Primitives Data types

Group	Type	Size	Minimum value	Maximum value
Integer	byte	1 Byte	-128	127
	short	2 Bytes	-32768	32767
	int	4 Bytes	-2147483648	2147483647
	long	8 Bytes	-9223372036854775808	9223372036854775807
Floating	float	4 Bytes	1.4 E-45	3.402,823,5 E+38
	double	8 Bytes	4.9 E-324	1.7976931348623157 E+308
Others	char	2 Bytes	0	65535
	boolean	1 Bit	true or false	

String to store group of characters, for example “Programming”, “Java”, “First program”, “12345”, “-+-03,cd”

Arithmetic operators

Operator (العملية)	Operands (الحدود)	Example
+ Additive	2	5 + 2
- Subtraction	2	5 – 2
* Multiplication	2	5 * 2
/ Division	2	5 / 2
% Remainder	2	10 % 2
= Assignment	2	x = 10
+ Unary plus	1	+x
- Unary minus	1	-x
++ Increment	1	x++ or ++x
– Decrement	1	x-- or –x
+= Addition assignment	2	x += 2
-= Subtractions assignment	2	x -= 2
*= Multiplication assignment	2	x *= 2
/= Division assignment	2	x /= 2
%= Reminder assignment	2	x %= 2

Operator Precedence

Level	Operators	Associativity
1	() + (unary plus) - (unary minus)	Left to right
2	* / %	Left to right
3	+ -	Left to right
4	= += -= *= /= %=	Right o left

Rules for naming identifier

- **Keywords not allowed.**
- **Only letters (a-z A-Z), digits 0-9, _ (under score), $**
- **Not allowed to start with digits.**
- **Duplication not allowed.**
- **Do not use special characters (. - * \ / @ etc.) only (_ and $)**
- **It's better to use small letters (a-z), if the identifier has more then one word start the first word with small and the rest with capitol for example: totalNumbers or BestAverageInTheClass.**
- **Usually class name starts with capitol for examples Student, Teacher.**

Examples:

int number1; ✓
int 1number; ✗ you can not start with a digit
int total numbers; ✗ you can not use space
int total_numbers; ✓

Arithmetic Expression evaluations

When java evaluates the arithmetic expression, it will divided into operators and evaluate every operator alone, it will start with the height precedence.

Example:

Example 1

x = 4 + 3 * 6 – 9 / 3 we have * and / same level, you go left to right

1)- 3 * 6 = 18
x = 4 + 18 – 9 / 3

2)- 9 / 3 = 3
x = 4 + 18 – 3 we have + and - same level, you go left to right

3)- 4 + 18 = 22
x = 22 – 3

4)- x = 19

Example 2

x = (4 + 3) * 6 – 9 / 3 we have () , you must start with it.

1)- (4 + 3) = 7
x = 7 * 6 – 9 / 3

2)- 7 * 6 = 42 we have * and / same level, you go left to right
x = 42 – 9 / 3

3)- 9/3 = 3
x = 42 – 3

4- x = 39

Example 3

x = y = z = 3 **we have = repeated 3 times, you go right to left**

1)- z = 3
x = y = 3

2)- y = 3
x = 3

at the end we have x = 3, y = 3, z = 3

Note about the assignment operator (=)

The assignment operator has 2 operands, the left operand must be identifier not a constant or equation.

Example:

x = 10; ✓
x = y * 3; ✓
10 = x; ✗
4 = x * 3; ✗

Working with ++ and --

What happens when you have x++;

This post increment and it is the only operator in the expression so it will run the same as ++x;

But what will happen when you have

y = x++; (post increment)

suppose x = 4;

then it will do the following:

1)- y = x;
y becomes 4

2)- x++
x becomes 5

finally x = 5 and y = 4

What will happen when you have

y = ++x; (pre increment)

suppose x = 4;

then it will do the following:

1)- ++x
x becomes 5

1)- y = x;
y becomes 5

finally x = 5 and y = 5

Note that a both cases will be 5 but y first case 4, second case 5, so when you change the value of x it effects y.

Example for ++ and --

Suppose t = 4, f = 2, d = 10, y = 3

x = y * t++ - 4 * ++f / 2 + --d

1)- ++f

f = f + 1

f = 2 + 1 = 3

x = y * t++ - 4 * 3 / 2 + --d

2)- --d

d = d - 1

d = 10 - 1 = 9

x = y * t++ - 4 * 3 / 2 + 9

3)- y * t = 3 * 4 = 12 it will do ++ with t at the end

x = 12 - 4 * 3 / 2 + 9

4)- 4 * 3 = 12

x = 12 - 12 / 2 + 9

5)- 12 / 1 = 12

x = 12 – 6 + 9

6)- 12 – 6 = 0

x = 3 + 9

7)- x = 15

8)- Now you add 1 to t (t++), t will be 5

9)- Final values : x = 15 , y = 3, t = 5, d = 9, f = 3

Example about casting:

Suppose we have the following

int x = 7;
int y = 2;
int z;

z = x/y;

(int / int = int, it will truncate the decimal value, the actual value is 3.5, but it will make it 3 with out rounding.

z = 7/2 = 3

So z will have the value of 3.

suppose we have:
double r =7.0;
double s = 2.0;
double t;

t = x / y;
x / y = 3;
It will store the value as double 3.0 because t is double.
t == 3.0

t = r/s;
double / double = double
7.0 / 2.0 = 3.5;
t = 3.0

z = r/s;
This is will give an error. You can not store double value into int variable.

If we have two different types, then the result will be double
t = x/s
7/2.0 = 3.5

be careful if you do the following it will give compile error:
z = x/s; because you can not store double into int varaible.

Explicit casting: تحويل صريح

Suppose you want to have double as a result when you do x/y (7/2), you can do casting as following:
t = (double) x / y;
or
t = x / (double) y;

if you do t = (double) (x/y);
You got 3.0, because it will calculate first x/y (int/int) you got 3, then it will convert it to 3.0

You can cast from double to int, but you have to be careful.

How to write a program in java:

```
public class Ex1
{
        public static void main(String args[]) or [] args
        {


        }
}
```

We call { } block. You can have block as many as you need.

Comments:

The comments are to explain what you are doing in your program.
There are two type of comments:
1- Multiple lines comments, starts with /* and ends with */
you can put the comments any where in the program.

Example:

```
/*
        Name: aaaa
        ID : 12345
        HW: 1
*/
```

2- One line comment, starts with // and ends with enter.

Example :

```
int name; // this is the name of the person
```

Errors types:

- **compile or syntax:** **With this error the program will not execute.**
  Examples:

  ```
  int x, error, you should put ; not ,

  int total;
  Total = 10; error Total not defined

  int x,a,b;
  x = a > b; error x is int not boolean

  int x;
  if (x == 0) error x has no value
      System.out.println("YES");
  ```

- **Run time error:** **This error happens when executing the program.**
  Example:

  ```
  int x = 0;
  System.out.println(1/x); when executing it will give message can not divide by 0

  int x = -10;
  System.out.println(Math.sqrt(x)); error negative square root not defined
  ```

- **Logic error:** **This error (the worst type of errors), no compile error, the program will execute but with wrong results.**

Output:

System.out.print(); prints and stay in the same line.
System.out.println(); prints and goes to next line.

Printing message:

System.out.println("Java is good language");
It will prints any thing inside the "....", except if we have \ (back slash) with an other character.
\n new line
\t tab
\" "
**\\ **

Examples:

System.out.println("Java is \ngood language");
prints:
Java is
good language

System.out.println("Java is \n\n\ngood language");
\n\n\n means three new line
prints:
Java is

good language

System.out.println("Java is \tgood language");
prints:
Java is good language

System.out.println("My name is \"Khaled\"");
prints:
My name is "Khaled"

System.out.println("\\\\\\");
prints:
\\\ each two back slashes (\\) will print one back slash (\)

Printing variables:

name = “Khaled”
System.out.println(name);
prints:
Khaled

---

id = 412345678
System.out.println(id);
prints:
412345678

Printing variables with messages:
System.out.println(“My name is “ + name); **+ means print the message and the value of name**

prints:
My name is Khaled

int x = 50;
System.out.println(“x = “ + x + 1);
prints:
x = 501
it will not add 1 to x, it will print the value of x and after that 1, because x is not operator, it means print the message then x then value 1.

int x = 50;
System.out.println(“x = “ + (x + 1));
prints:
x = 51
Here it will add 1 to x, because we have ().

Be careful:
int x = 50;
System.out.println(x + 1 + “= x“);
prints:
51 = x

Input

Scanner:

To read we have to use the Scanner, first out before the class definition the following:

```
import java.util.Scanner;
```

then inside the main block you put the following:

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

You can out any name for the Scanner, here we call it input.

Using Scanner:

```
byte hours = input.nextByte();
short salary = input.nextShort();
int id = input.nextInt();
long size = input.nextLong();
float avg = input.nextFloat();
double gpa = input.nextDouble();
boolean isMarried = input.nextBoolean();
String name = input.next(); or input.nextLine();
char grade = input.nextLine().charAt(0);
```

Relational operator:

Operator	Explain
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
==	Equality
!=	Not equal

Example:

```
int x = 10;
int y = 3;
boolean b;

b = x > y;
b = true

b = x == y;
b = false

b = x != y;
b = true

b = x + y; Error x+y is arithmetic expression, can not store in boolean
```

Logical Operators:

Operator	Explain
&&	and
||	or
!	not
&	Bitwise AND
|	Bitwise or

true && true	true
true && false	false
false && true	false
false && false	false

true || true	true
true || false	true
false || true	true
false || false	false

! true	false
! false	true

```
int x = 10;
int y = 2;
int z = 0;

boolean b;

b = x > y && z != 0;
b = true && false
b = false
```

using &&:
when the first condition is true the second will not be evaluate it.

```
b = x < y && z++ != 0;
b = false
```

it will not check z++ != 0 and z will stay 0
if use or (||), if first condition is true then the second condition will not be checked.

Both & or | will check the 2 operands.

if statement

```
if (condition)
        statement;
```

or
use block {} if you have more then one statement.

```
if (condition)
{
        statement1;
        statement2;
        statement3;


        statementn;
}
```

```
if (condition)
        statement;
else
        statement;
```

```
if (condition)
        statement;
else if (condition)
        statement;
else if (condition)
        statement;
.
.
.
else
        statement;
```

else is not must.

Examples

```
if (y != 0)
{
        z = x/y;
        System.out.pritnln("Z = " + z);
}
else
        System.out.pritnln("Can not divide by 0");

if (maek >= 90)
        System.out.pritnln("Your grade is A");
else if (maek >= 80)
        System.out.pritnln("Your grade is B");
else if (maek >= 70)
        System.out.pritnln("Your grade is C");
else if (maek >= 60)
        System.out.pritnln("Your grade is D");
else
        System.out.pritnln("Your grade is F");
```

be careful

```
int x = 1;

if(x == 0);
        System.out.println("OK");
```

prints:
OK

because we have ; as empty statement after if ( ), the print message will be execute always, it is not in the if control.

boolean b = true;

if (b)
System.out.println(“OK”);
else
System.out.println(“NOT OK”);

prints:
OK

Since b is true, the if condition will be evaluated to true.

---

boolean b = true;

if (! b)
System.out.println(“OK”);
else
System.out.println(“NOT OK”);

prints:
NOT OK

Since b is true, the if condition will be evaluated to ! true which is false.

---

if (x = 4)
System.out.println(“OK”);

error because = is not allowed you should use ==

switch/case/default

```
day = 1;

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

Rules for switch:

- **switch will work only with int or char. with new version (Java 7) switch will works with int, String, char, enum.**
- **You can have many case statements.**
- **Duplicate case not allowed.**
- **Order of cases is not required.**
- **default is not required.**
- **When executing switch it will go to the required case, for example if we have day = 2, it will go to case 2 and execute all statement, if there is break in this case it will go out of the switch or it will continue until it founds break or reaches the end of switch.**

for:

for (first value for the counter ; condition to stop ; change the counter)
statement;

if you have more then statement you must put block {}.

```
int i;

for (i = 1 ; i <= 10 ; i++)
        System.out.println("OK");
```

prints OK 10 times

you can put the counter change i++ or ++i

```
int i;

for (i = 1 ; i <= 3 ; i++)
        System.out.println("OK");

System.out.println("I = " + i);
```

prints
OK
OK
OK
4

```
for (int i = 1 ; i <= 3 ; i++)
        System.out.println("OK");

System.out.println("I = " + i);
```

error in the print statement, i is not defined
i is defined inside for statement, you can use it only inside for.

```
int i;

for (i = 1 ; i <= 10 ; i++);
        System.out.println(i);
```

print
11

it prints 11, because there is ; empty statement after the for.

```
int i = 1;

for (  ; i <= 3 ; i++)
        System.out.println("OK");
```

prints:
OK
OK
OK

you can leave the first statement empty, but you have to give the counter value before the condition.

```
int i = 1;

for (  ; i <= 3 ; )
{
        System.out.println("OK");
        i++;
}
```

prints:
OK
OK
OK

you can leave the first statement empty, but you have to give the counter value before the condition.

Print even numbers from 1 to 100

bad design:

```
for  (int i = 1 ; i <= 100 ; i++)
{
        if (i % 2 == 0)
                System.out.println(i);
}
```

good design

```
for  (int i = 2 ; i <= 100 ; i = i + 2) or i += 2
        System.out.println(i);
```

---

read n (number of marks) calculate the average and print it, print the maximum value.

```
Scanner input = new Scanner(System.in);
int n , total = 0;
int mark;
String name;
int max = 0;
String MaxName = “”;

System.out.print(“Enter how many numbers: “);
n = input.nextInt();

for (int i = 1 ; i <= n ; i++)
{
        System.out.print(“Enter mark: “);
        mark = input.nextInt();

        System.out.print(“Enter name: “);
        name = input.nextLine();

        total = total + mark; // or total += mark

        if (mark > max)
        {
                max = mark;
                maxName = name;
        }
}

if (n == 0)
        System.out.pritnln(“No marks”);
else
{
        double avg = total / n;
        System.out.pritnln(“Average : ” + avg);

        System.out.pritnln(“Maximum mark : ” + max);
        System.out.pritnln(“Maximum mark name : ” + maxName);
}
```

while

read marks calculate the average and print it, print the maximum value.
Enter negative mark or 0 to finish.

```
Scanner input = new Scanner(System.in);
int mark,counter = 0, total = 0;
String name;
int max = 0;
String MaxName = “”;

System.out.print(“Enter mark: “);
mark = input.nextInt();

while(mark > 0)
{
        System.out.print(“Enter name: “);
        name = input.nextLine();

        total = total + mark; // or total += mark
        counter++;

        if (mark > max)
        {
                max = mark;
                maxName = name;
        }

        System.out.print(“Enter mark: “);
        mark = input.nextInt();
}

if (counter == 0)
        System.out.pritnln(“No marks”);
else
{
        double avg = total / counter;
        System.out.pritnln(“Average : ” + avg);

        System.out.pritnln(“Maximum mark : ” + max);
        System.out.pritnln(“Maximum mark name : ” + maxName);
}
```

do/while

read marks calculate the average and print it, print the maximum value.
Enter negative mark or 0 to finish.

```
Scanner input = new Scanner(System.in);
int mark,counter = 0,total = 0;
String name;
int max = 0;
String MaxName = “”;

do
{
        System.out.print(“Enter mark: “);
        mark = input.nextInt();

        if (mark > 0)
        {
                System.out.print(“Enter name: “);
                name = input.nextLine();

                total = total + mark; // or total += mark
                counter++;

                if (mark > max)
                {
                        max = mark;
                        maxName = name;
                }
        }
}while(mark > 0);

if (counter == 0)
        System.out.pritnln(“No marks”);
else
{
        double avg = total / counter;
        System.out.pritnln(“Average : ” + avg);

        System.out.pritnln(“Maximum mark : ” + max);
        System.out.pritnln(“Maximum mark name : ” + maxName);
}
```