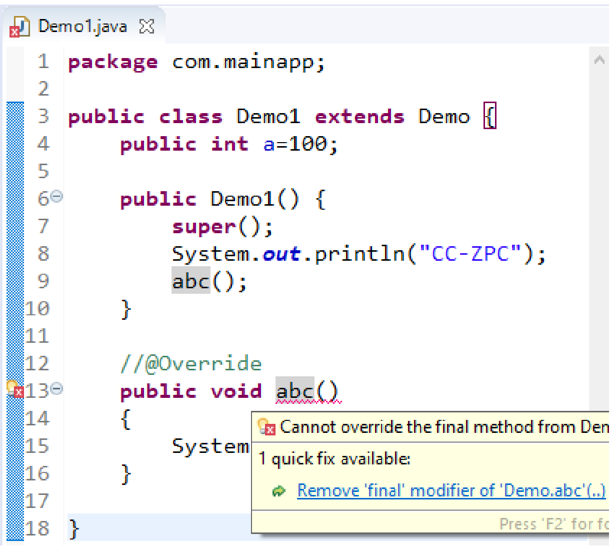
```
The type Demo1 cannot subclass the fi
1 quick fix available:
Remove 'final' modifier of 'Demo'
Pre
```

3.method : we cant override it



Launch.java   Demo.java

```java
package com.mainapp;

public class Demo {

    public Demo() {

        System.out.println("PC-ZPC");
    }

    public final void abc()
    {
        System.out.println("abc");
    }
}

```



Demo1.java

```java
package com.mainapp;

public class Demo1 extends Demo {
    public int a=100;

    public Demo1() {
        super();
        System.out.println("CC-ZPC");
        abc();
    }

    //@Override
    public void abc()
    {
        System
    }

}
```

```
Cannot override the final method from Den
1 quick fix available:
Remove 'final' modifier of 'Demo.abc'(..)
Press 'F2' for fo
```

- We can override final method but if we wants logic of final method then we can it by object object

Polymorphism

->Code Reduction(70-80%)

->Many forms:- if a single statement display multiple result

Polymorphism

1.Compile time polymorphism(static binding or early binding)

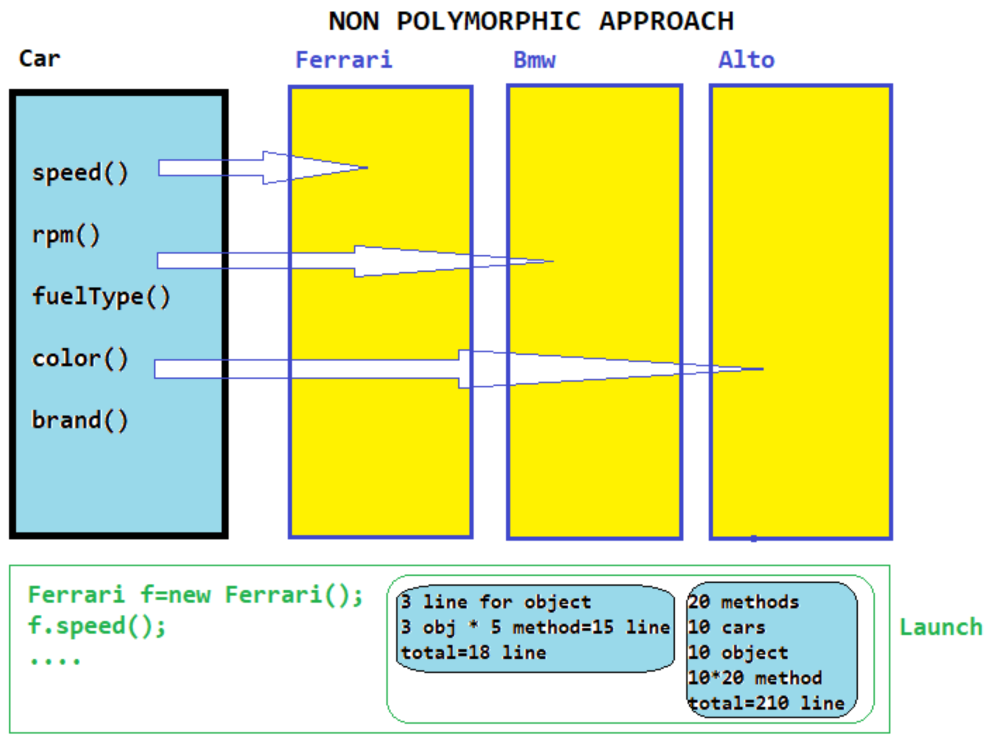
->Method overloading

2.Runtime polymorphism(dynamic binding or late binding)

->Method overriding(Inheritance involved) + Upcasting

Upcasting: holding child object inside Parent reference

```
Car c;
c=new Car();
c=new Ferrari(); //Upcasting
```



```
Ferrari f=new Ferrari();
f.speed();
....
```

3 line for object
3 obj * 5 method=15 line
total=18 line

20 methods
10 cars
10 object
10*20 method
total=210 line

```java
package poly;

public class Launch {

	public static void main(String[] args) {

		Ferrari f = new Ferrari();
		f.speed();
		f.color();
		f.brand();
		f.fuelType();
		f.rpm();

		Bmw b = new Bmw();
		b.speed();
		b.color();
		b.brand();
		b.fuelType();
		b.rpm();

		Alto a = new Alto();
		a.speed();
		a.color();
		a.brand();
		a.fuelType();
		a.rpm();
	}
}
```

```java
package poly;

public class Car {

	public void speed() {
		System.out.println("speed");
	}

	public void rpm() {
		System.out.println("rpm");
	}

	public void fuelType() {
		System.out.println("fuelType");
	}
	public void color() {
		System.out.println("color");
	}

	public void brand() {
		System.out.println("brand");
	}

}


package poly;

public class Ferrari extends Car{

	@Override
	public void speed() {
		System.out.println("300kmph");
	}
	@Override
	public void rpm() {
		System.out.println("12000\n");
	}
	@Override
	public void fuelType() {
		System.out.println("petrol");
	}
	@Override
	public void color() {
		System.out.println("red");
	}
	@Override
	public void brand() {
		System.out.println("ferrari");
	}
}


package poly;

public class Bmw extends Car {
	@Override
	public void speed() {
		System.out.println("200kmph");
	}
	@Override
	public void rpm() {
		System.out.println("8000\n");
	}
	@Override
	public void fuelType() {
		System.out.println("diesel");
	}
	@Override
	public void color() {
		System.out.println("blue");
	}
	@Override
	public void brand() {
		System.out.println("BMW");
	}
}
```

```
package poly;

public class Alto extends Car {

	@Override
	public void speed() {
		System.out.println("180kmph");
	}
	@Override
	public void rpm() {
		System.out.println("4000");
	}
	@Override
	public void fuelType() {
		System.out.println("CNG");
	}
	@Override
	public void color() {
		System.out.println("silver");
	}
	@Override
	public void brand() {
		System.out.println("Suzuki");
	}
}
```

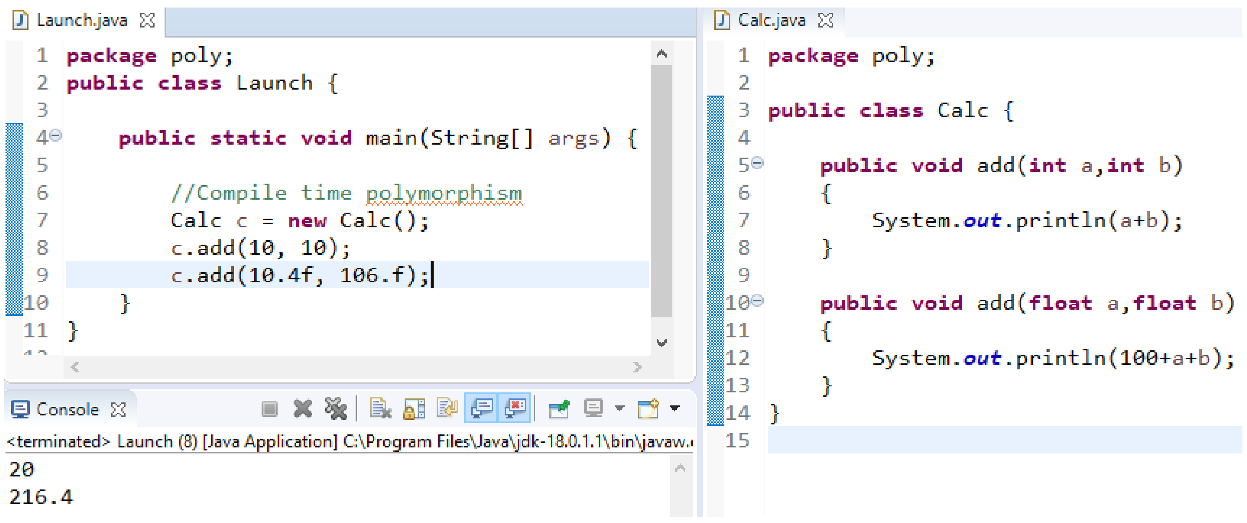
Polymorphic

```
package poly;
public class Car {

	public void speed() {
		System.out.println("speed");
	}

	public void rpm() {
		System.out.println("rpm");
	}

	public void fuelType() {
		System.out.println("fuelType");
	}

	public void color() {
		System.out.println("color");
	}

	public void brand() {
		System.out.println("brand");
	}
}


package poly;
public class Ferrari extends Car{

	@Override
	public void speed() {

		System.out.println("300kmph");
	}
	@Override
	public void rpm() {
		System.out.println("12000\n");
	}
	@Override
	public void fuelType() {
		System.out.println("petrol");
	}
	@Override
	public void color() {
```

```
                System.out.println("red");
        }
        @Override
        public void brand() {
                System.out.println("ferrari");
        }
}


package poly;
public class Bmw extends Car {
        @Override
        public void speed() {
                System.out.println("200kmph");
        }
        @Override
        public void rpm() {
                System.out.println("8000\n");
        }
        @Override
        public void fuelType() {
                System.out.println("diesel");
        }
        @Override
        public void color() {
                System.out.println("blue");
        }
        @Override
        public void brand() {
                System.out.println("BMW");
        }
}


package poly;
public class Alto extends Car {

        @Override
        public void speed() {
                System.out.println("180kmph");
        }
        @Override
        public void rpm() {
                System.out.println("4000");
        }
```

```java
	@Override
	public void fuelType() {
		System.out.println("CNG");
	}
	@Override
	public void color() {
		System.out.println("silver");
	}
	@Override
	public void brand() {
		System.out.println("Suzuki");
	}
}

package poly;
public class Garage {

	public void permit(Car c)
	{
		c.speed();
		c.color();
		c.brand();
		c.fuelType();
		c.rpm();
	}
}
package poly;
public class Launch {

	public static void main(String[] args) {

		Garage g = new Garage();

		Ferrari f = new Ferrari();
		g.permit(f);

		Bmw b = new Bmw();
		g.permit(b);

		Alto a = new Alto();
		g.permit(a);
	}
}
```

Compile time polymorphism



Launch.java

```java
package poly;
public class Launch {

    public static void main(String[] args) {

        //Compile time polymorphism
        Calc c = new Calc();
        c.add(10, 10);
        c.add(10.4f, 106.f);
    }
}
```

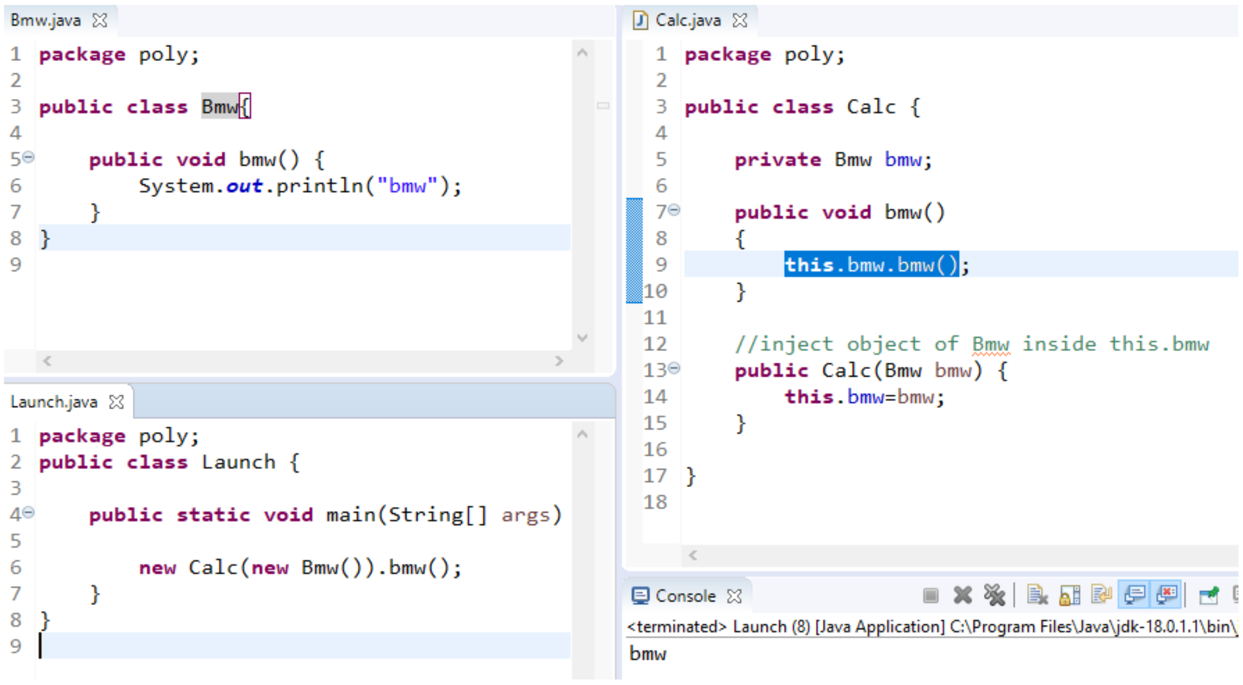
Console

```
<terminated> Launch (8) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.
20
216.4
```

Calc.java

```java
package poly;

public class Calc {

    public void add(int a,int b)
    {
        System.out.println(a+b);
    }

    public void add(float a,float b)
    {
        System.out.println(100+a+b);
    }
}
```

Task



Bmw.java

```java
package poly;

public class Bmw{

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

Launch.java

```java
package poly;
public class Launch {

    public static void main(String[] args)

        new Calc(new Bmw()).bmw();
    }
}
```

Calc.java

```java
package poly;

public class Calc {

    private Bmw bmw;

    public void bmw()
    {
        this.bmw.bmw();
    }

    //inject object of Bmw inside this.bmw
    public Calc(Bmw bmw) {
        this.bmw=bmw;
    }

}
```

Console

```
<terminated> Launch (8) [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\
bmw
```

static keyword

static keyword is mainly used to save memory

we can use static keyword with

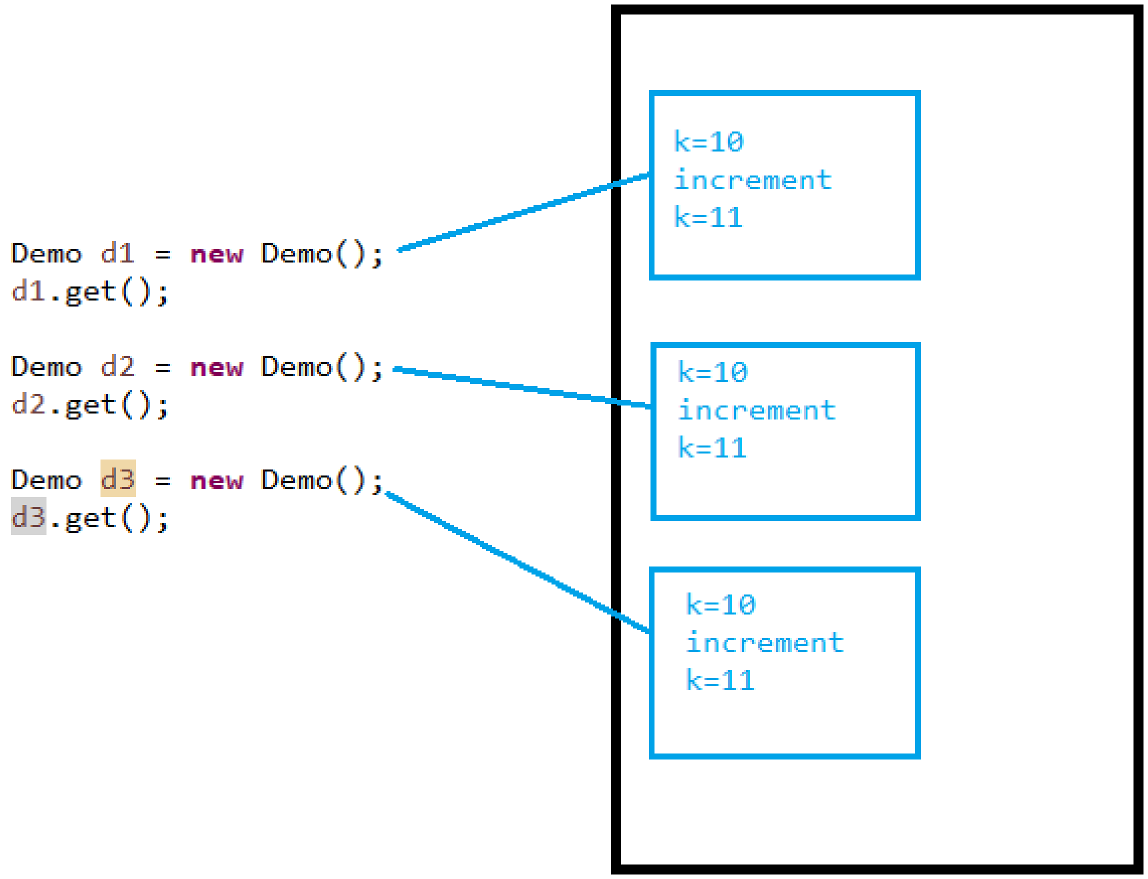
1.block

2.method

3.variable

➢ Static variable

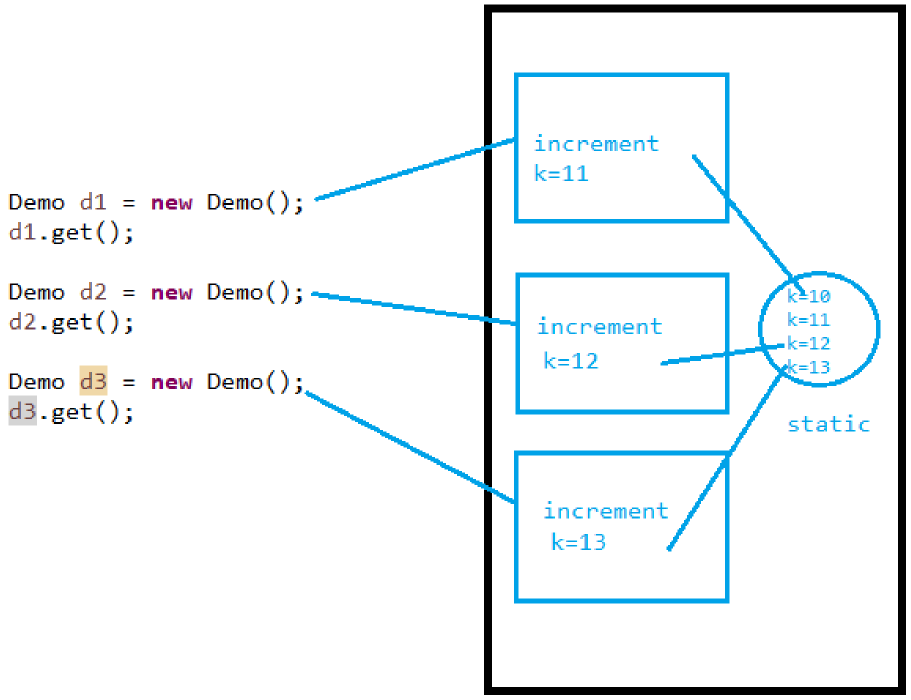
It is not associated with object



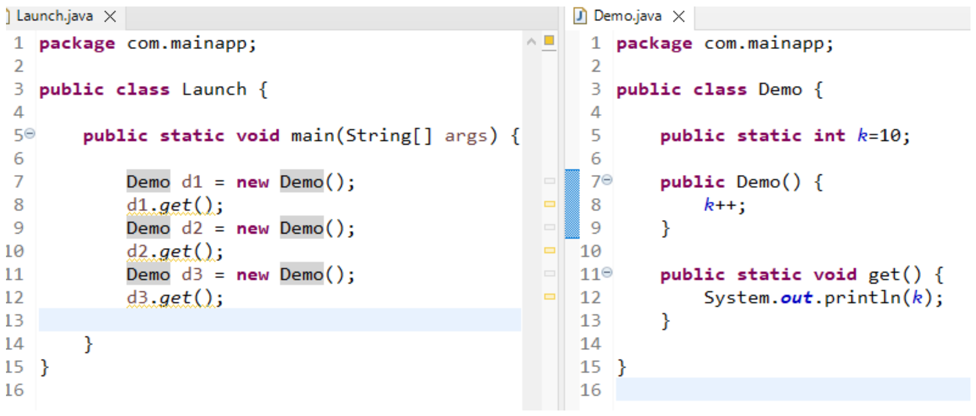
```
Demo d1 = new Demo();
d1.get();

Demo d2 = new Demo();
d2.get();

Demo d3 = new Demo();
d3.get();
```

Ex. Clg: 1000 std

- Roll
- Address
- Phone
- Clg code(Common data from each object)->this should be static to save memory



Example of static variable

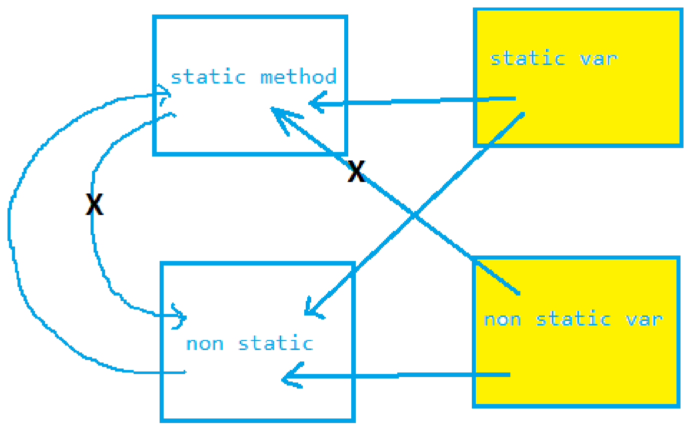


```java
package com.mainapp;

public class Launch {

    public static void main(String[] args) {

        Demo d1 = new Demo();
        d1.get();
        Demo d2 = new Demo();
        d2.get();
        Demo d3 = new Demo();
        d3.get();

    }
}
```

```java
package com.mainapp;

public class Demo {

    public static int k=10;

    public Demo() {
        k++;
    }

    public static void get() {
        System.out.println(k);
    }

}
```

➢ Static method

- To call static method we need not to create object of the class
- Static method is nowhere bound with object
- `you cant use this keyword here coz we are calling static method without object creation`

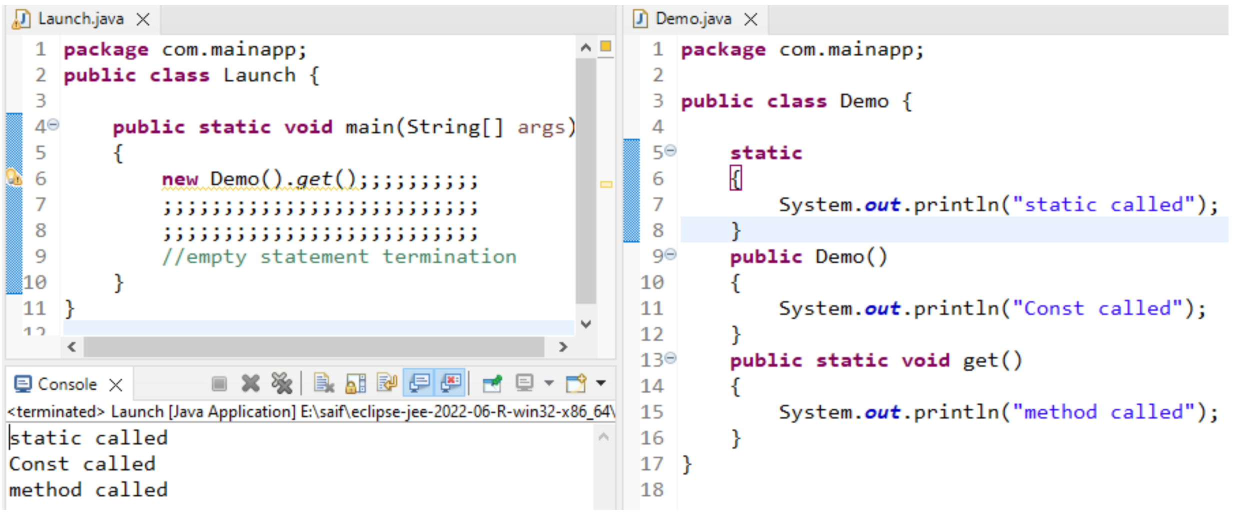


Static block

```
package com.mainapp;
public class Launch {

	public static void main(String[] args)
	{
		Demo.get();
	}
}
package com.mainapp;
public class Demo {

	public static int a;
	static
	{
		System.out.println("static called");
		a=100;
	}

	public static void get() {
```

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

static,constructor,abstraction



```
package com.mainapp;
public class Launch {

    public static void main(String[] args)
    {
        new Demo().get();;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;
        //empty statement termination
    }
}
```

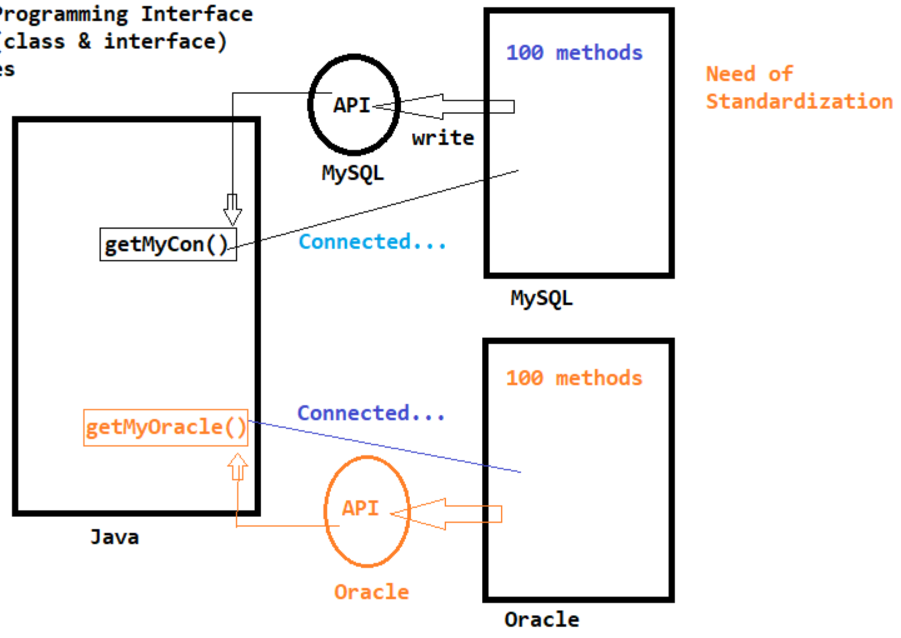
```
<terminated> Launch [Java Application] E:\saif\eclipse-jee-2022-06-R-win32-x86_64\
static called
Const called
method called
```

```
package com.mainapp;

public class Demo {

    static
    {
        System.out.println("static called");
    }
    public Demo()
    {
        System.out.println("Const called");
    }
    public static void get()
    {
        System.out.println("method called");
    }
}
```

Abstraction

- Hide implementation and use functionality

API: Application Programming Interface
It contains codes(class & interface)
Packages, libraries



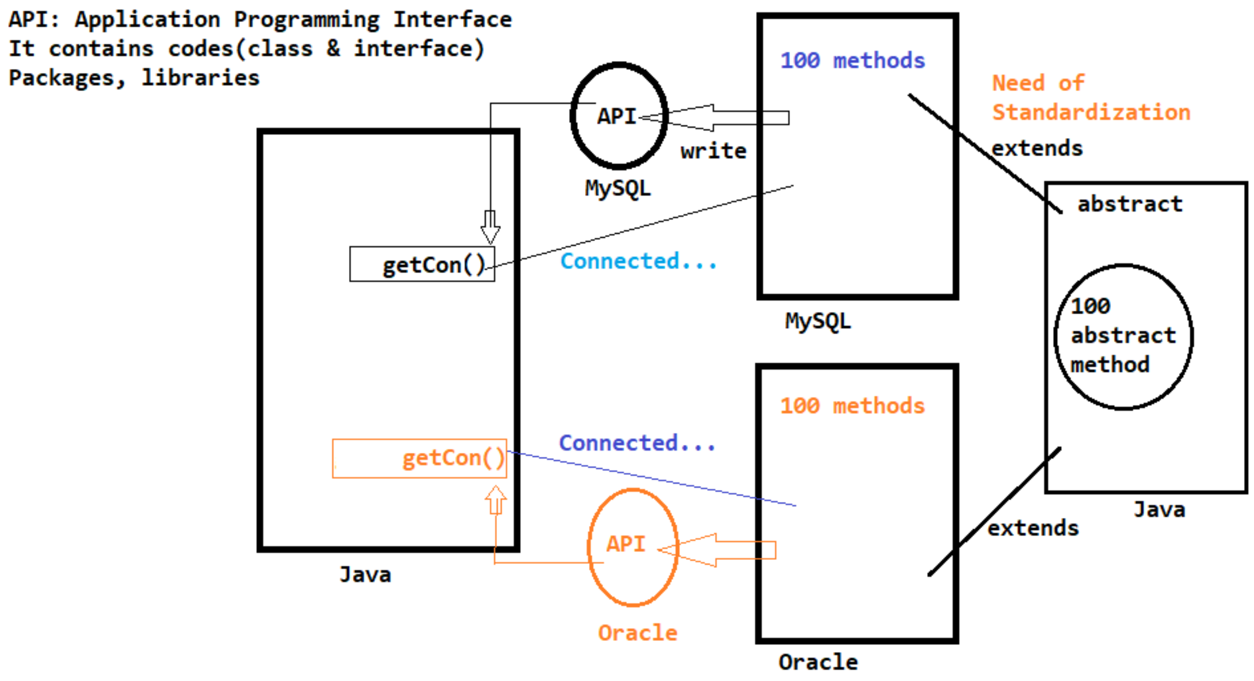
- We can achieve standardization through **abstract class** and interface
- Abstract class can be empty or it can have abstract or non abstract method or combination of both

Program: Standardization

```
package com.mainapp;
public abstract class Standard {

	 //abstract method
	public abstract void insert();
	public abstract void delete();
	public abstract void update();
	public abstract void read();

}
package com.mainapp;
public class Demo extends Standard {

	@Override
	public void insert() {
		// TODO Auto-generated method stub

	}

	@Override
	public void delete() {
		// TODO Auto-generated method stub
	}

	@Override
	public void update() {
		// TODO Auto-generated method stub
	}

	@Override
	public void read() {
		// TODO Auto-generated method stub
	}
}
```

- We cant instantiate abstract class coz abstract class is an incomplete class and java doesn’t allow class loading of incomplete class

Standardization Example



Factory Design Patter(Abstraction example)

```
package com.mainapp;
import com.factory.ConnectionFactory;
import com.standard.Connection;

public class Launch {

	public static void main(String[] args) {
	Connection c=ConnectionFactory.getObj("mysql");
	 c.getConnection();
	}
}

package com.standard;
public abstract class Connection {

	public abstract void getConnection();

}
```

```
package com.databases;
import com.standard.Connection;

public class Oracle extends Connection {

	@Override
	public void getConnection() {

		System.out.println("Connect with oracle");
	}
}
package com.databases;
import com.standard.Connection;

public class MySQL extends Connection {

	@Override
	public void getConnection() {
		System.out.println("Connect with mysql");

	}
}

package com.factory;
import com.databases.MySQL;
import com.databases.Oracle;
import com.standard.Connection;

public class ConnectionFactory {

	public static Connection getObj(String s)
	{
		if(s.equalsIgnoreCase("mysql"))
		{
			return new MySQL();
		}
		else if(s.equalsIgnoreCase("oracle"))
		{
			return new Oracle();
		}

		return null;
	}
}
```

Interface

- it is used to achieve abstraction(100%),standardizatio and multiple inheritance
- We cant instantiate interface

- Interface consists abstract method
- Java 8->static,default
- Java 9->private

```
public abstract interface DataBaseOperation {

	public abstract void insert();

}
```

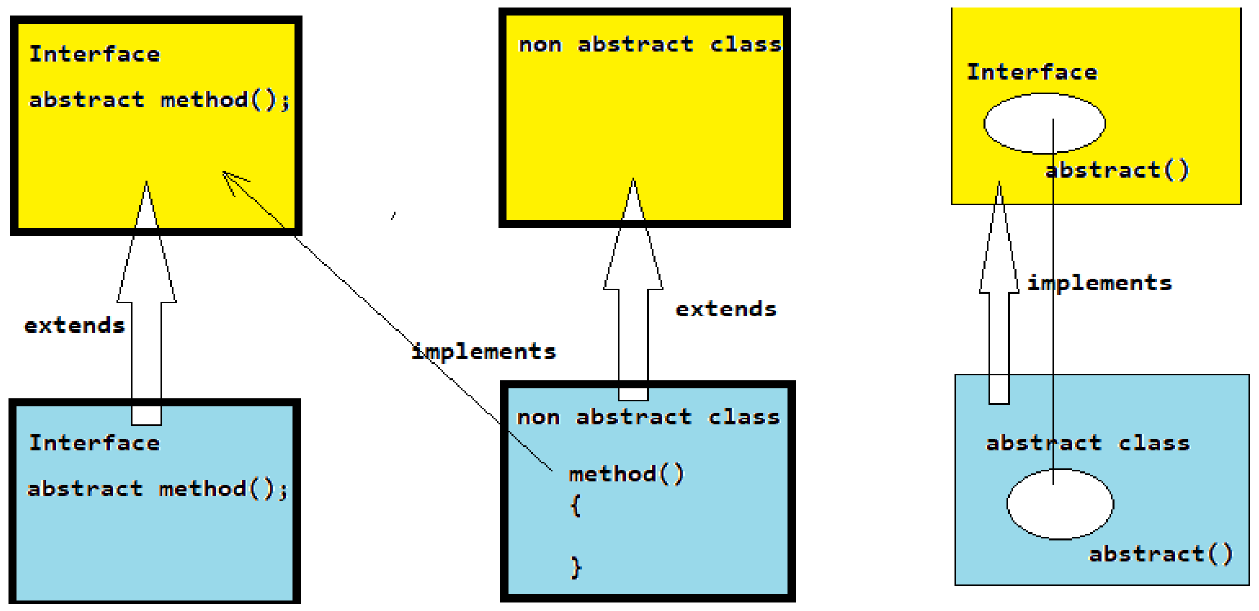
- in interface abstract keyword is optional
- We can extends and implements simulteneously but extends comes first

Multiple Inheritance:

```
public class Employee extends ConnectionApi implements
UserModule,DataBaseOperation
```

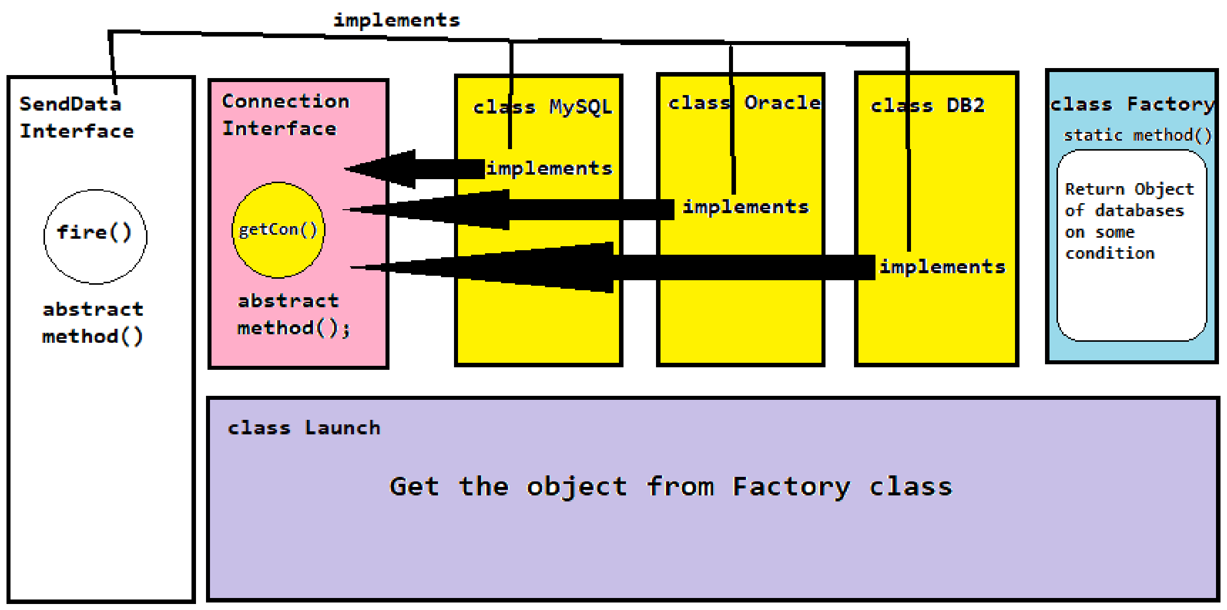
```
package com.interfaces;
public abstract interface DataBaseOperation {

	public abstract void insert();
	public abstract void update();
	public abstract void delete();
	public abstract void read();

}


package com.abstracts;

public abstract class ConnectionApi {

	public abstract void mysql();
	public abstract void oracle();

}


package com.interfaces;

public interface UserModule {

	public void login();
	public void logout();

}


package com.dao;

import com.abstracts.ConnectionApi;
import com.interfaces.DataBaseOperation;
import com.interfaces.UserModule;

public class Employee extends ConnectionApi implements
UserModule,DataBaseOperation  {

	@Override
	public void login() {
		// TODO Auto-generated method stub

	}

	@Override
	public void logout() {
		// TODO Auto-generated method stub
```

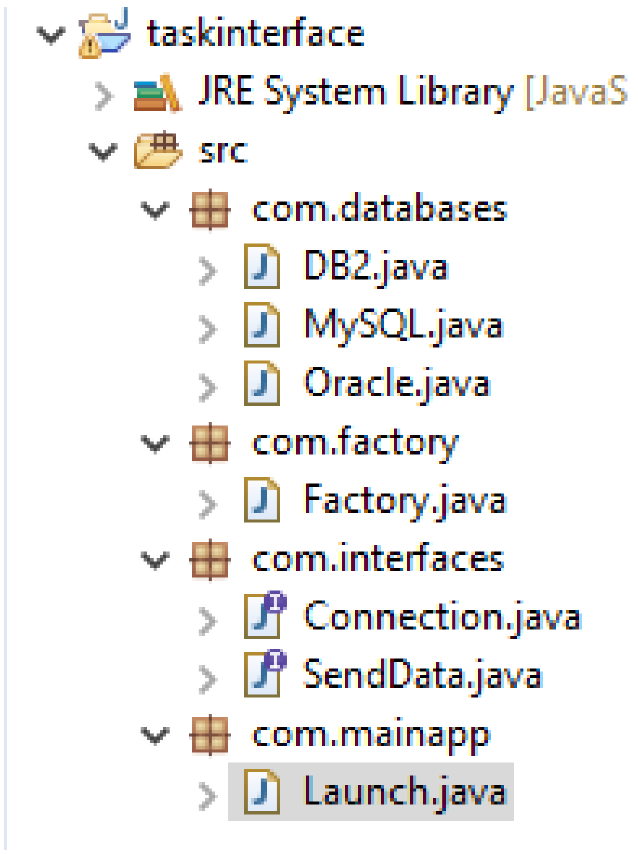
```java
	}
    ////////////////////////////////////////////////////////
	@Override
	public void insert() {
		// TODO Auto-generated method stub
		
	}

	@Override
	public void update() {
		// TODO Auto-generated method stub
		
	}

	@Override
	public void delete() {
		// TODO Auto-generated method stub
		
	}

	@Override
	public void read() {
		// TODO Auto-generated method stub
		
	}
     ///////////////////////////////////////////////////////////
	@Override
	public void mysql() {
		// TODO Auto-generated method stub
		
	}

	@Override
	public void oracle() {
		// TODO Auto-generated method stub
		
	}

}
```



Note: Implements keyword comes only with Interface but extends can be used with both class and interface

Factory Design Pattern(Abstraction & Standardization)





```
taskinterface
  JRE System Library [JavaS
  src
    com.databases
      DB2.java
      MySQL.java
      Oracle.java
    com.factory
      Factory.java
    com.interfaces
      Connection.java
      SendData.java
    com.mainapp
      Launch.java
```

```java
package com.interfaces;
public interface SendData {

	public void fire();

}

package com.interfaces;
public interface Connection extends SendData{

	public void getCon();

}

package com.databases;
import com.interfaces.Connection;
import com.interfaces.SendData;
public class MySQL implements SendData,Connection {

	@Override
	public void getCon() {
		System.out.println("Mysql connection");

	}
	@Override
	public void fire() {
		System.out.println("Mysql query fire");


	}
}

package com.databases;
import com.interfaces.Connection;
import com.interfaces.SendData;

public class DB2  implements SendData,Connection  {

	@Override
	public void getCon() {
		System.out.println("Db2 connection");

	}
	@Override
	public void fire() {
		System.out.println("Db2 query fire");

	}
}

package com.databases;
import com.interfaces.Connection;
import com.interfaces.SendData;
public class Oracle  implements SendData,Connection  {

	@Override
	public void getCon() {
		System.out.println("Oracle connection");

	}
	@Override
	public void fire() {
		System.out.println("Oracle query fire");

	}
}
```

```java
package com.factory;
import com.databases.DB2;
import com.databases.MySQL;
import com.databases.Oracle;
import com.interfaces.Connection;
public class Factory {

	public static Connection getConnection(String s) {

		if(s.equalsIgnoreCase("mysql"))
		{
			return new MySQL();
		}
		else if(s.equalsIgnoreCase("oracle"))
		{
			return new Oracle();
		}
		else if(s.equalsIgnoreCase("db2"));
		{
			return new DB2();
		}
	}
}
package com.mainapp;
import com.factory.Factory;
import com.interfaces.Connection;
public class Launch {

	public static void main(String[] args) {

		Connection con = Factory.getConnection("mysql");
		con.getCon();
		con.fire();
	}
}
```

Exception Handling

Program: Division of two number

```
package com.mainapp;
import java.util.Scanner;

public class Launch {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter first digit");
		int a=sc.nextInt();
		System.out.println("Enter second digit");
		int b=sc.nextInt();
		System.out.println("divison "+(a/b));
	}
}
Enter first digit
ten
Exception in thread "main" java.util.InputMismatchException
	at java.base/java.util.Scanner.throwFor(Scanner.java:943)
	at java.base/java.util.Scanner.next(Scanner.java:1598)
	at java.base/java.util.Scanner.nextInt(Scanner.java:2263)
	at java.base/java.util.Scanner.nextInt(Scanner.java:2217)
	at com.mainapp.Launch.main(Launch.java:10)
```

- Exception Occurs by giving faulty inputs
- Exception is alsp knows as abnormal termination

- We can handle Exception by using two ways

1.try-catch block

2.throws keyword

Note:

- Exception is not error
- Exception is not RuntimeError : STACKOVERFLOW ERROR

Exception is of two types

1.checked Exception

->During compile time

->try-catch block and throws keyword

2.Unchecked Exception

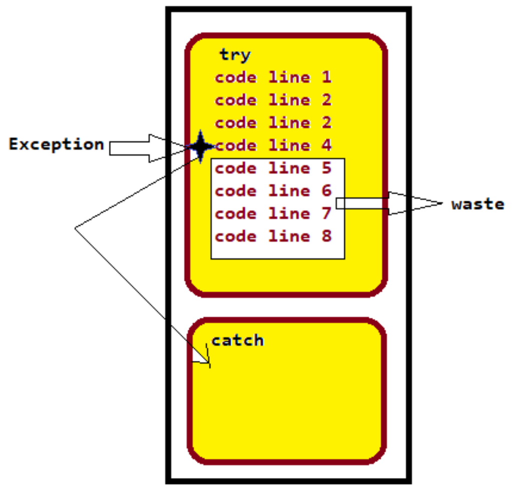
->During Runtime

->try-catch block

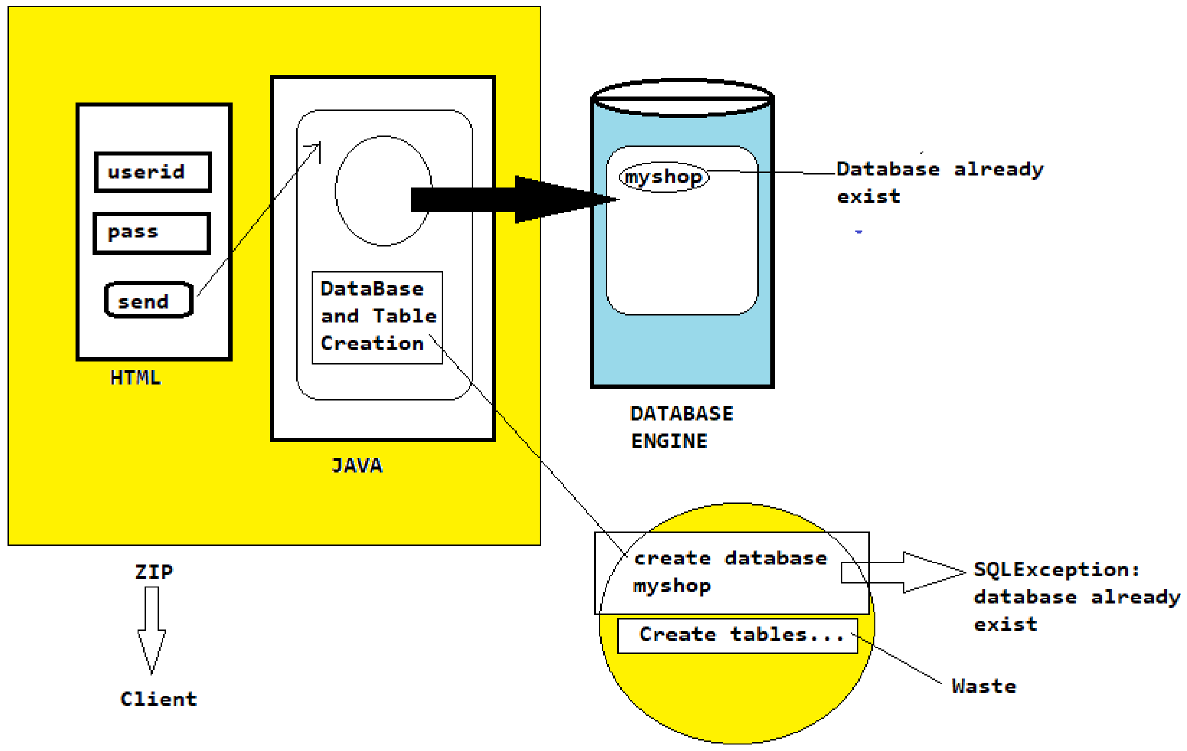
```
package com.mainapp;
import java.util.Scanner;
public class Launch {
      public static void main(String[] args) {
            //Unchecked: Runtime
            try
            {
                  Scanner sc = new Scanner(System.in);
                  System.out.println("Enter first digit");
                  int a=sc.nextInt();
                  System.out.println("Enter second digit");
                  int b=sc.nextInt();
                  System.out.println("divison "+(a/b));
            }
            catch(Exception e)
            {
```

```
                        System.out.println("Insert digit only(ex. 10)");
                        main(args);
                    }
                }
            }
```

Problems with try catch block



Database table creation problem



Solution: finally block

- If exception occurs it will run after catch block otherwise it will run just after try block

```
package com.mainapp;
import java.util.Scanner;
public class Launch {

	public static void main(String[] args) {

		//Unchecked: Runtime
		try
		{
			//module 1
			Scanner sc = new Scanner(System.in);
			System.out.println("Enter first digit");
			int a=sc.nextInt();
			System.out.println("Enter second digit");
			int b=sc.nextInt();
			System.out.println("divison "+(a/b)+"\n\nSUM\n");
		}
		catch(Exception e)
		{
			System.out.println("Insert digit only(ex. 10)");
		}
		finally
		{
			try
			{
				//module 2
				Scanner sc1 = new Scanner(System.in);
				System.out.println("Enter first digit");
				int aa = sc1.nextInt();
				System.out.println("Enter second digit");
				int bb = sc1.nextInt();
				System.out.println("Sum " + (aa + bb));
			}
			catch (Exception e2) {

				System.out.println("Insert digit only(ex. 10)");
				System.out.println(e2);
			}
```

```
                finally
                {
                        //module 3
                }
            }
        }
}
```

catch block

```
catch(Exception e)
{
    System.out.println("Insert digit only(ex. 10)");

}
```

- Exception class is the Parent class of all the Exception class so here we can handle all types of exception

```
catch(NullPointerException n)
{
        System.out.println("Insert digit only(ex. 10)");

}
```

- It can handle only null pointer exception

```
catch(NullPointerException | ArithmeticException a)
{
        System.out.println("Insert digit only(ex. 10)");

}
```

- It can handle only null pointer exception and Arithmetic Exception

Checked Exception

ClassNotFoundException

```java
package com.mainapp;
public class Launch {

	public static void main(String[] args) {

		try
		{
		Class.forName("java.lang.String"); //class loading
		}
		catch (Exception e)
		{
			System.out.println(e);
		}

	}
}

//class->keyword Class->Predefined Class(java.lang)
```

SQLException

```java
package com.mainapp;
import java.sql.DriverManager;
import java.sql.SQLException;
public class Launch {

	public static void main(String[] args) {

		try {
			DriverManager.getConnection("");
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			System.out.println(e);
		}

	}
}

//class->keyword Class->Predefined Class(java.lang)
```

throws keyword

it is used to throw exception to the JVM and then JVM will handle accordingly

```
package com.mainapp;
import java.sql.DriverManager;
public class Launch {

	public static void main(String[] args) throws Exception {

			DriverManager.getConnection("");
	}
}

//class->keyword Class->Predefined Class(java.lang)
```

Drabacks of checked Exception and throws keyword

- Checked Exception is a drawback coz it force developer to handle exception whether it would or not
- In Spring or Hibernate there is no checked type exception

```
package com.mainapp;
import java.sql.DriverManager;

public class Demo {

	public void abc() throws Exception
	{

	   DriverManager.getConnection("");
	    //It Will Impact Launch class by throwing
	    Exception so preferred approach is try and
	    catch block
//		try
//		{
//			DriverManager.getConnection("");
//			System.out.println("connected");
//		}
//		catch (SQLException e)
//		{
//			System.out.println("something went worng");
//		}

	}
}
```

```
package com.mainapp;
public class Launch {

	public static void main(String[] args) throws Exception  {

			new Demo().abc();
	}
}
```

Valid Syntax:

```
package com.mainapp;
import java.sql.DriverManager;
import java.sql.SQLException;
public class Demo {
	public void abc()
	{
		try {
			try {
				DriverManager.getConnection("");
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			int a=10/0;
			System.out.println("connected");
		}
		finally
		{
			System.out.println("finally");
		}
	}
}
```

- try with finally : valid
- only try: invalid
- try-catch-finally: valid
- try { try-catch-finally } finally{} : valid
- try {try-catch{ try-catch-finally } } finally{ try-catch-finally } : valid

ATM

Cash<100

- *Exception: try again*

Cash<100

- Exception: try again

Cash<100

- Exception: try again

Program terminate

```
package com.mainapp;

import java.util.Scanner;

public class Launch {

	public static void main(String[] args) throws Exception  {

				Scanner s = new Scanner(System.in);
				System.out.println("Enter amount");
				int i = s.nextInt();
				if(i<100)
				{
					try
					{
					  int a=10/0;
					}
					catch(Exception e)
```

```
{
        System.out.println(e);
}
finally
{
        System.out.println("Enter amount");
        i = s.nextInt();
        if(i<100)
        {
                try
                {
                   int a=10/0;
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
                finally
                {
                System.out.println("Enter amount");
                        i = s.nextInt();
                        if(i<100)
                        {
                                try
                                {
                                   int a=10/0;
                                }
                                catch(Exception e)
                                {
                                System.out.println(e);
                                }
                                finally
                                {
                                System.exit(0);
                                //program termination
                                }
                        }
                        else
                        {
        System.out.println("Take Amount : "+i);
                        }
                }
        }
        else
        {
        System.out.println("Take Amount : "+i);
```

```
                        }
                    }
                }
                else
                {
                    System.out.println("Take Amount : "+i);
                }
        }
}
```

CUSTOM EXCEPTION

Some Exception Cases:

- 10/0
- nextInt(): string
- Class.forName("wrong location")

Ex.Need of Custom Exception

Gym Application

Registration From

- Name
- Email
- Age <18 ->faulty input (CUSTOM EXCEPTION)
- Blood Gr

➢ Here we need to throw exception when user enter age<18

Note:What is the diff between throw,throws,throwable

throw: it is a keyword which generates CUSTOM EXCEPTION

```
package com.mainapp;
import java.util.Scanner;

public class Launch {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter your age");
		int age = sc.nextInt();
		if(age<18)
		{
//Custom Exception
throw new ArithmeticException("age should be greater than or equal to 18");
		}
		else
		{
			System.out.println("eligible");
		}
	}
}
```

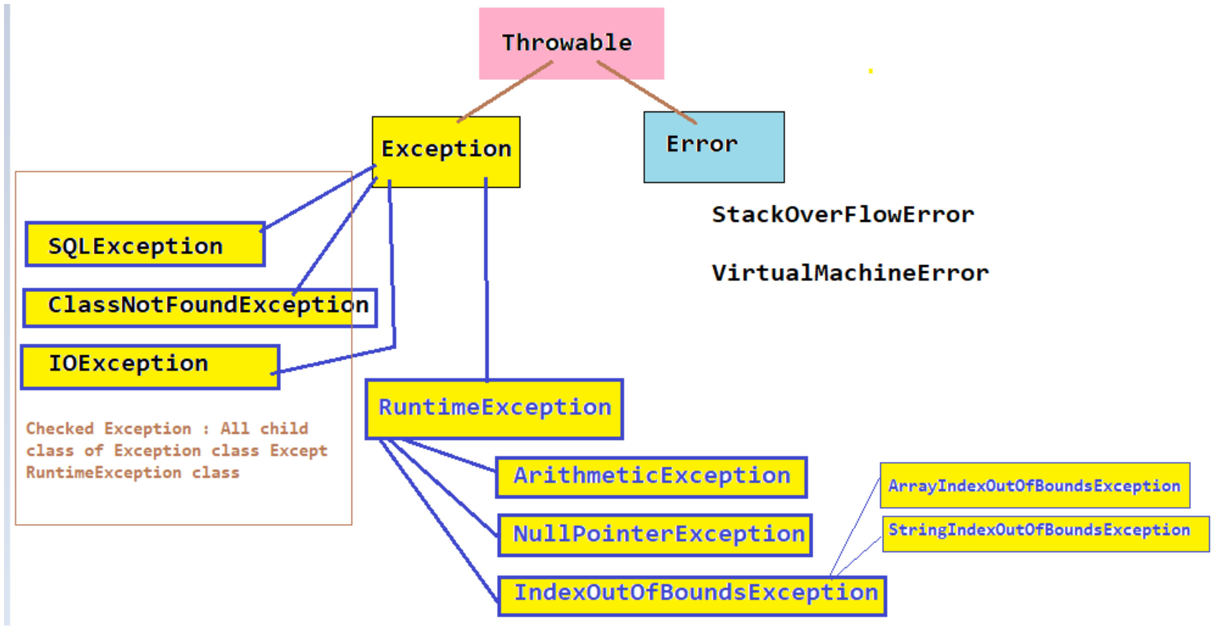
- We can handle exception generated by throw keyword by using throws keyword (throws on throw)

```
package com.mainapp;
import java.util.Scanner;
public class Launch {

	public static void main(String[] args) throws Exception {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter your age");
		int age = sc.nextInt();
		if(age<18)
		{
	//Custom Exception
	throw new Exception("age should be greater than or equal to 18");
		}
		else
		{
			System.out.println("eligible");
		}
	}
}
```

- We can handle exception generated by throw keyword by using try catch block (try catch on throw)

```
package com.mainapp;
import java.util.Scanner;
public class Launch {

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter your age");
		int age = sc.nextInt();

		if(age<18)
		{
			//Custom Exception
			try
			{
	throw new Exception("age should be greater than or equal to 18");
			}
			catch (Exception e)
			{
				System.out.println(e);
				e.printStackTrace();//trace exception
			}
		}
		else
		{
			System.out.println("eligible");
		}
	}
}
```

Exception Hierarchy


Throwable
Exception
Error
StackOverFlowError
VirtualMachineError
SQLException
ClassNotFoundException
IOException
Checked Exception : All child class of Exception class Except RuntimeException class
RuntimeException
ArithmeticException
NullPointerException
IndexOutOfBoundsException
ArrayIndexOutOfBoundsException
StringIndexOutOfBoundsException

Collection Framework

- Collection framework provides inbuild data structure
- DataStructure:

Ex.

- ARRAYLIST
- LINKEDLIST
- VECTOR

- STACK

- PRIORITYQUEUE
- QUEUE
- DEQUEUE

- HASHSET
- LINKEDHASHSET
- TREESET

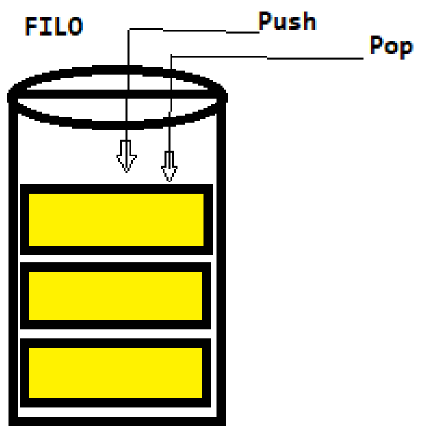
- HASHMAP
- LINKEDHASHMAP
- TREEMAP

WHY DATASTRUCURE IS IMPORTANT ?

it holds the data on some restrictions

Array->Basic DataStructure

Ex. STACK is a dataStructure

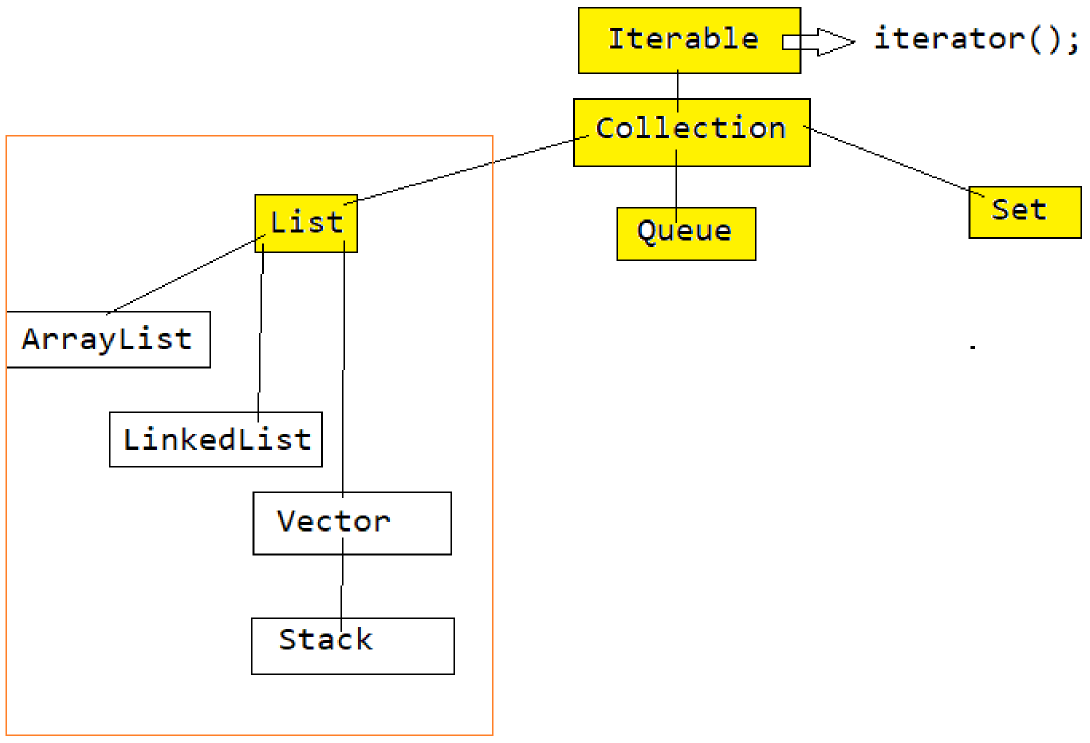


Array has many drawback

1.fixed size

2.homogeneous

3.contiguous memory location



What is Framework ?

- Framework may contain packages and libraries and sometime API's
- Framework provides an inbuilt control flow in which it will only calls our code and manage accordingly

- Library-> Categorize packages
- Package->Categorize classes or interfaces
- API->Bridge between two application
  Ex. phonepay inside amazon

1.ArrayList

ArrayList can be Generic or non generic

Generic->For Specific Datatype(Ex. String)

Non Generic->Object type

```
package com.list;
import java.util.ArrayList;
import java.util.Iterator;
public class MyArrayList {

	public static void main(String[] args) {

		  //No Primitive Supported
		ArrayList<Object> al = new ArrayList<Object>();
		al.add(15);
		al.add(true);
		al.add("hello");
		al.add(15.f);
		al.add('&');

		System.out.println(al);
		al.add("ewrwerr");
		System.out.println(al);

		//How to iterate
		Iterator<Object> itr = al.iterator();//ALT SHIFT L
		while(itr.hasNext())
		{
		  System.out.println(itr.next());
		}
	}
}
```

Add elements

```
package com.list;
import java.util.ArrayList;
import java.util.Iterator;
public class MyArrayList {

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

```java
		  //No Primitive Supported
		  ArrayList<Object> al0 = new ArrayList<Object>();
		al0.add("raju");
		al0.add("kaju");


		ArrayList<Object> al = new ArrayList<Object>();
		al.add(15);
		al.add(true);
		al.add("hello");
		al.add(15.f);
		al.add('&');

		System.out.println(al);

		al.add(2, "surya");

		System.out.println(al);

		al.addAll(al0);

		System.out.println(al);

		al.addAll(0,al0);

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

- Remove(“object”)->It will delete only first object of given type

```java
package com.list;
import java.util.ArrayList;
public class MyArrayList {

	public static void main(String[] args) {

		  //No Primitive Supported

		ArrayList<Object> al = new ArrayList<Object>();
		al.add(15);
		al.add(true);
		al.add("hello");
		al.add(15);
		al.add(15.f);
		al.add("hello");
		al.add('&');
		
		System.out.println(al);
		al.remove("hello");
		System.out.println(al);
	}
}
OUTPUT:
[15, true, hello, 15, 15.0, hello, &]
[15, true, 15, 15.0, hello, &]
```

- If you are deleting element on the basic of int value then remove() will treat value as index of Collection and if you are giving Integer type value (`Integer i = new ~~Integer~~(15);`) then remove() will treat as Data

```
package com.list;
import java.util.ArrayList;
import java.util.Iterator;
public class MyArrayList {

	public static void main(String[] args) {

		 //No Primitive Supported

		ArrayList<Object> al = new ArrayList<Object>();
		al.add(15);
		al.add(true);
		al.add("hello");
		al.add(15);
		al.add(15.f);
		al.add("hello");
		al.add('&');
		
		System.out.println(al);
		Integer i = new ~~Integer~~(15);
```

```
            al.remove(i);
            System.out.println(al);
      }
}
```

```
[15, true, hello, 15, 15.0, hello, &]
[true, hello, 15, 15.0, hello, &]
```

We can remove elements on the basic of

1. index
2. object
3. collection

```
package com.list;
import java.util.ArrayList;
import java.util.Iterator;
public class MyArrayList {

      public static void main(String[] args) {

             //No Primitive Supported

             ArrayList<Object> al0 = new ArrayList<Object>();
             al0.add("raju");
             al0.add("kaju");
             al0.add("sachin");

            ArrayList<Object> al = new ArrayList<Object>();
            al.add(15);
            al.add(true);
            al.add("hello");
            al.add("raju");
            al.add("kaju");

            System.out.println(al);
            al.removeAll(al0);
            System.out.println(al);

             al.clear();
            System.out.println(al);
```

```
	}
}
```

Generic Collection : it can store only similar data type

```
ArrayList<String> al0 = new ArrayList<String>();
al0.add("raju");
al0.add("kaju");
al0.add("sachin");
```

Iterate Using for loop

```
package com.list;
import java.util.ArrayList;
public class MyArrayList {
	public static void main(String[] args) {

		ArrayList<String> al0 = new ArrayList<String>();
		al0.add("raju");
		al0.add("kaju");
		al0.add("sachin");
		for(int i=0;i<al0.size();i++)
		{
			System.out.println(al0.get(i));
		}
	}
}
```

Task

WAP to make one employee Management system like

WELCOME TO MY APPLICATION

Number of Employee: userinput //10

Press 1: Add Employee

Enter Employee Details:

- Enter Id :user input
- Enter name :user input
- Enter Address :user input
- Enter Salary :user input

Press 3: Update Employee on the basis of eid

- Enter Id :user input
- Enter New Name :user input
- Enter New Address :user input
- Enter New Salary :user input

Press 3: ReadALL

Press 4: Delete

Press 4: Exit

Here are some Restrictions

Employee data->

eid(int unique),ename(string),eaddress(string),esalary(int)

-> eid is unchangable, eid<1000,eid>0 : incorrect Id

-> 0< ename <20 character : incorrect name

-> 0< eaddress <200 character : incorrect address

Launch

```
package com.mainapp;
import java.util.ArrayList;
import java.util.Scanner;
```

```java
import com.pojo.Employee;
import com.validation.Validation;
public class Launch {
      public static void main(String[] args) {

            ArrayList<Employee> al = new ArrayList<Employee>();
            Scanner sc = new Scanner(System.in);
            while (true) {
                  System.out.println("WELCOME TO MY APPLICATION");
                  System.out.println("Press 1->add\nPress 2->update");
                  System.out.println("Press 3->delete\nPress 4-
                                    >read\npress 5->exit");
                  System.out.println("Enter Choice: ");
                  int choice = sc.nextInt();
                  if(choice==5)break;

                  switch (choice) {
                  case 1:
                        add(sc, al);
                        break;
                  case 2:
                        update(sc,al);
                        break;
                  case 3:
                        delete(sc,al);
                        break;
                  case 4:
                        read(al);
                        break;
                  default:
                        System.out.println("Enter Correct Choice");
                        break;
                  }
            }
      }
      private static void delete(Scanner sc, ArrayList<Employee> al) {

            System.out.println("\n****Delete Employee Section****\n");
            System.out.println("Enter Employee Id");
            int eid = sc.nextInt();
            int count=0;
            for(int i=0;i<al.size();i++)//emp1  emp2
            {
                  Employee e=al.get(i);
                  int geteid = e.getEid();
```

```
				if(eid==geteid)
				{
					al.remove(i);
					System.out.println("record deleted");
					count++;
					break;
				}
			}
			if(count!=1)System.out.println("Id doesnt exist");
		}

		private static void update(Scanner sc, ArrayList<Employee> al) {

			System.out.println("\n****Update Employee Section****\n");

			System.out.println("Enter Employee Id");
			int eid = sc.nextInt();

			System.out.println("Enter New Employee Name:");
			String ename = sc.next();

			System.out.println("Enter New Employee Address:");
			String eaddress = sc.next();

			System.out.println("Enter New Employee Salary:");
			int esalary = sc.nextInt();

			int count=0;
			for(int i=0;i<al.size();i++)//emp1  emp2
			{
				Employee e=al.get(i);
				int geteid = e.getEid();

				if(eid==geteid)
				{
					e.setEname(ename);
					e.setEaddress(eaddress);
					e.setEsalary(esalary);

					al.set(i, e);//update
					count++;
					System.out.println("Record Updated");
					break;
				}
			}
			if(count!=1)System.out.println("Id doesnt exist");
```

```
}

private static void read(ArrayList<Employee> al) {

      System.out.println("\n****Read Employee Section****\n");

      for(Employee e:al)
      {
            System.out.println(e.toString());
      }
}

private static void add(Scanner sc, ArrayList<Employee> al) {

      System.out.println("\n****Add Employee Section****\n");
      while (true) {
            System.out.println("Enter Employee Id:");
            int eid = sc.nextInt(); sc.nextLine();

            System.out.println("Enter Employee Name:");
            String ename = sc.nextLine();
            System.out.println(ename);

            System.out.println("Enter Employee Address:");
            String eaddress = sc.next();

            System.out.println("Enter Employee Salary:");
            int esalary = sc.nextInt();

//validation
String res = Validation.validate(eid,ename,eaddress,esalary);
            if(res.equals("valid"))
            {

      Employee emp = new Employee(eid, ename, eaddress, esalary);
      al.add(emp);

      System.out.println("Do you want to continue: Y/N");
      char c=sc.next().charAt(0);
      if(c=='N' || c=='n')break;
            }
            else
            {
                  System.out.println(res);
                  break;
            }
```

```
				}
		}
}
```

Employee

```
package com.pojo;
public class Employee {

	private final int eid;
	private String ename;
	private String eaddress;
	private int esalary;

public Employee(int eid, String ename, String eaddress, int esalary) {
		super();
		this.eid = eid;
		this.ename = ename;
		this.eaddress = eaddress;
		this.esalary = esalary;
	}

	public int getEid() {
		return eid;
	}

	public String getEname() {
		return ename;
	}
	public void setEname(String ename) {
		this.ename = ename;
	}
	public String getEaddress() {
		return eaddress;
	}
	public void setEaddress(String eaddress) {
		this.eaddress = eaddress;
	}
	public int getEsalary() {
		return esalary;
	}
	public void setEsalary(int esalary) {
		this.esalary = esalary;
	}
@Override
public String toString() {
return "Employee [eid=" + eid + ", ename=" + ename + ", eaddress=" +
eaddress + ", esalary=" + esalary + "]";
      }
}
```

Validation

```
package com.validation;
public class Validation {

private static String result;
public static String validate(int eid, String ename, String eaddress,
int esalary) {
            if(eid<0 ||  eid>1000)
            {
                  result="invalid employee id";
            }
            else if(ename.length()<2 || ename.length()>20)
            {
                  result="invalid employee name";
            }
            else if(eaddress.length()<5 || eaddress.length()>200)
            {
                  result="invalid employee address";
            }
            else if(esalary<0 ||  esalary>50000)
            {
                  result="invalid employee salary";
            }
            else
            {
                  result="valid";
            }
            return result;
      }
}
```

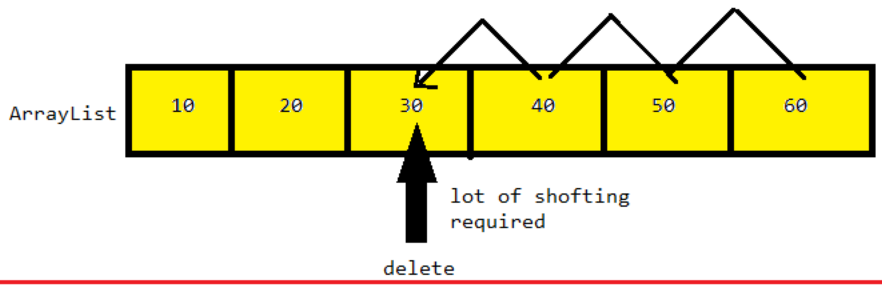
In above program we have manipulated data by using ArrayList but this is not preferred approach

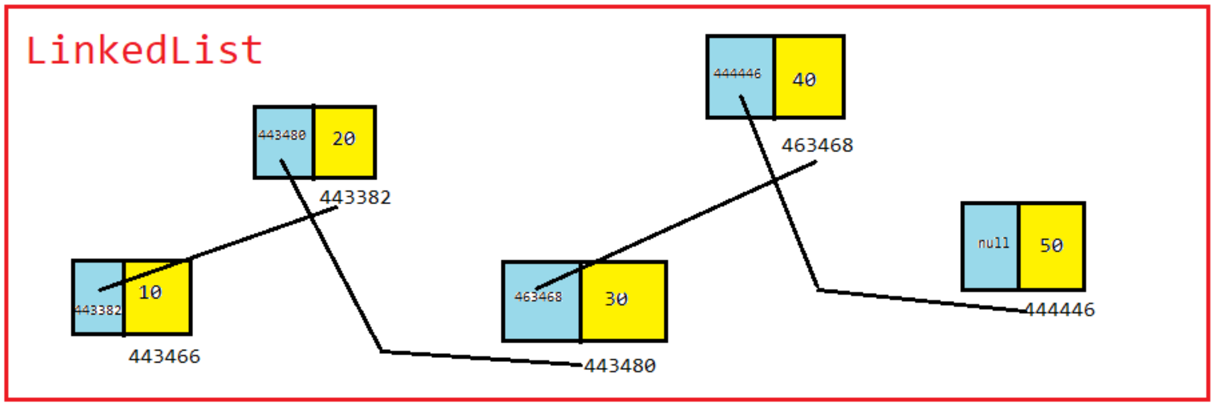
Ex.ArrayList is best suitable for data sorting

When it comes to DataManipulation ArrayList is less Efficient

LinkedList

- To Manipulate data we can use LinkedList
- It stores data in non contiguous memory location
- It occupies more memory





Note: In collection Framework All collections are not synchronized except vector

Task

WAP to copy all data of ArrayList to LinkedList

```
package collection;
```

```java
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedList;
public class Launch {

	public static void main(String[] args) {

		ArrayList<Object> al = new ArrayList<>();
		al.add("raju");
		al.add(12);
		al.add(false);
		al.add('e');
		al.add(12.5f);

		LinkedList<Object> l = new LinkedList<>();

		for(int i=0;i<al.size();i++)
		{
			l.add(al.get(i));
		}

		Iterator<Object> itr = l.iterator();
		while(itr.hasNext())
		{
			System.out.println(itr.next());
		}

	}
}
```

TASK

Launch

```java
package collection;
import java.util.LinkedList;
public class Launch {

	public static void main(String[] args) {
	Account acc = new Account(10, "viman", "qwerty");
	Employee emp1 = new Employee(111, "raju", "yerwada", 2000, acc);
	//emp2 should not accept account of emp1
	Employee emp2 = new Employee(111, "raju", "yerwada", 2000, acc);
```

```
            LinkedList<Employee> l = new LinkedList<Employee>();
            l.add(emp1);
            l.add(emp2);

            for(Employee e:l)
            {
                  e.getData();
                  System.out.println();
            }
      }
}
```

Employee

```
package collection;
public class Employee {

      private final int eid;
      private String ename;
      private String eaddress;
      private int esalary;
      private Account acc;

public Employee(int eid, String ename, String eaddress, int esalary,
Account acc) {
            super();
            this.eid = eid;
            this.ename = ename;
            this.eaddress = eaddress;
            this.esalary = esalary;
            this.acc = acc;
      }
      public void getData()
      {
            System.out.println("Employee ID: "+this.eid);
            System.out.println("Employee Name: "+this.ename);
            System.out.println("Employee Address: "+this.eaddress);
            System.out.println("Employee Salary: "+this.esalary);

            System.out.println("Account Number: "+acc.getAn());
            System.out.println("Account Number: "+acc.getBranch());
            System.out.println("Account Number: "+acc.getIfsc());
      }
}
```

Account

```
package collection;
public class Account {

	private final int an;
	private String branch;
	private String ifsc;
	public Account(int an, String branch, String ifsc) {
		super();
		this.an = an;
		this.branch = branch;
		this.ifsc = ifsc;
	}
	public String getBranch() {
		return branch;
	}
	public void setBranch(String branch) {
		this.branch = branch;
	}
	public String getIfsc() {
		return ifsc;
	}
	public void setIfsc(String ifsc) {
		this.ifsc = ifsc;
	}
	public int getAn() {
		return an;
	}
}
```

Solution:

```
package collection;
import java.util.LinkedList;
import java.util.Scanner;
public class Launch {

	private static LinkedList<Employee> l=new LinkedList<Employee>();
	private static Scanner s=new Scanner(System.in);
	private static String result;

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

```
            System.out.println("****Enter Employee info****\n");
            int count=0;
            while (true)
            {

             System.out.print("Enter employee id: ");
             int eid = s.nextInt();


             System.out.print("Enter employee name: ");
             String ename = s.next();

             System.out.print("Enter employee address: ");
             String eaddress = s.next();

             System.out.print("Enter employee salary: ");
             int esalary = s.nextInt();

             System.out.print("Enter employee account number: ");
             int an = s.nextInt();

             System.out.print("Enter employee branch: ");
             String branch = s.next();

             System.out.print("Enter employee ifsc: ");
             String ifsc = s.next();

             String check = check(eid,an);
             if(count!=0)
             {
                  if(check.endsWith("exist"))
                 {
                        System.out.println("Record Already
Exist:"+check);
                        break;
                 }
             }
             count++;
             Account acc = new Account(an, branch, ifsc);
             Employee emp = new Employee(eid, ename, eaddress, esalary,
acc);

             l.add(emp);
             System.out.println("Do you want to continue:Y/N");
             char c = s.next().charAt(0);
```

```
                if(c=='n' || c=='N')break;

            }
        }

        private static String check(int eid, int an) {

            for(int i=0;i<l.size();i++)
            {
                Employee emp = l.get(i);
                Account acc = emp.getAcc();

                int geteid = emp.getEid();
                int getan = acc.getAn();

                if(eid==geteid)
                {
                    result="id exist";
                }
                else if(an==getan)
                {
                    result="an exist";
                }
                else
                {
                    result="not exist!!!";
                }
            }
            return result;
        }
    }
```

Vector

- Vector is the only collection which is synchronized

Task

Arraylist[raju kaju ajay vijay mohan sohan emma jonny]

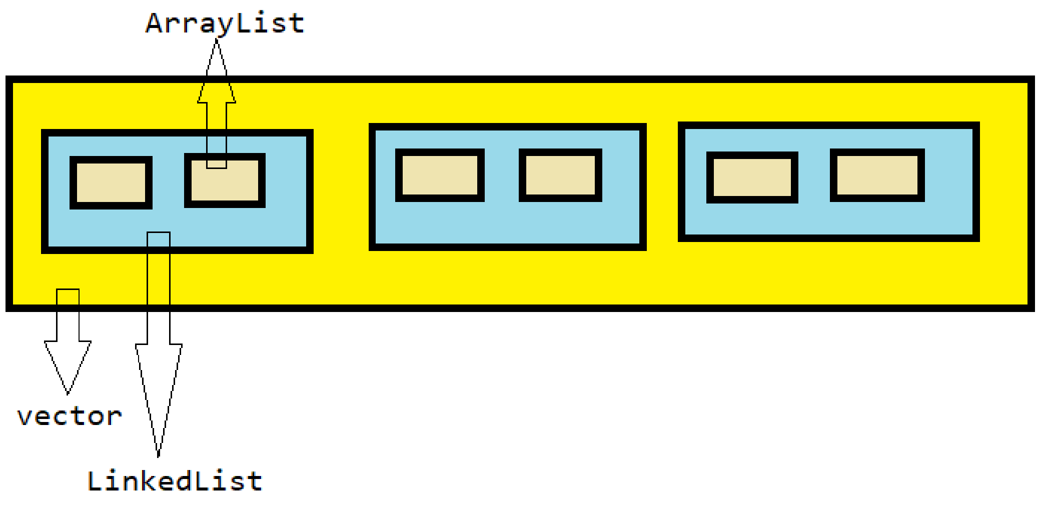
Linkedlist[sheetal priya kiran pitan tamba loha teena plastic]

Vector[starts with vowels]

```
package collection1;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Vector;
public class Launch {

	public static void main(String[] args) {

		ArrayList<String> al = new ArrayList<String>();
		al.add("raju");
		al.add("ayaj");
		al.add("german");

		LinkedList<String> ll = new LinkedList<String>();
		ll.add("Kaar");
		ll.add("omkar");

		Vector<String> v = new Vector<String>();

		for(int i=0;i<al.size();i++)
		{
			String s = al.get(i);
```

```
                        if(s.startsWith("a")|| s.startsWith("e") ||
s.startsWith("i")
                                        || s.startsWith("o") ||
s.startsWith("u"))
                        {
                                v.add(s);
                        }
                }

                for(int i=0;i<ll.size();i++)
                {
                        String s = ll.get(i);
                        if(s.startsWith("a")|| s.startsWith("e") ||
s.startsWith("i")
                                        || s.startsWith("o") ||
s.startsWith("u"))
                        {
                                v.add(s);
                        }
                }

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

Task: Vector of LinkedList of ArrayList



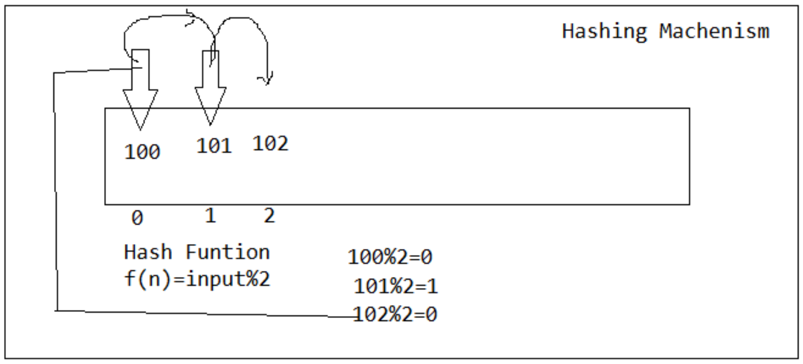
```
new Vector<LinkedList<ArrayList<Object>>>();
```

STACK

```
package collection1;
import java.util.Stack;
public class Launch {

    public static void main(String[] args) {

        Stack<Object> s = new Stack<>();
        s.push("raju");
        s.push(12);
        s.push(12.78f);
        s.push(false);
        s.push('@');
        System.out.println(s);

        System.out.println(s.pop());//remove element

        System.out.println(s);

        System.out.println(s.peek());//show top
element

        System.out.println(s);

           System.out.println(s.pop());//remove
element

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

HashSet

- It uses hashing machenism internally



- Insertion order not preserved
- Duplicates not allowed(Not throw any exception)
- Best suited for unique data
- It can be generic or non generic
- Index wise insertion or deletion not allowed

```
package collection1;
import java.util.HashSet;
import java.util.Vector;
public class Launch {

	public static void main(String[] args) {

Vector<Object> v = new Vector<>();//same as arrayList but synchronized
		v.add("abcd");
		v.add("pqrs");

		HashSet<Object> h = new HashSet<>();
		h.add("raju");
		h.add("kaju");
```

```
            h.add("mohan");
            h.add("shaktiman");
            h.add("yamrajkilwish");
            h.add("rohit");
            h.add("a");
            h.add("b");
            h.add("c");

            h.addAll(v);

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

LinkedHashSet

->same as HashSet but it maintains insertion order

```
package collection1;
import java.util.LinkedHashSet;
import java.util.Vector;

public class Launch {

      public static void main(String[] args) {

Vector<Object> v = new Vector<>();//same as arrayList but synchronized
            v.add("abcd");
            v.add("pqrs");

            LinkedHashSet<Object> h = new LinkedHashSet<>();
            h.add("raju");
            h.add("kaju");
            h.add("mohan");
            h.add("shaktiman");
            h.add("yamrajkilwish");
            h.add("rohit");
            h.add("a");
            h.add("b");
            h.add("b");

            h.addAll(v);
            System.out.println(h);
      }
}
```

```
[raju, kaju, mohan, shaktiman, yamrajkilwish, rohit, a, b, abcd, pqrs]
```

Task

String s=”abcdefijdgcterherjgsefhlesfgeygfeawfuyvbery”;