escape sequences

```
/*

//

1)    \b    back space
2)    \t    tab
3)    \n    new line
4)    \f    form feed          (new line + tab)
5)    \r    carriage return    (go to the start of current line and print.)
6)    \\    print \
7)    \"    print "
8)    \'    print '

*/


//This is a single line comment.
/*
This is a multi line comment.
This is a multi line comment.
*/

//change words
//ctrl+f (find and replace)

//ctrl+ -  zoom out
//ctrl+ + zoom in
```

datatypes

```
//    datatypes      size     primitive/reference     values

//    boolean        1 bit            p                 true/false

//    byte           1 byte           p              -128/127
//    short          2 bytes          p              -32768/32767
//    int            4 bytes          p              -2 billion/2 billion
//    long           8 bytes          p              -9 quintillion / 9 quintillion        //L
```

```
        //decimal point number.  (3.14)
//      float           4 bytes         p           6-7 fractional points           //F
//      double          8 bytes         p           15 digits fractional points

//      char            2 bytes         p           char -'a'/ascii
//      String          unlimited       r           // number of chars+1

//      primitive                       reference
//      8 bytes or less                 varies
//      quickly(directly)               step wise
//      1 value                         more than 1 or 1
//      less memory                     more memory
//      fast                            slow to execute.
```

Increment / Decrement

```
       // Increment                    Decrement
        // pre  ++x                    pre --x
        // post x++                    post x–
```

Type casting

```
 //Implicit typecasting    char->int->long->float->double
        char a ='A';    //'A'
        int b=a;        //65
        long c=b;       //65
        float d=c;
        double e=d;
```

//explicit typecasting

```
        String str="12";
        int i = Integer.parseInt(str);
        float f = Float.parseFloat(str);
        double dbl = Double.parseDouble(str);
        double dblv = 12345.923;
        int intv= (int)dblv;
        long longv= (long)dblv;
```

Math Class

```
int x=10;
int y=20;
System.out.println("Math.max(x,y) = "+Math.max(x, y));
System.out.println("Math.min(x,y) = "+Math.min(x, y));
int a = -12;
System.out.println("Math.abs(a)="+Math.abs(a));

int b=12;
System.out.println("Math.abs(b)="+Math.abs(b));

int c=27;
double squareRoot = Math.sqrt(c);          //always returns double.  (only for +ve numbers)
System.out.println("Math.sqrt(c)="+squareRoot);

double d=3.14;
double lowerRound = Math.round(d);          //always returns double.
System.out.println("Math.round(d)="+lowerRound);

double e =3.59;
double upperRound = Math.round(e);          //always returns double.
System.out.println("Math.round(e)="+upperRound);

double f =3.59;
double ceil = Math.ceil(f);          //always returns double.
System.out.println("Math.ceil(f)="+ceil);

double g =3.59;
double floor = Math.floor(g);          //always returns double.
System.out.println("Math.floor(g)="+floor);

double base=3;
double altitude=4;
double hypotenous;

hypotenous = Math.sqrt((base*base)+(altitude*altitude));
```

User Input

```
Scanner scanner = new Scanner(System.in);
System.out.print("Enter your name: ");
String name = scanner.nextLine();
```

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

System.out.print("Enter your age: ");
try {
   int age = scanner.nextInt();
   System.out.println("Age: "+age);
} catch (Exception e) {
   //handle exception
   System.out.println("Age must be a number.");
   System.out.println("Exception is: "+e);
   // e.printStackTrace();  //detailed view.
}
finally{
   //this code will always be executed no matter if there is an exception or not.
   scanner.nextLine();       //it will clear the buffer(which has '\n' in it.) for us. And our program will run nicely.
}

System.out.print("Enter your gender: ");
String gender = scanner.nextLine();
System.out.println("Gender: "+gender);

scanner.close();
```

Random Number

```
Random rnd = new Random();

int a=rnd.nextInt();     //-2^32 to 2^32 range
System.out.println("Integer random number: "+a);

int b=rnd.nextInt(6);   //0-5 range
System.out.println("Integer random number less than 6(0-5): "+b);

int c=rnd.nextInt(6)+1;   //1-6 range
System.out.println("Integer random number (1-6): "+c);

double d = rnd.nextDouble(); 0.0-1.0 range
System.out.println("Double random number (0.0-1.0): "+d);

boolean e = rnd.nextBoolean();  true/false
System.out.println("Boolean random: "+e);
```

logical Operators(if else statements)

```
// && (both must be true).
// || (either must be tre).
// !  (reverse of condition).
```

Switch Statement

```
switch (day.toLowerCase()) {
        case "monday":
            System.out.println("Today is monday.");
            break;
        case "tuesday":
            System.out.println("Today is tuesday.");
            break;
        case "wednesday":
            System.out.println("Today is wednesday.");
            break;
        case "thursday":
            System.out.println("Today is thursday.");
            break;
        case "friday":
            System.out.println("Today is friday.");
            break;
        case "saturday":
            System.out.println("Today is saturday.");
            break;
        case "sunday":
            System.out.println("Today is sunday.");
            break;
        default:
            System.out.println("It's not a day.");
            break;
    }
```

While / Do While Loops

```
int i=1;
while(i<=10)
{
    System.out.print(i+"\t");
    i++;
}
```

```
int fact = 4;
int ans = 1;
int ind = fact;
while(ind>=1)
{
   ans*=ind;
   ind--;
}
System.out.println("\nFactorial("+fact+") = "+ans);

int j=1;
do {
   System.out.print(j+"\t");
   j++;
} while (j<=10);

System.out.println();

j=10;
do {
   System.out.print(j+"\t");
   j--;
} while (j>=1);
```

For Loop

```
for (int i = 1; i <= 10; i++) {
   System.out.print(i+"\t");
}

int fact = 4;
int ans = 1;
for (int i = fact; i >= 1; i--) {
   ans*=i;
}
System.out.println("\nFactorial("+fact+") = "+ans);

Int []arr = {1,3,1,2,4,3,2,4,2,3,5,2,3,4,5};
for(int i: arr)   //advanced for loop for iteration in collections.
{
      System.out.print(i);
}
```

```
// for-each  =   traversing technique to iterate through the elements in an array/collection
//                 less steps, more readable
//                 less flexible

      //String[] animals = {"cat","dog","rat","bird"};
      ArrayList<String> animals = new ArrayList<String>();

      animals.add("cat");
      animals.add("dog");
      animals.add("rat");
      animals.add("bird");

      for(String i : animals) {
         System.out.println(i);
      }
```

Nested Loops

```
     for (int i = 1; i <= 4; i++) {
        for (int j = 1; j <= 4; j++) {
           System.out.print("*"+"\t");
        }
        System.out.println("\n");
      }

      System.out.println();

      for (int i = 1; i <= 4; i++) {
        for (int j = 1; j <= i; j++) {
           System.out.print("*"+"\t");
        }
        System.out.println("\n");
      }

      System.out.println();

      for (int i = 4; i >= 1; i--) {
        for (int j = 1; j <= i; j++) {
           System.out.print("*"+"\t");
        }
        System.out.println("\n");
      }
```

Array

```
String []strArr =new String[6];
for (int i = 0; i < strArr.length; i++) {
   strArr[i] = i+1+"";
}

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

2D Array

```
 int [][]myArray = new int[3][4];

for (int i = 0; i < myArray.length; i++) {
   for (int j = 0; j < myArray[i].length; j++) {
      myArray[i][j]=i+j+1;
   }
}

for (int i = 0; i < myArray.length; i++) {
   for (int j = 0; j < myArray[i].length; j++) {
      System.out.print(myArray[i][j]+"\t");
   }
   System.out.println();
}

System.out.println();

int [][]secondArray = {
   {1,2,3},
   {1,2,3,4},
   {3,4,5,6}
};

for (int i = 0; i < secondArray.length; i++) {
   for (int j = 0; j < secondArray[i].length; j++) {
      System.out.print(secondArray[i][j]+"\t");
   }
   System.out.println();
}
```

String Methods

```
String name = "Bro";

if(name.equals("Bro"))
{
   System.out.println("Name: "+name);
}

if(name.equalsIgnoreCase("bro"))
{
   System.out.println("Small name: "+name.toLowerCase());
}

int lnth = name.length();
System.out.println("Length of name: "+lnth);

char ch =name.charAt(0);
System.out.println("First char of name is: "+ch);

int ind = name.indexOf("B");
System.out.println("Index of B in name is: "+ind);

if(!name.isEmpty())
{
   System.out.println("Name is defined!");
}

System.out.println("NAME: "+name.toUpperCase());
System.out.println("name: "+name.toLowerCase());

String secondName = "    bhai    ";
System.out.println("Second name is: "+secondName.trim()+"!");

System.out.println("Replaced name: "+name.replace('B', 'P'));
name = "Bruhh";
System.out.println("Replaced name: "+name.replace("Br", "Pr"));
```

Wrapper Classes(Reference Data Type)

//Wrapper Classes = **Use primitive datatype as a Reference datatype**.
//String, Integer, Double, Character, Boolean are all wrapper classes.
//**It has methods can be used with collections. (like ArrayList and String ....)**

```
// Primitive Datatype          Wrapper Class(Reference datatype)
// boolean                     Boolean
// char                        Character
// int                         Integer
//double                       Double
//                            String

//Disadvantage of wrapper classes      Advantages of wrapper classes
//It's slow to execute                 Useful methods
```

//Autoboxing = Automatic conversion from primitive datatype to wrapper class type(Reference datatype).

```
Boolean a = true;
Character b= 'A';
Integer c = 123;
Double d = 3.00;
String str = "Hassan Raza";
// a.hashCode();
```
//now can use useful functions of wrapper classes.

//unboxing = Reverse of Autoboxing (Automatic conversion from wrapper class type(Reference datatype) to primitive datatype.)

```
boolean e = a;
char f = b;
int g = c;
double h = d;
```

ArrayList

//**ArrayList (Resizeable array or dynamic array.)**

//declaration

```
ArrayList<String> foodList = new ArrayList<String>();       //only for String values.
// ArrayList<Integer> numList = new ArrayList<Integer>();    //only for int values.
// ArrayList<Object> objList = new ArrayList<Object>();
//Above one is for all types(int,double,String....) of data.
```

//Insertion

```
foodList.add("Pizza");
foodList.add("Burger");
foodList.add("Shawarma");
foodList.add("Sandwich");
```

```
//updation
foodList.set(2, "Chicken Tikka");

//deletion
foodList.remove(1);

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

//clean list (delete all values from ArrayList)
foodList.clear();

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

Output:

```
[Pizza, Chicken Tikka, Sandwich]
Pizza
Chicken Tikka
Sandwich
[]
```

2D ArrayList

```
ArrayList<String> bakeryList = new ArrayList<String>();
bakeryList.add("Cookies");
bakeryList.add("Morgan Rusk");
bakeryList.add("Gourmet Bun");

System.out.println("Bakery list: "+bakeryList+"\n");

ArrayList<String> drinksList = new ArrayList<String>();
drinksList.add("Pepsi");
drinksList.add("Coca Cola");
drinksList.add("Bold Sprite");
drinksList.add("Sting");

System.out.println("Drinks List: "+drinksList+"\n");

ArrayList<ArrayList<String>> groceryList = new ArrayList<ArrayList<String>>();
groceryList.add(bakeryList);
```

```
    groceryList.add(drinksList);

    System.out.println("Grocery list: "+groceryList+"\n");
    System.out.println("Grocery list: ");

    for (int i = 0; i < groceryList.size() ; i++)
    {
        for (int j = 0; j < groceryList.get(i).size(); j++)
        {
            System.out.print(groceryList.get(i).get(j)+"\t");
        }
        System.out.println();
    }
```

Methods

```
public static void main(String[] args) {
    // methods = block of code that is executed when ever it is called.
    hello();
    hello("Bro",22);

    int x=3;
    int y=4;

    int sum = sum(x, y);
    System.out.println("Sum of "+x+" and "+y+" is "+sum+".");
}

public static void hello()
{
    System.out.println("Hello world!");
}

public static void hello(String name, int age)
{
    System.out.println("Hello "+name+". Your age is "+age);
}

static int sum(int x,int y)
{
    return x+y;
}
```

overloaded methods

//overloaded methods = methods that have the same name but have different parameters.
// method name + parameters = function signature.

```
System.out.println(add(1, 2.0, 3));
static int add(int a, int b)
{
   return a+b;
}
static int add(int a, double b, int c)
{
   return a+b+c;
}
```

printf() (Format Specifier)

//printf = an optional method to control, format and display text on the console window.
//two arguments = format string + (object/variable/value)
//for it we use format specifier:.
// **% [flag] [precision] [width] [conversion-character]**

// format specifiers [conversion-characters]
// **%d** (int)decimal number
// **%b** boolean
// **%c** char
// **%s** String
// **%f** double

```
//[conversion-character]
      System.out.printf("My name is %s","Hassan Raza");
      System.out.printf("\nMy age is %d",21);
      System.out.printf("\nMy gender is %c",'M');
      System.out.printf("\nI am a %b Pakistani",true);
      System.out.printf("\nMy heights is %f feets",8.67);


//[width]     (define the width in which your data should display.)
      //          // minimum number of characters to be written as output
      //%16s       //defines from right to left and is shifted towards right if data is small.
      //%-16s      //defines from left to right and is shifted towards left if data is small and
remaining space will be empty.
      System.out.printf("\n\nMy name is %16s","Hassan Raza");
```

```
        //[precision]
        // sets number of digits of precision when outputting floating-point values
        System.out.printf("\n\nPi is %.3f",3.14159);       //rounded to three precision points.

        // [flags]
        // adds an effect to output based on the flag added to format specifier
        // - : left-justify
        // + : output a plus ( + ) or minus ( - ) sign for a numeric value
        // 0 : numeric values are zero-padded
        // , : comma grouping separator if numbers >= 1000

        System.out.printf("\n\nPrice of torch is %-15f",10000.0);   //second arg must be double for
%f
        System.out.printf("\nPrice of torch is %+f",10000.000);     //shows + sign with value
        System.out.printf("\nPrice of torch is %+f", -10000.000);           //shows - sign with value
        System.out.printf("\nPrice of torch is %015f",10000.000);
        System.out.printf("\nPrice of torch is %,15f",-10000.000);          //commas added

        System.out.printf("\n\nPrice of torch is %,-+15f",-10000.000);       //, / left-align / negative
sign everything there.
```

Final (constant variable)

```
//it's same like const in other languages.
final double PI = 3.14159;

//PI = 4;  //You can't change a final variable

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