|  |  |
| --- | --- |
|  | **Cours:** Object-Oriented Programming  **Deadline:** December 17, 2023 till (23:59) |

**Instructor:** Toleu Altynbek

**ASSIGNMENT 1**

**Lesson plan:**

**Introduction. Installing the IDE**

**Variables**

**Types**

**Conditional operator**

**Cycles**

Java is a strongly typed object-oriented programming language. It was created in 1995 and has received many improvements since then. Consistently occupies a leading position in rankings of programming languages and in rankings of developer salaries.

And most importantly, the Java rating does not jump from year to year, but remains consistently high. Let's figure out what makes Java popular.

**1. Cross-platform** - the written code is translated into bytecode, which is executed by the JVM. There are implementations of the JVM for a variety of platforms, and this means that the code written once will work on Windows, Linux, macOS, and even on various exotic devices, such as Arduino, a smart refrigerator, a vacuum cleaner... That is, the code will work on different platforms and operating systems, and at the same time it does not need to be adapted for each of them.

**2. Automatic memory management** - the developer does not need to think about which RAM addresses the variables are stored in, manually ensure they are read/written, and worry about data integrity. When you solve a customer’s business problem, you need to think about the problem, and not about how and where to write some bytes.

**3. Speed of operation (just-in-time compiler)** – in addition to static compilation, which occurs “in advance,” dynamic compilation works. This is true for server code, which can run for months or even years. Code that is executed frequently is compiled in different variants and its execution time is measured. Thus, the longer the application runs, the faster it runs. And this is with the server actively running. Cool, right?

**4. Backward compatibility** - code written in older versions of Java will work in newer versions. This is convenient: after updating, you don’t need to rewrite half of the project “for the sake of updating,” and you can still get the latest security patches.

**5. Object-oriented** - a person thinks in objects: a table, a trolleybus, a smartphone. The developer does not need to think in unusual paradigms while working, and this protects our code from unnecessary errors. You need to concentrate only on those details that are important within the task. For example, from the point of view of an interior designer, the size of the table and its position in space are important. The date of its production, the name of the worker who cut this table, the telephone number of the courier who delivered it are not important. In addition, in the code, data and methods for working with them are stored together.

**6. Static typing (fail-fast)** – checking the consistency of variable types at the compilation stage. All developers run the compilation, so compilation errors are caught almost instantly. The later the stage at which the error is discovered, the more expensive it is to correct it.

**7. Code as documentation** – Java reads like sentences in English. Due to this, in most cases there is no need to spend effort on maintaining documentation, since any developer, having looked at the code, understands what this method does or what behavior the interface is responsible for. In addition, the practice of “correctly” naming all entities in the code is highly developed. Often the name of a method tells you what it does.

For example, the getContext() method returns the context, and the age field is responsible for storing the age. In Java, the length of the name does not affect the amount of system resources needed to work with entities. This is where C sins - when a developer comes to a new project, instead of understanding the logic of the code, he deciphers it.

**8. Many open source libraries and** frameworks - 99% of the problems that a developer faces in daily practice have already been solved by someone at some point. If such solutions are successful, over time they grow into libraries and even frameworks. What's better - Google for 5 minutes or reinvent your own bicycle on square wheels?

**9. Large community** - a popular language has many developers, they ask a lot of questions on the Internet, give a lot of answers, write a lot of code, encounter many problems and solve them. And the more developers there are, the more popular the language is, and the faster it develops. A vicious circle, however.

Many positive “qualities” of Java have already been mentioned, but I would like to add a few more aspects:

The JVM (Java Virtual Machine) itself manages the process memory: due to this, it is safe, and is the #1 language for financial instruments.

It is very convenient to write back-end servers (server logic) in Java.

Until recently, Java was the main language for Android applications.

Now, instead of it, Kotlin, which is a JVM language, differs from Java in the presence of a couple of features and “syntactic sugar”. Switching from Java to Kotlin and vice versa will take several days. And given that the Java update cycle is now six months, it is possible that in the next release it will have all the hype that Kotlin was written for.

Many famous companies use Java: Google, Facebook, Twitter, Amazon, LinkedIn, eBay, JavaRush and many, many others.

**Install your preferred IDE on desktop. Project folder should be archived together with IDE data. You can use** [**this link**](https://www.jetbrains.com/shop/eform/students) **to apply for student license free of charge.**

Variables are special things for storing data. Any data. All data in Java is stored using variables. Most of all, a variable in meaning resembles a box: the most ordinary box.

|  |  |
| --- | --- |
| **Code** | **Description** |
| **int** i;  **int** a, b;  **int** x; | The variable i is created  Variables a, b are created  Variable x is created |
| i = 3; | The value 3 is entered into the variable i. |
| a = 1;  b = a + 1; | The variable a is set to 1.  The variable b is set to 2. |
| x = 3;  x = x + 1; | The variable x is set to 3.  On the next line, the value of x is increased by 1, x is now equal to 4. |

**Task 1**

|  |
| --- |
| Write a program that declares the following variables in the **main** method:  name is of type **String,** age is of type **int** and city is of type **String**.  Note: "declare a variable" means the same as "create a variable".  Requirements:  • Declare a variable name of type String.  • Declare a variable age of type int.  • Declare a variable city of type String.  • 3 variables must be declared. |

**Task 2**

|  |
| --- |
| Declare variables a, b of type int.  Immediately in the declaration line assign them different values.  The values can be any integer.  Requirements:  • Declare two variables of type int.  • Variable names must be a and b.  • Variables must be assigned values immediately.  • Variables a and b must have different values. |

**Task 3**

|  |
| --- |
| It is now 3126. My friend was born 8 years ago.  Write a program that displays the year of birth of my friend.  Requirements:  • The program must use screen output.  • The year displayed must be 4 digits long.  • The year displayed must begin with "31".  • The year derived must match the assignment. |

**Task 4**

|  |
| --- |
| In the main method, declare the variables word, phrase, line and text of type String.  Give them different meanings. Use one command to create and initialize variables.  The values can be any strings.  Requirements:  • In the main method, declare four variables of type String.  • Name the variables word, phrase, line and text.  • Immediately assign values to variables.  • All variables must have different values. |

**Conditional operator**

if-else statement

Programs would be of little use if they did the same thing no matter how external circumstances changed. The program needs to be able to adapt to different situations and do some actions in some cases, others in others.

In Java, this is implemented using a conditional operator - a special keyword that allows you to execute different blocks of commands depending on the truth of the condition.

A conditional statement consists of three parts: a condition, command 1 and command 2. If the condition is true (true), command 1 is executed, otherwise command 2 is executed. The commands are never executed simultaneously. The general form of this operator is as follows:

|  |
| --- |
| **if (condition)**  **command 1;**  **else command 2;** |

|  |  |
| --- | --- |
| **Code** | **Description** |
| **int** age = 17;  **if** (age < 18)  System.out.println("You're still a child");  **else**  System.out.println("You are already an adult"); | The following message will be displayed on the screen:  You're still a child |
| **int** temperature = 5;  **if** (temperature < 0)  System.out.println("It's frosty outside");  **else**  System.out.println("Warm"); | The following message will be displayed on the screen:  Warm |
| **int** age = 18;  **if** (age == 18)  System.out.println("Come to the recruiting office");  **else**  System.out.println("Come anyway"); | The following message will be displayed on the screen:  Come to the recruiting office |

**Task 5**

|  |
| --- |
| Enter the outside temperature using the keyboard. If the temperature is less than 0, display the inscription “it’s cold outside,” otherwise, display the inscription “it’s warm outside.”  Requirements:  • The program must read the temperature value from the keyboard.  • The program must use the System.out.println() or System.out.print() command.  • If the temperature is less than 0, display only the message “it’s cold outside.”  • If the temperature is greater than or equal to 0, display only the message “it’s warm outside.” |

**Task 6**

|  |
| --- |
| Enter your name and age using the keyboard. If the age is between 18-28 (inclusive), then display the message “Name, come to the military registration and enlistment office”, where Name is the name entered earlier from the keyboard.  Example input:  Alex  18  Example output:  Alex, come to the military registration and enlistment office  Requirements:  • The program must read data from the keyboard twice.  • The program must use the System.out.println() or System.out.print() command.  • If the age is between 18-28 (inclusive), then display only the message “Name, come to the military registration and enlistment office.”  • If the age is not within the specified limits, then nothing needs to be displayed. |

**Task 7**

|  |
| --- |
| The program reads the body temperature value from the keyboard and displays a message indicating that the body temperature is high, low or normal, depending on the conditions.  The class declares two Boolean variables isHigh (high temperature) and isLow (low), into which you need to put the corresponding conditions and use these variables instead of comparison expressions.  Requirements:  • The program must read the temperature value from the keyboard.  • Do not change the declaration of the isHigh and isLow variables.  • The variables isHigh and isLow in the main method must be assigned appropriate comparison expressions.  • The variables isHigh and isLow should be used instead of comparison expressions in if() statements.  • The functionality of the program should not change. |

**Task 8**

|  |
| --- |
| Let's write a program that will read age from the keyboard. If the age is from 20 to 60 (inclusive), then you do not need to display anything, otherwise, display the phrase “you don’t have to work.” This can (and should!) be done using only one if statement without else.  Hint: use the logical operator "||" (or).  Requirements:  • The program must read one number from the keyboard.  • If the age is less than 20 or more than 60, then you need to display the phrase “you don’t have to work.”  • If the age is in the range from 20 to 60 (inclusive), then nothing needs to be displayed.  • The program must use one if statement without else and the logical operator ||. |

**loop**

The if-else statement has greatly expanded our programming capabilities: we can now write programs that perform different actions in different situations. But there is one more thing that will make our programs an order of magnitude more powerful - loops.

There are 4 types of loops in Java: while, for, for-each and do-while. Now we will analyze the very first of them.

The while loop statement is very simple and consists of only two parts: a condition and a loop body. The body of the loop is executed over and over again as long as the condition is true. The general form of a while loop is this:

|  |  |
| --- | --- |
| **Code** | **Description** |
| **int** n = 0;  **while** (n < 10)  {  System.out.println(n);  n++;  } | 10 lines will be displayed on the screen:  0  1  ...  8  9 |

**Task 9**

|  |
| --- |
| Using a while loop, display the quote one hundred times (quote variable):  “I will never work for pennies."  Output each value on a new line.  Requirements:  • The program must display text on the screen.  • The program must display the contents of the quote variable 100 times.  • The program must use a while loop. |

**Task 10**

|  |
| --- |
| Let's write a program in which we need to enter integers from the keyboard and count their sum until the user enters the word "ENTER".  Display the received amount and end the program.  Requirements:  • The program must read data from the keyboard.  • It is necessary to calculate the sum of the entered integers and display it on the screen if the user has entered a stop word.  • The program must use a while loop. |

**Task 11**

|  |
| --- |
| Using nested while loops (a loop within a loop), display a 5 (height) by 10 (width) rectangle filled with the letter 'Q'.  Example output:  QQQQQQQQQQ  QQQQQQQQQQ  QQQQQQQQQQ  QQQQQQQQQQ  QQQQQQQQQQ  Requirements:  • The program must display text on the screen.  • The program should output a rectangle of height 5 and width 10 filled with the letters 'Q'.  • The program must use nested while loops (loop within a loop).  • Printing to the screen should occur in a while loop. |

**Task 12**

|  |
| --- |
| Display the sum of numbers from 1 to 100 inclusive, not divisible by 3. To do this, use a while loop.  Hint: to move to the next number in a loop, use the continue operator.  Requirements:  • The program should print to the console the sum of all numbers from 1 to 100 inclusive, not divisible by 3.  • The program must use a while loop.  • The program must use the continue statement. |

**Comparison of loops: for vs while**

The while loop can be used whenever a command or group of commands needs to be executed multiple times. But of all the cases, one case stands out in particular.

This is the case when the programmer (author of the program) knows in advance how many times his loop should be executed. This is usually solved by introducing a special counter variable, and each iteration of the loop the variable is incremented (or decremented) by 1.

Everything seems to work as it should, but it’s not very convenient. Before the loop, we set the starting value of the counter variable, then in the condition we check whether it has already reached the final value. But we usually change it at the very end of the cycle body.

What if the body of the cycle is large? Or do we have several nested loops? In general, in such cases, I would like to collect all this information about the counter variable in one place. This is how the for loop appeared in Java. It doesn't look very complicated either:

|  |
| --- |
| for (command1; condition; command2)  {  COMMAND BLOCK  } |

|  |  |
| --- | --- |
| **Loop while** | **Loop for** |
| **int** i = 3;  **while** (i >= 0)  {  System.out.println(i);  i--;  } | **for** (**int** i = 3; i >= 0; i--)  {  System.out.println(i);  } |
| **int** i = 0;  **while** (i < 3)  {  System.out.println(i);  i++;  } | **for** (**int** i = 0; i < 3; i++)  {  System.out.println(i);  } |
| **boolean** isExit = false;  **while** (!isExit)  {  String s = console.nextLine();  isExit = s.equals("exit");  } | **for** (**boolean** isExit = false; !isExit; )  {  String s = console.nextLine();  isExit = s.equals("exit");  } |
| **while** (true)  System.out.println("C"); | **for** (; true; )  System.out.println("C"); |
| **while** (true)  {  String s = console.nextLine();  **if** (s.equals("exit"))  **break**;  } | **for** (; true; )  {  String s = console.nextLine();  **if** (s.equals("exit"))  **break**;  } |

**Task 13**

|  |
| --- |
| Write a program in which, using a for loop, even numbers from 1 to 15 will be displayed on the screen.  Each value must be printed on a new line.  Requirements:  • The program must display text on the screen.  • The program should print even numbers from 1 to 15. Print each value on a new line.  • The program must use a for loop. |

**Task 14**

|  |
| --- |
| In the main method, 3 integers are read from the keyboard: start, end (start <= end), multiple.  Add a program so that the sum of numbers from start (inclusive) to end (not inclusive), multiples of multiple, is displayed on the screen.  To do this, use a for loop.  Hint: To move to the next iteration of a loop, use the continue statement.  Requirements:  • The program should print to the console the sum of numbers from start (inclusive) to end (not inclusive), multiples of multiple.  • The program must use a for loop.  • A continue statement must be used in a for loop. |

**Task 15**

|  |
| --- |
| Let's use a for loop to display a right triangle of eights with sides 10 and 10.  That is, the first line should have one 8, starting from the left, the second line should have two, etc.  Example of screen output:  8  88  888  8888  88888  888888  8888888  88888888  888888888  8888888888  Requirements:  • The program should display numbers on the screen.  • The program should produce a right-angled triangle of eights with sides 10 and 10.  • The program must use a for loop.  • Output to the screen must occur in a for loop. |