If we have a statement like

```
a.somthing()
```

Here ‘a’ is an object and ‘something’ is a function of the class which ‘a’ belongs to.

While writing a program it is a good practice to take care of every error.

Example: look into the following code

```
1.    n = in.nextInt();
2.    sr = Math.sqrt(n);
```

In line 1, if we entered any string rather than integer then it will cause an error. To overcome this error the code is rewritten as below

```
1.    if( in.hasNextInt() )
2.    {
3.          n = in.nextInt();
4.    }
5.    sr = Math.sqrt(n);
```

in the above program if we entered any string then 'n' takes some garbage value to overcome this we need to set some default value. If we entered any negative number then also we will get an error in line5. To overcome these errors we will write the code as below

```
1.    if( in.hasNextInt() )
2.          n = in.nextInt();
3.    else
4.          n = -1;
5.    if( n > 0 )
6.          sr = Math.sqrt(n);
7.    else
8.          System.error.println(" Please enter positive integer only" );
```

There are three types of standard streams

1. standard input
2. standard outuput

and 3. standard error

Generally standard input is connected to Keyboard and standard output and standard error are connected to screen. We can send the output of one stream to input of another stream this is called as I/O Redirection

Example: we can print the number of files in a directory using I/O redirection as below

```
$  ls -l | wc -l
```

in the above statement "ls -l" give the long list of files of the present working directory and that list is given as input to the "wc -l" where "wc" is a word count and "-l" for lines. Now it prints the total number of available files.

Type Casting

Typecast is a unary operator which is used to convert the type of an expression by retaining the value as close as possible

Typecasting is done as below by giving the name of the type in brackets to the expression

```
(double)n
```

Example:

```
d = (double)5;    // now d = 5.0
n = (int)5.0           // now n = 5
n = (int) 5.1          // now n = 5
n = (int) 5.9          // now n = 5
```

Typecasting is not same as round off.

The above four examples shows **explicit typecasting**.

While evaluating any expression the two operands of the operator must be same type. if they are not same then if possible then java implicitly convents them

Otherwise it will show an error.

Example:

int i =10;

double d= 10.5

double sum = i + d;

// now 'i' is converted to double and the result is assigned to 'sum'.

The above example shows the **Implicit Typecasting**

Java can convert any object to String.

Example:

String s = "" + O1 + O2;

the above statement works as

(String s = (("" + O1) + O2)) // first O1 is converted to string and then O2 is converted to Strings

if we have statement like below

String s = O1 + O2

this will give an error because the compiler doesn't know how to add O1 and O2

We can convert the char to String Object by using String constructor

new String(c); // this is not typecasting.

.