5 volt current = 1 bit binary

0 volt current = 0 bit binary

**Material Review**

* data types, arithmetic
* variables (data types – can set them to variables)
  + byte
  + char - *character* can be a single character – must be enclosed in single quotes *’G’*
  + int – *integer* variable that can be postive and negative numbers, including 0 as value
  + double – variable with floating point numbers (4.1)
  + float
  + boolean – data type that can pick up one logical value: true or false
  + String – (not really a type, its more like an object) type that can store *„text”* as value, enclosed in double qoutes

Operators:

* + **=** - assign values to variables

Arithmetric Operators: *They always (except string con.) return an integer value*

* + **+** - add numbers (we can concatenat „összefűzni” strings)
  + **-** - substract numbers
  + **\*** - multiply numbers
  + **/** - divide numbers
  + **%** - *modulo,* returns the remainder of dividing two numbers

Relational Operators: *They always return a boolean value (true or false) !*

* + **>=** - greater than / equal to
  + **<=** - less than / equal to
  + **>** - greater than
  + **<** - less than

Equality Operators: *They always return a boolean value (true or false) !*

* + **==** - equal to
  + **!=** - not equal

Assignment operator: *They always return an integer value*

* + **+=** - add number on the right to the variable left
  + **-=** - subtract number on the right to the variable left
  + **\*=** - multiply number on the right to the variable left
  + **/=** - divide number on the right to the variable left

Logical operators:

* + **||** - OR operators, one statment has to be true to return true
  + **&&** - AND operator, both statment has to be true to return true
  + **!** – NOT operator, will return the opposite of the expression

**Whitespace** - is one or more characters (such as a space, tab, enter, or return) that do not produce a visible mark or text. Whitespace is often used to make code visually presentable.

**Control Flow** -

**CTRL + /** - comments out the selected lines

**CTRL + SHIFT + up or down** – Moves the complete line of code up or down

**CTRL + ALT + L** – indents the code in right order

**! JAVA reads statments left to right, if they not separeted by brackets!**

boolean a = true || true && false || false && true

**Scanner**: *getting user input*

System.out.println(„Please enter your name”);

Scanner userInput = new Scanner(System.in);

String input = userInput.nextLine ();

System.out.println(„Your name is: ” + input);

**IF / ELSE IF / ELSE statements:**

if (b)}

System.out.println(„true”)

{

else if (a) {

System.out.println („true”)

}

else {

System.out.println(„false”)

}

*if* – Checking the the condition given, if the condition is true execute statment, if false skips the code

*else if* **-** Checking the the condition given, if the condition is true execute statement, if false skips the

code

*else* – if every statments above is false (its does not have a condition), then it executes the else statment

**WHILE / DO WHILE / FOR (loop) statment:**

*Using WHILE loop:*

int a = 0;

while (a < 100) {

System.out.println(a);

a ++; (it means add 1 to variable every timet he statement runs)

}

System.out.println(„Loop finished!”)

It is going to run for 100 time, until it adds to a = 100, then it comes out of the while loop.

*Using FOR loop:*

for (int a = 0 ; a < 100; a++) {

System.out.println(a)

}

System.out.println(„Loop finished!”)

Give the statement a variable, then a coondition, then an operation.

*Using DO WHILE loop:*

int a = 10;

do {

System.out.println(„I print it out even the statement is false

} while (a < 10) {

System.out.println(„This loop is True!”)

}

System.out.println(„Loop finished!”)

The DO statement going to run the code given before the while loop checks its conditions, so we get an output even the loop condition is false and not going to run.

* difference between variable declaration and statements
* **System.out.println()** – prints out the value givin on the Terminal
* Hello World
* escape character
* "Java's stuff" and 'stuff of Java'
* concatenation
* condition, if, else
* loops: for, while
* block
* scanner
  + System.in
  + nextLine()
  + nextInt()

**Ternary Conditional Statement:** compresses the if and else statments into one statment

int fuelLevel = 3;

char canDrive = (fuelLevel > 0) ? 'Y' : 'N';

System.out.println(canDrive);

**Switch Statement:** Java provides a way to execute code blocks based on whether a block is equal to a specific value. For those specific cases, we can use the switch statement

int restaurantRating = 3;

switch (restaurantRating) {

case 1: System.out.println("This restaurant is not my favorite.");

break;

case 2: System.out.println("This restaurant is good.");

break;

case 3: System.out.println("This restaurant is fantastic!");

break;

default: System.out.println("I've never dined at this restaurant.");

break;