DATE
PAGE

```
Class A {
    void ShowA() {
        System. out. Println(" Show A");
    }
}
Class B {
    Void ShowB() {
        System. out. Println(" Show B");
    }
}
Class C {
    void ShowC() {
        System. out. Println(" Show C ");
    }
    Public static void main() {
        A a = new. A();
        a. ShowA();
        B b = new B();
        b. ShowB();
        C c = new C();
        c. ShowC();
    }
}
```

integer $\left\{\begin{array}{l}\text{byte - 1}\\ \text{short - 2}\\ \text{int - 4}\\ \text{long - 8}\end{array}\right.$ $\left.\begin{array}{l}\text{float - 4}\\ \text{double - 8}\end{array}\right\}$ float point

Char - 2 $\longrightarrow$ as it supports unicode value.

boolean - 1.

```
Class Demo {
  public static void main(String[] args) {
      int x;
      x = 45;
     System.out.println(x);
     System.out.println("X Value is" + x);
  }
}
```

if any of the operand is string then concatin

• error when using x without initialize

• float x = 23.212; $\longrightarrow$ X (auto conversion of double to float not allowed in java).
  float x = 23.212f; $\longrightarrow$ ✓

```
Class Demo {
      Public static void main( String[] args) {
          int x = 45, y = 12, z;
          z = x + y;
          System.out.Println("Res : " + z);
           z = x - y;
          System.out.println("Res : " + z);
          z = x * y;
          System.out.println("Res : " + z);
          z = x / y;
```

System.in.read() → Takes in binary format
Scanner converts it understandable datatype.

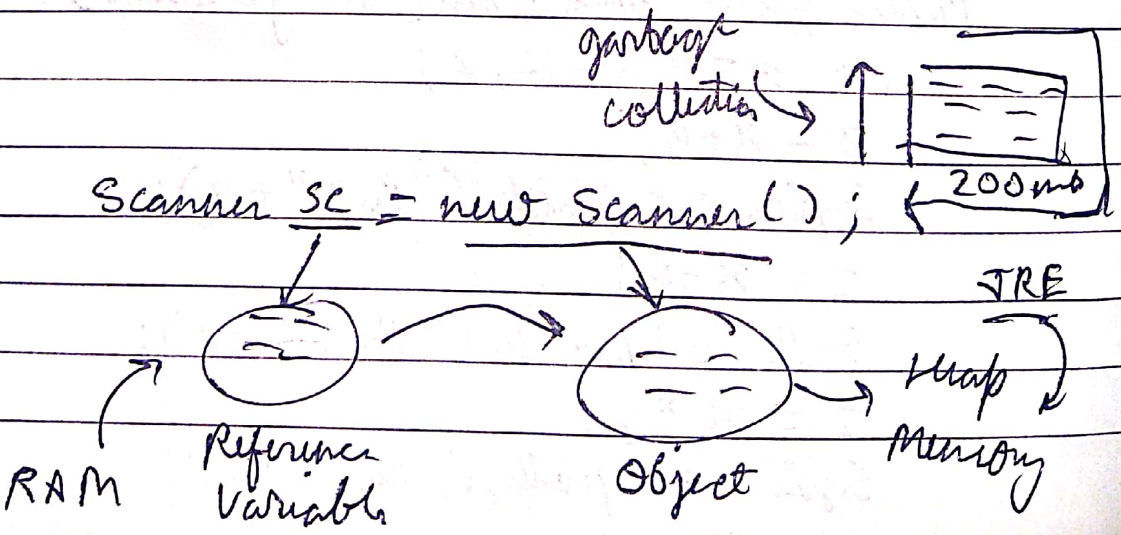
```
        System.out.println("Res:"+z);
        z = x%y;
        System.out.println("Res: "+z);
    }
}
```

class to convert in user understandable form. source

- for user input : Scanner sc = new Scanner(System.in);
- default package : Java.lang → it contains system, string class.
- Scanner class in Java.util package

```
import Java.util.Scanner;
class Demo{
    public static void main(String [] args){
        int x,y,z;
        Scanner sc = new Scanner(System.in);
        System.out.println("Input 2 Numbers:");
        x = sc.nextInt();
        y = sc.nextInt();
        z = x+y;
        system.out.println("Sum: "+z);
    }
}
```

Scanner sc = new Scanner();



Scanner Sc = new Scanner();
(RAM.) Reference Variable ⌐ ↳ Object (In heap)

```
import Java. util. Scanner;
Class Demo {
   public static void main (String [] args) {
      String name;
      int s1, S2, S3, S4, S5, total;
      Scanner sc = new Scanner(System.in);
      system.out.print ("Name:  ");
      name = sc.next();
      System.out.print("physics:  ");
       S1 = sc.nextInt();
      System.out.print ("chemistry: ");
       S2 = sc.nextInt();
      System.out.print ("Maths :  ");
       S3 = sc.nextInt();
      system.out.print (" Social scine:  ");
       S4 = sc.nextInt();
      system.out.print (" Hindi :   ");
       S5 = sc.nextInt();

       total = s1 + s2 + s3 + s4 + s5;

     s.o.Println ("Name:  " + name);
      s.o.pln( "total:  " + total);
   }
}
```

- When main completly executed Refrence variable gets destroyed but objects remain in heap.
- garbage collection : process to free objects without reference variables.

- JRE has limited memory so limited objects can be made.
(not a issue in student life but we can increase JRE size in java setting when we face issue afterwards during developer Job life..

- Type Casting: → up (lower datatype to higher datatype)
  → down (high datatype to lower datatype)

  ↖ temporary conversion of datatype

```
Class Demo {
    public static void main(String[] args){
        String name;
        int s1,s2,s3;
        Scanner sc = new Scanner(System.in);
         s.o.p ("Name: ");
         name = sc.next();
         s.o.p(" Physics: ");
         s1 = sc.nextInt();
         s.o.p(" Chemistry: ");
          s2 = sc.nextInt();
         s.o.p(" maths: ");
          s3 = sc.nextInt();
        int total = s1+s2+s3;
        int size = 3;
        double Per = (double) total/size;        up type cast
        s.o.p(" Percentage: "+ Per);
    }
}
```

Relational operator returns boolean value

- boolean can't be Typecast to int.

- operators:
  - ↳ arithmetic
  - ↳ relational (return boolean in Java, int in C/C++)
  - ↳ increment/decrement
  - ↳ Logical ( )
  - ↳ assignment (compound operators)

- Java don't have (::) scope resolution; (*, &, ->) pointer operators.

- Decision Making Statements:

if, switch

must be a boolean value
int will generate error.

```
if (condition) {
    // true block
}
else {      ⟶   this is optional and require a if block.
    // False block
}
```