* Notes:

- SOP(« _____ ») ✗ SOP(« _____ » + « _____ ») ✓

- SOP (Hello) ✗ compilation error (no quotations)

- Escape sequences:

\t	tab
\n	new line
\" _ \"	quotations
\\	backslash

- System.out.println («_____»);

object — method name — parameters

int total;

data type — variable name

Java is case sensitive
Total ≠ total ≠ TOTAL

- constants are like variables except that it holds the same value during its entire existance
the compiler produces an error message if you attempt to change its value
Why to use constants:
1) giving a constant a value a name helps explains its role
2) compiler protects constant values (prevent changing constant value avoid inadvertent errors)
3) They facilitate program maintenance

we have 8 primitive types: byte (8), short (16), int (32), long (64) (whole numbers)
float (32), double (64) (decimals)
char
boolean

- in float we have 7 significant digits
in double we have 15 significant digits

- char stores a single character
char c = 'A' ; '$'
char c = 'ab' (compilation error)

- boolean variables have only two valid values true and false

- unary operations (-1, -4, +5)

- SOP (-4 % -5) ; → -4

a = 10
int d = 5.6 * a
the answer will be 56 but you will get compilation error because of the 5.6 at first so you should put double

double d = 5.6 * a

if a = 10
int d = a * 5.6 you get compilation error or not

∴ Int a = 10, b = 4;
double e = a * b → 40.0

- SOP (4.5 + 2) → 6.5
SOP (109 / 10); → 10 (logical error because you expect to get 10.9)

to correct this error
SOP (109.0 / 10); → 10.9
read as double

- SOP (-5 * 10 + 6); → -44

Highest priority

1	+	Unary plus
	−	Unary minus
2	*	multiplication
	/	division
	%	remainder
3	+	Addition
	−	subtraction
4	=	assignment

lowest priority

- precedence can be forced by using parentheses (14+8)/2
- Arithmetic operators with the same precedence are evaluated from left to right.

- answer = Sum /4 + Max * lowest

  $\boxed{4}$ $\boxed{1}$ $\boxed{3}$ $\boxed{2}$

- Increment and Decrement

a = a+1	a = a −1	
a++	a−−	postfix form
++a	−−a	prefix form

- Deference between ++y y++

```
int y=5;
int t= y++;
```

means store the value of y in t and then increment)

```
SOP(t); → 5
SOP(y); → 6
t= ++y
SOP(t) → 7
SOP(y) → 7
```

Assignment operators:

int a

a+ = 5 ; $\Longleftrightarrow$ a = a + 5

a- = 5 ; $\Longleftrightarrow$ a = a - 5

a* = 6 ; $\Longleftrightarrow$ a = a * 6

a /= 7 ; $\Longleftrightarrow$ a = a / 7

a - = b + c $\Longleftrightarrow$ a = a - (b+c)

- byte can be changed to short, int, long, float, double (no loss)

  short < int < long < float < double

- narrowing leads to loss of information

- conversion techniques

  1) Assignment conversion (only widening)

     it occurs when a value of one type is assigned to a variable of another type

     money is float variable and dollars is int variable

     money = dollars $\Rightarrow$ dollars is converted to float

  2) by promotion:

     it occurs automatically to convert their operands to perform operations

     if sum is float and count is integer

     result = sum/$\underbrace{\text{count}}_{\text{converted to float}}$;

3) Casting (widening and narrowing)
specified by a type name in parentheses placed in front of the value to be converted

```
float money = 84.69;
int dollars;
dollars = (int)money; ⇒ dollars = 84
```

- double s = 6.7;
  int res = (int) s; if you put int res = s compilation error

make sure → SOP(s) = 6.7

- int a = 10;
  int b = 4;
  SOP((double) a/b); → 2.5
  SOP((double)(a/b)); → 2.0
  SOP(1.0 * (a/b)); → 2.0
  SOP(1.0 * a/b); → 2.5

- name = new String("James Bond");
  This calls the string constructor, which is a special method that sets up the object

shortcut String s1 = "Hello"
int l = s1.length() ⇒ 5

- String s1 = "Hello"
  String s2 = "Hi"
  s2 = s1
  s1 = "world";
  SOP(s1); → world
  SOP(s2); → world.

  s1 & s2 are called aliases

in case of numbers
num1 = 38
num2 = 96
num2 = num1;
num1 = 38
num2 = 38

- 2 or more references referring to same object are called aliases of eachother
- object with no reference to it becomes garbage
  the object can no longer contribute to the program

- Keyboard input is represented by the System.in object

- answer = scan.nextLine();
  reads the input line and returns it as one string
  scan.nextInt(), scan.nextDouble
  used to read data of particular types

- once string object is created
  its value can not be lengthened or shortened
  nor can any of its characters change
  string objects are immutable

- import java.util.Scanner;
  public class name {
  public static void main (String [] args) {
  Scanner S = new Scanner (System.in)
  String mass = S.nextLine();
  SOP (mass.length());

- int x = S.next double x
  double2 = S.nextInt()

~~Str~~

- String s = "Hi"
  s = "Hello";
  char c;
  c = s.charAt(0);
  SOP(c); → H
  SOP(s.charAt(0)); → H
  int x = 6
  SOP(s.charAt(2*x - 10)); → f
  s.charAt(6) → run time error
- SOP(s.equals(s2)) → capital lower case matter
  SOP(s.equalsignore(s2)) → capital lower case doesn't matter.
- import java.util.*; → any class inside the package util might be used
- A java compiler translates java source code into java bytecode, a low-level architecture-neutral representation of the program.