JAVA

ОСНОВИ СИНТАКСИСУ

СИНТАКСИС

Оператори:

https://uk.wikibooks.org/wiki/%D0%9E%D1%81%D0%B2%D0%BE%D1%8E%D1%94%D0%BC%D0%BE_Java/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8#%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%B8

Типи даних:

https://uk.wikibooks.org/wiki/%D0%9E%D1%81%D0%B2%D0%BE%D1%8E%D1%94%D0%BC%D0%BE_Java/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8#%D0%A2%D0%B8%D0%BF%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85

JAVA JDK. ENTRY POINT

One of the class must have the **`main()`** method as the execution starting point of the program

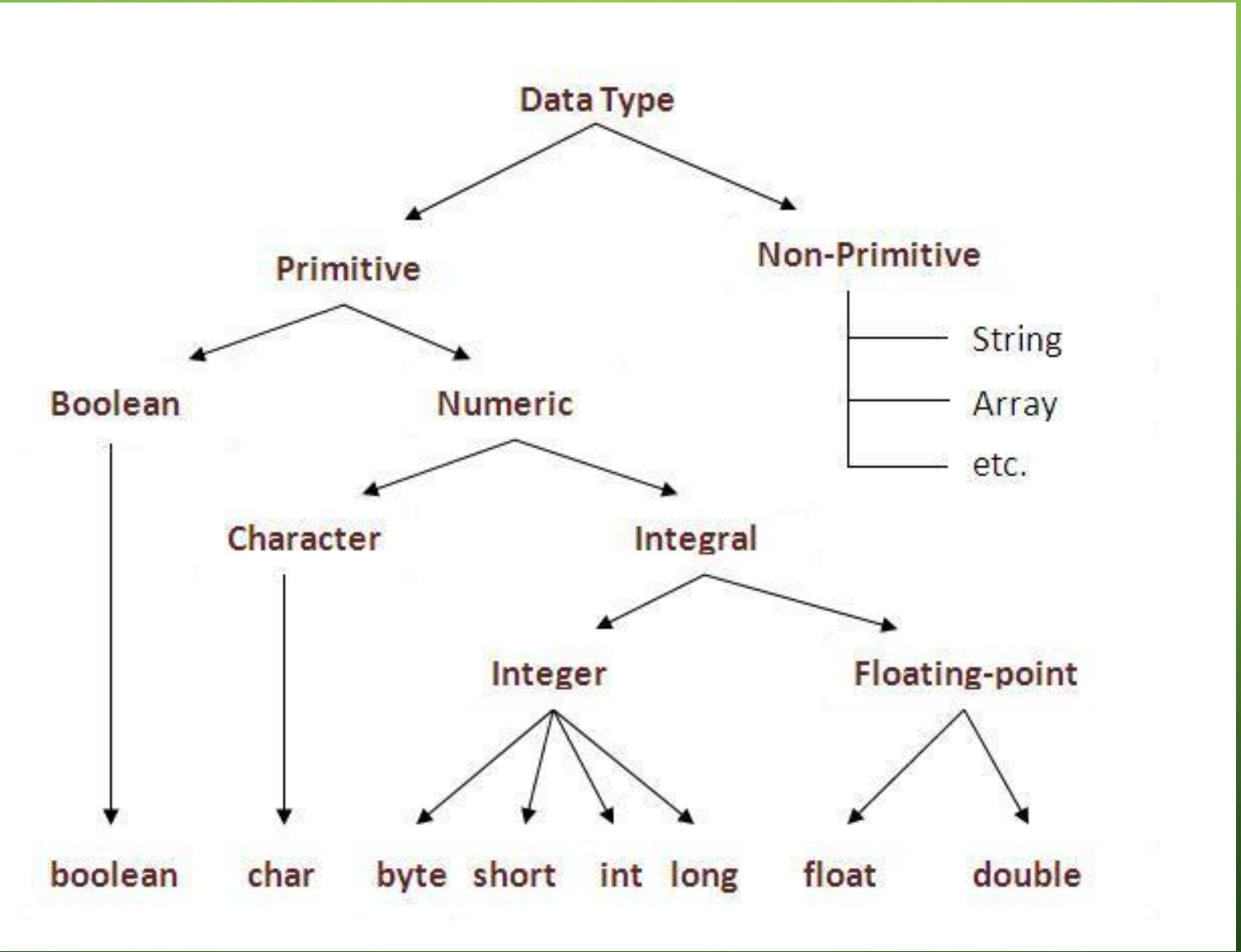
```
public class Example {
    public static void main(String[ ] args)   {
        System.out.println("My first program");
    }
}
```

To display information on the screen it's used

- ✓ **`System.out.print(...)`** – prints the required output on the ***same line*** continuously again and again on the screen;
- ✓ **`System.out.println(...)`** – prints the output in the ***next line*** of the previous result on the screen

```
System.out.println("Text1");
System.out.println("Text 1 " + "Text 2");
```

ТИПИ ДАНИХ JAVA



ПРИМІТИВНІ ТИПИ ДАНИХ

Data Type	Default Value	Default size	Example
boolean	false	1 bit	true, false
char	'\u0000'	2 byte	‘r’
byte	0	1 byte	5
short	0	2 byte	12
int	0	4 byte	-5; 0; 123
long	0L	8 byte	34
float	0.0f	4 byte	5.2
double	0.0d	8 byte	-6.4; 0.0; 123.56

JAVA VARIABLES

Variable – symbolic name associated with a value and whose associated value may be changed.

Declaration – process of variable's specifying. Usually declaration consist of defining type, name and default value of variable.

```
int a, b, c;           // Declares three ints, a, b, and c.
int x = 10, y = 10;    // Example of initialization
byte d = 22;           // initializes a byte type variable d.
double pi = 3.14159; // declares and assigns a value of PI.
char e = 'a';          // the char variable e is initialized with value 'a'
```

A process in which a variable is set to its first value is called initialization.

ЗАПИС ЛІТЕРАЛІВ

```
int a = 0777// вісімковий запис
int b = 0xAA// шістнадцятковий
int c = 0b01101101// з версії 1.7
```

Вивід через

```
System.out.println("byte b = " + Integer.toOctalString(b));
System.out.println("short h = " + Integer.toBinaryString(h));
```

JAVA COMMENTS

The Java language supports three kinds of comments:

- the compiler ignores everything from // to the end of the line.

```
// This example demonstrates the use of single line comments
```

- the compiler ignores everything from /* to */.

```
/* This is a sample class which is used to demonstrate the use
of multi-line comments. This comment does not appear in the
java documentation */
```

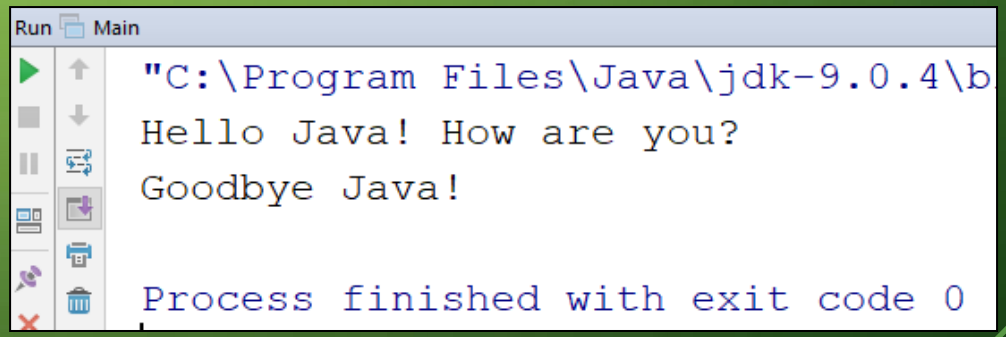
- /** and */ indicates a documentation comment. The compiler ignores this kind of comment, just like it ignores comments that use /* and */. The JDK javadoc tool uses doc comments when preparing automatically generated documentation.

ВИВІД ДАНИХ

- For displaying ***simple text*** messages to the user:
  - ✓ **`System.out.print(...)`** – prints the required output on the ***same line*** continuously again and again on the screen;
  - ✓ **`System.out.println(...)`** – prints the output in the ***next line*** of the previous result on the screen.

  Java language supports few special **escape sequences** for **`String`** and **`char`** literals as well. (\n \r \t \" \' \\)

```
public class Main {
    public static void main(String[] args) {
        System.out.print("Hello ");
        System.out.print("Java!");
        System.out.println(" How are you?");
        System.out.println("Goodbye Java!");
    }
}
```



ЧИСЛОВІ ДАНІ ФОРМАТОВАНИЙ ВИВІД

запис констант через final

```
// нескінченність
double x = 0.3;      double y = 0.0;      double z = x*y;
double inf = x/y;
System.out.printf("%7.3f / %7.3f = %7.3f%n", x,y,inf);
double nanf = z/y;
System.out.printf("%7.3f / %7.3f = %7.3f%n", z,y,nanf);
```

INPUT DATA

To input value during the run time you can use

```
BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));

String text = br.readLine();
int age = Integer.parseInt(br.readLine());
double t = Double.parseDouble(br.readLine());
```

Or

```
Scanner sc = new Scanner(System.in);
name = sc.nextLine();
```

АРИФМЕТИЧНІ ОПЕРАТОРИ

Arithmetic Operators

- **+** Additive operator (also used for String concatenation)
- **-** Subtraction operator
- ***** Multiplication operator
- **/** Division operator
- **%** Remainder operator

```
int x = 11;
int y = 7;

int a = x + y;  // a = 18
int s = x - y;  // s = 4
int m = x * y;  // m = 77
int d = x / y;  // d = 1
int r = x % y;  // r = 4
```

УНАРНІ ОПЕРАТОРИ

```java
int x = 5; // x++; ++x; x = x + 1;
int a, b;
a = x++;  // a = 5 x = 6
x--;      // x = 5
b = ++x;  // b = 6 x = 6
++x;      // x = 7
boolean bool = true;
System.out.println(bool); // true
System.out.println(!bool); // false
```

Unary Operators

+	Unary plus; indicates positive value
-	Unary minus; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

RELATIONAL OPERATORS

Equality and Relational Operators

==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

```
int x = 5;
int y = -5;
System.out.println(x == y); // false
System.out.println(x != y); // true
System.out.println(x >= y); // true
```

BOOLEAN OPERATORS

The **&&** and **||** operators perform Conditional-AND and Conditional-OR operations on two boolean expressions

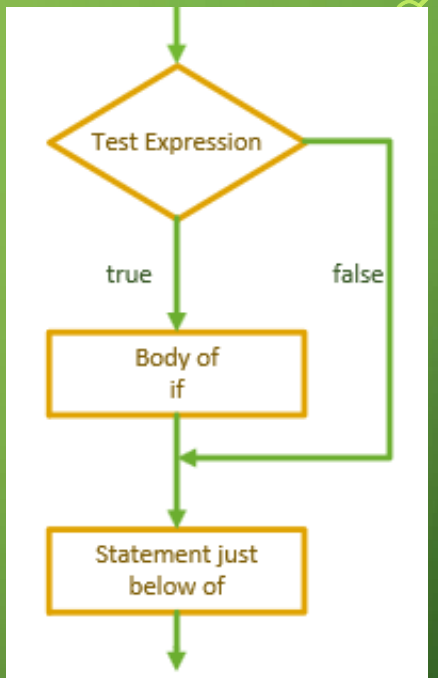
- **&&, & –** Conditional-AND
- **||, | –** Conditional-OR
- **! –** Logical NOT)

What is the result?

```
int t = 5, s = 4, v = 7;
System.out.println(t > s && t > v || s < v);
System.out.println(t > s || t > v && s > v);
System.out.println((t > s || t > v) && s > v);
```

IF STATEMENT

if statement is used to test a condition and decide the execution of a block of statements based on that condition result.
If the condition is **true**, the block of statements is executed;
if it is **false**, then the block of statements is ignored.




```
if (condition) {
    then-statement;
}
```

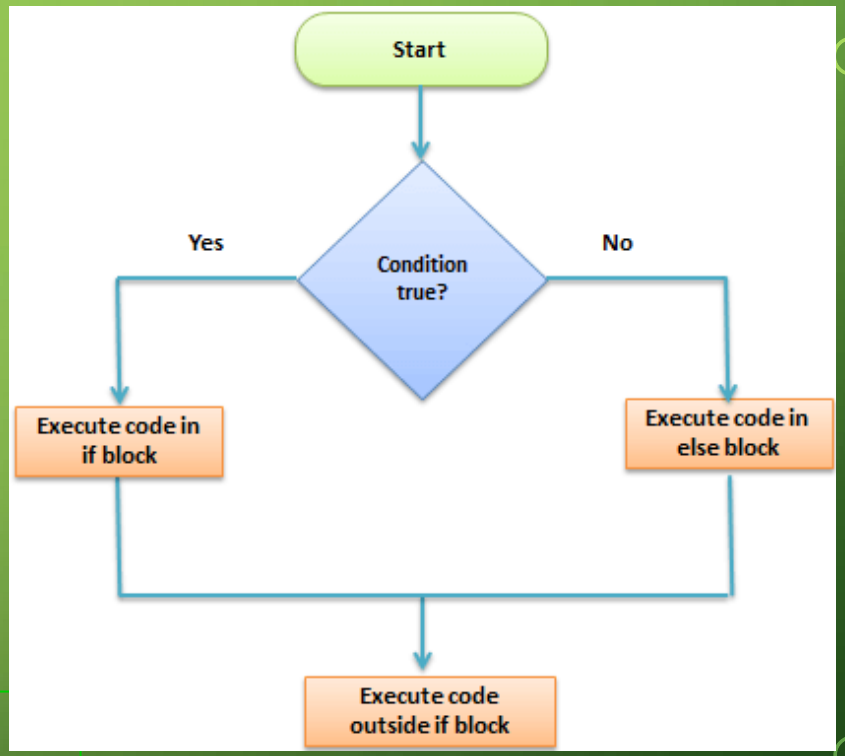
```
double mark = 4.3;

if (mark > 3) {
    System.out.println("You passed the exam");
}
```

IF-ELSE STATEMENT

if-else statement is used to run a specific block of a program if the condition is true or else

```
if (condition) {
    then-statement;
} else {
    else-statement;
}
```




```
if (temperature < 10) {
      System.out.println("It's too cold");
} else {
   System.out.println("It's Ok"); ;
}
```

Ternary Operator ? :

Ternary (conditional operator) consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide which value should be assigned to the variable

```
condition ? value1 : value2
```

What are the values of variable str1 and str2?

```
int t = 5, s = 4;
String str1 = (t >= ++s) ? "yes" : "no";

int a = 3, b = 2;
String str2 = a-- == b ? "yes" : "no";
```

SWITCH STATEMENT

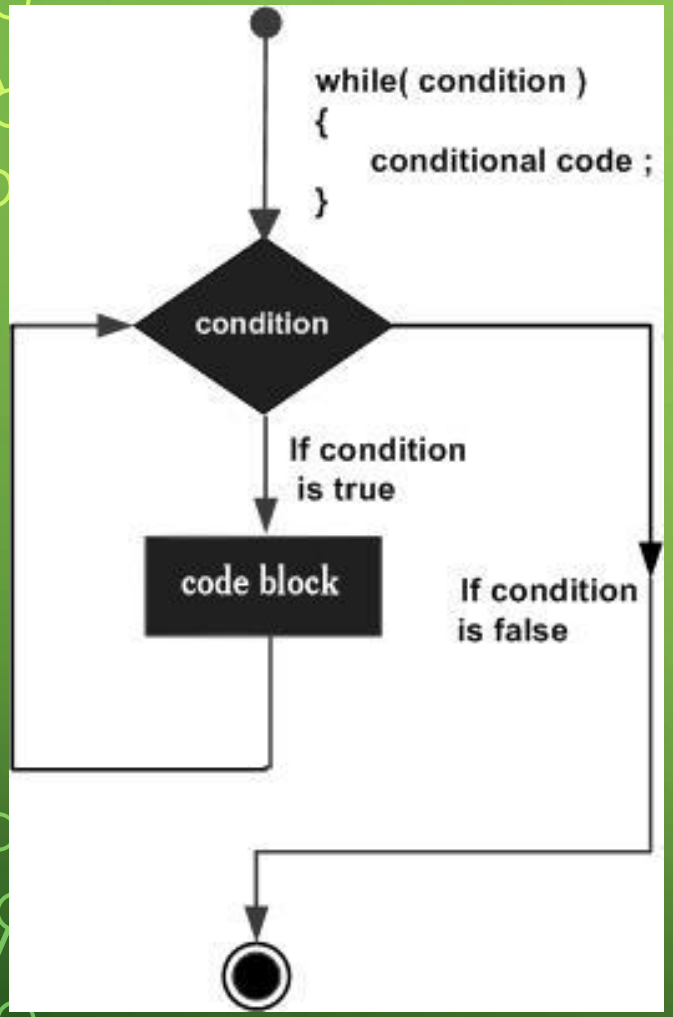
The ***switch*** statement selects one of many code blocks to be executed

The value of the expression is compared with the constant of each case. If there is a match, the associated block of code is executed.

The break and default keywords are optional

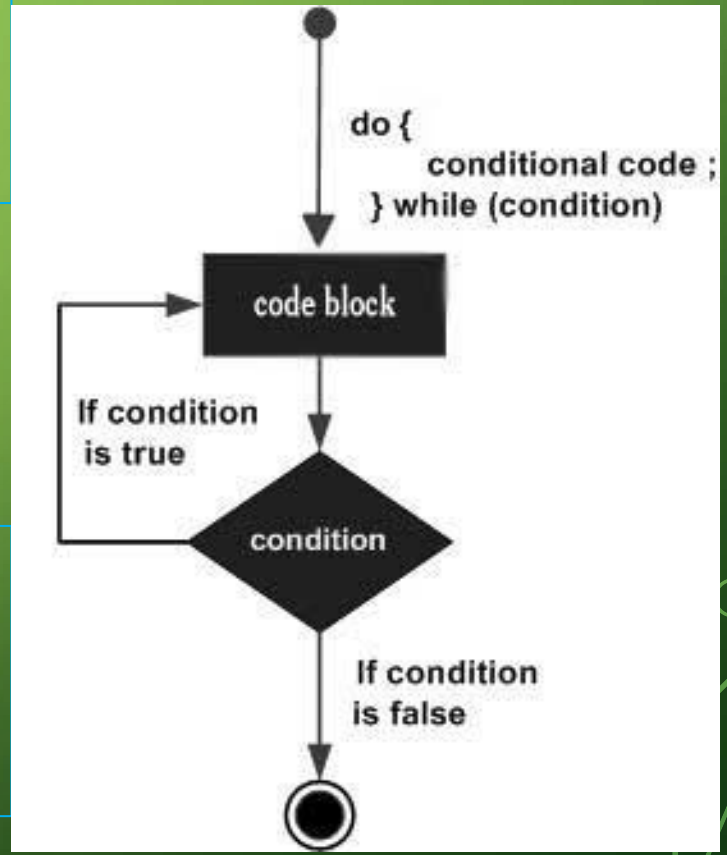
```
switch (expression)
{
      case const-expr1 :
            statement(s);
            break;
      case const-expr1 :
            statement(s);
            break;
      default :
            statement(s);
            break;
}
```

WHILE LOOPS




```
while (condition){
 statements;
}
```

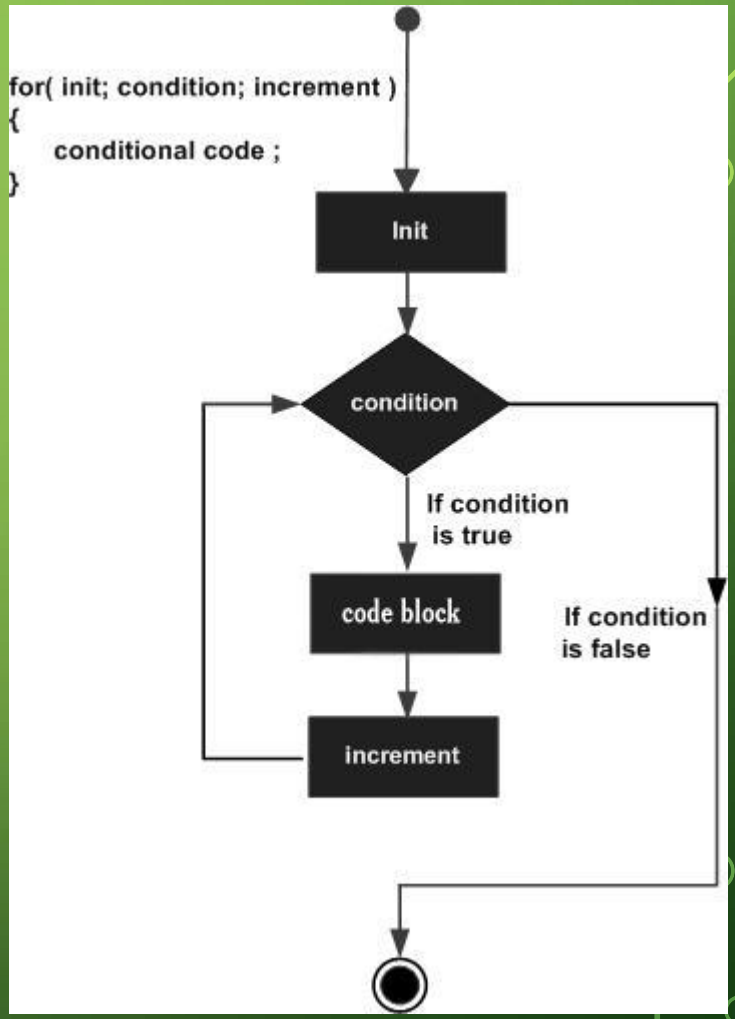
```
do {
 statements;
} while (condition);
```



FOR STATEMENTS

```
for (init; condition; increment) {
    statement(s);
}
```

```
for (type variable : collection) {
    statement(s);
}
```



The **break** statement terminates the for, while and do-while loop

```
Scanner sc = new Scanner(System.in);
int sum = 0;
int n;
for (int i = 0; i < 5; i++) {
    System.out.println("Input number");
    n = sc.nextInt();
    if (n < 0){
        continue;
    }
    sum += n;
}
System.out.println(sum);
sc.close();
```

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

int n = 0;
for (int i = 0; i < 5; i++) {
    System.out.println("Input number");
    n = sc.nextInt();
    if (n < 0){
        break;
    }
}
System.out.println(n);
sc.close();
```

The **continue** statement skips the current iteration the **for**, **while** and **do-while** loop

JAVA VS KOTLIN

```
public class Program2 {
    public static void main(String[] args) {
        int t, tm;
        System.out.println("Введіть число секунд, " +
                "що пройшли від початку доби: ");
            java.util.Scanner scanner =
                new java.util.Scanner(System.in);
        t = scanner.nextInt();//інструкція вводу даних
        tm = t / 60;    int s = t % 60;
        int h = tm / 60;   int m = tm % 60;
        System.out.printf("Час %02d:%02d:%02d\n", h, m, s);
    }
}
```

JAVA VS KOTLIN

```
object Program2b {
    @JvmStatic
    fun main(args: Array<String>) {
        val t: Int
        val tm: Int
        println( "Введіть число секунд, " +
                "що пройшли від початку доби: " )
        val scanner = java.util.Scanner(System.`in`)
        t = scanner.nextInt() //інструкція вводу даних
        tm = t / 60;         val s = t % 60
        val h = tm / 60;         val m = tm % 60
        System.out.printf("Час %02d:%02d:%02d\n", h, m, s)
    }
}
```

JAVA VS KOTLIN

```
fun main(args: Array<String>) {
    println("Введіть число секунд, " +
       "що пройшли від початку доби: ")
    val t: Int = readln().toInt()
    val tm = t / 60;     val s = t % 60
    val h = tm / 60;     val m = tm % 60
    println("%02d".format(h) + ":" +
        "%02d:%02d".format(m,s))
}
```