**JAVA**

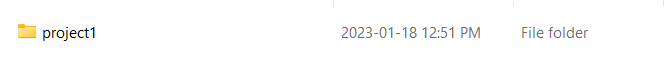
* Every java file need a header to describe
* Cach thuc hoat dong: dau tien no se convert file HelloWorld.java into file HelloWorld.class, va sau do no se execute file class HelloWorld
* If (var1 = true) means that it will store value true to var1, then it will evaluate the value of var1 with condition
* The name convention of package is lowercase and need dot (.) between the level of folder

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated



* Java and C# have the garbage collection which will delete automatically the dynamic object

Graphical user interface, text, application

Description automatically generated

1. Big Decimal

* Equivalent to type long double in C# ?
* The BigDecimal class provides operations on double numbers for arithmetic, scale handling, rounding, comparison, format conversion and hashing. It can handle very large and very small floating point numbers with great precision but compensating with the time complexity a bit.
* A BigDecimal consists of a random precision integer unscaled value and a 32-bit integer scale. If greater than or equal to zero, the scale is the number of digits to the right of the decimal point. If less than zero, the unscaled value of the number is multiplied by 10^(-scale).
* BigDecimal đại diện cho một số thập phân có độ chính xác cao. Một BigDecimal object là immutable và được chia làm 2 phần:
  + Precision- Biểu diễn tất các ký số có trong BigDecimal dưới dạng số nguyên không phân biệt phần thập phân.
  + Scale (32 bit) – Số nguyên biểu diễn số chữ số thập phân.
* Khởi tạo BigDecimal: Chúng ta có thể khởi tạo BigDecimal object từ String, mảng character int, long, double, BigInteger.

Text

Description automatically generated

* + Lưu ý: Chúng ta cũng có thể tạo BigDecimal với double. Hình bên dưới đã khởi tạo BigDecimal với giá trị là 0.1d, mong muốn BigDecimal của mình mang giá trị 0.1 thế nhưng kết quả lại khác hoàn toàn. Đó là vì 0.1 không có giá trị chính xác trong double, BigDecimal chỉ đơn giản là lấy giá trị sai của 0.1 trong double sang và tất nhiên là giá trị cũng sẽ bị sai.

Text

Description automatically generated

* The BigDecimal class provides operations for arithmetic, scale manipulation, rounding, comparison, hashing, and format conversion
* We need

1. Header comment

* To type header comment fast, type /\*\* then press Enter

1. Stack and Heap:

* Stack will contain the primitives vatiable and the name of object with value is the memory address of that object, and that memory address will reference the object in Heap
* Heap will store the properties of that object with the name is memory address of that object, and the memory address of that object in heap will the same address memory in stack

A white paper with writing on it

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

This is wrong because wrapper type is immutable which is can’t change the memory space

Diagram, whiteboard

Description automatically generated

This is correct

Except the memory address in heap, what else can exist in heap?

How do wrapper class work? Are wrapper class immutable or not? Example

Text, whiteboard

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

The double equal comapares the value of variable in stack, if it contain primitive types in will compare the value of variable, or if the variable contain memory address, it will compare the memory address

1. Java Virtual Machine (JVM):

* Vietnamese Document: <https://topdev.vn/blog/jvm-la-gi/>
* The Java virtual machine manages (quản lý) application memory (bộ nhớ ứng dụng) and provides a portable execution environment (môi trường thực thi di động) for Java-based applications
* Developers reap (gặt haiis) the rewards in performance, stability, and predictable runtimes.
* The Java virtual machine is a program whose purpose is to execute other programs?
* Java Virtual Machine has two primary functions:
  + Allow Java programs to run on any device or operating system (known as the "write once, run anywhere" principle)
  + Manage and optimize program memory

Graphical user interface

Description automatically generated with low confidence

* Technical definition: The JVM is the specification for a software program that executes code and provides the runtime environment for that code
* Everyday definition: The JVM is how we run our Java programs. We configure (định cấu hình) the settings and then rely on the JVM to manage program resources during execution.
* When developers talk about the JVM, we usually mean the process running on a machine, especially a server, that represents and controls resource usage for a Java application
* JVM languages:
  + While it was once only for Java, the JVM is flexible and powerful enough to support many other languages today.
  + Among the most popular are Scala, used for real-time, concurrent applications, and Groovy, a dynamically typed scripting language. Another prominent example is Kotlin, which delivers a blend of object-oriented and functional styles
  + All of these (Scala, Groovy, Kotlin) are considered JVM languages, meaning that, even though they are not coding in Java, the programmer retains access to the vast ecosystem of Java librarie0073
* Garnage Collection:
  + The most common interaction with a running JVM is to check the memory usage in the heap and stack.
  + The most common adjustment is performance-tuning the JVM's memory settings.
  + Program memory is managed by the JVM. The JVM manages memory through a process called garbage collection, which continuously identifies and eliminates unused memory in Java programs. Garbage collection happens inside a running JVM.
* The three parts of the JVM:
  + The JVM specification:
    - The JVM is a software specification
    - the JVM spec highlights that its implementation details are not defined (xác định) within the spec in order to allow for maximum creativity in its realization (quá trình thực hiện)
    - To implement the Java virtual machine correctly, you need only be able to read the class file format and correctly perform the operations specified therein
    - All the JVM has to do is run Java programs correctly
    - It's a massive undertaking (công việc to lớn), especially given the power and flexibility of the Java language

Graphical user interface, text, application, email

Description automatically generated

* + JVM implementations:
    - Implementing the JVM specification results in an actual software program, which is a JVM implementation
    - There are many JVM implementations, both open source and proprietary (độc quyền)
      * OpenJDK's HotSpot is the JVM reference implementation. It remains one of the most thoroughly tried-and-tested codebases in the world.
    - HotSpot may be the most commonly used JVM, but it is by no means the only one
      * Another interesting and popular implementation is GraalVM which features high performance and support for other, traditionally non-JVM languages like C++ and Rust via the LLVM spec
      * There are also domain-specific JVMs like the embedded robotics JVM, LeJOS
  + A JVM instance:
    - After the JVM spec has been implemented and released as a software product, you may download and run it as a program. That downloaded program is an instance (or instantiated version) of the JVM.
    - Most of the time, when developers talk about "the JVM," we are referring to a JVM instance running in a software development or production environment

Graphical user interface, text

Description automatically generated

* In order to run Java applications, the JVM depends on the **Java class loader** and **a Java execution engine**:
  + The Java class loader:
    - Everything in Java is a class, and all Java applications are built from classes. An application could consist of one class or thousands
    - In order to run a Java application, a JVM must load compiled .class files into a context, such as a server, where they can be accessed
    - A JVM depends on its class loader to perform this function
    - When you type java classfile, you are saying: start a JVM and load the named class into it
    - The Java class loader is the part of the JVM that loads classes into memory and makes them available for execution
    - Class loaders use techniques like lazy-loading and caching to make class loading as efficient as it can be
    - That said, class loading isn't the epic brain-teaser that (say) portable runtime memory management is, so the techniques are comparatively simple?
    - Every Java virtual machine includes a class loader
    - The JVM spec describes standard methods for querying and manipulating the class loader at runtime, but JVM implementations are responsible for fulfilling these capabilities (đáp ứng những khả năng này)
    - From the developer's perspective, the underlying class loader mechanism is a black box
  + The execution engine:
    - Once the class loader has done its work of loading classes, the JVM begins executing the code in each class
    - The *execution engine* is the JVM component that handles this function
    - The execution engine is essential to the running JVM. In fact, for all practical purposes, it is the JVM instance
    - Executing code involves managing access to system resources
    - The JVM execution engine stands between the running program—with its demands for file, network, and memory resources—and the operating system, which supplies those resources
    - System resources can be divided into two broad categories:
      * Memory
      * Everything else
    - Recall that the JVM is responsible for disposing of unused memory, and that garbage collection is the mechanism (cơ chế) that does that disposal
    - The JVM is also responsible for allocating and maintaining the referential structure that the developer takes for granted. Example: The JVM's execution engine is responsible for taking something like the new keyword in Java, and turning it into an operating system-specific request for memory allocation
    - The execution engine manages resources for file system access and network I/O
    - Since the JVM is interoperable (có thể tương tác) across (trên) operating systems, this is no mean task. In addition to each application's resource needs (ngoài nhu cầu tài nguyên của mỗi ứng dụng), the execution engine must be responsive (đáp ứng) to each operating system environment. That is how the JVM is able to handle in-the-wild demands

1. Java Development Kit (JDK):

* Vietnames document: <https://bizflycloud.vn/tin-tuc/jdk-la-gi-2018121318321223.htm>
* JDK viết tắt của Java Development Kit là một bộ phần mềm cung cấp môi trường phát triển ứng dụng viết bằng ngôn ngữ Java. JDK bao gồm cả Java Runtime Environment giúp lập trình viên có thể chạy thử để kiểm tra ứng dụng trong quá trình phát triển ứng dụng
* JDK, JRE, Java, Java Virtual Machine và Java Compiler có gì khác nhau?

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

* The JDK is the development platform for building Java applications
* The Java Development Kit (JDK) is **one of three core technology packages** **used in Java programming,** along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment)
* The difference between those three technology packages are:
  + **The JVM** is the runtime that hosts running programs.
  + **The JRE** is the on-disk part of Java that creates the JVM and loads programs into them.
  + **The JDK** **provides the tools** necessary **to write Java programs** that can be executed and run by the JVM and JRE
  + **The distinction between JDK and JRE** is that the **JDK** is a **package of tools** for **developing Java-based software**, whereas the **JRE** is a **package of tools** for **running Java code**
* The JRE can be used as a standalone component to simply run Java programs, but it's also part of the JDK. The JDK requires a JRE because running Java programs is part of developing them.
* How the JDK fits into the Java application development lifecycle:

Diagram

Description automatically generated

* **Technical definition of the JDK (định nghĩa chuyên ngành)**: The JDK is an implementation of the Java platform specification, which includes the compiler and standard class libraries
* **Everyday definition of the JDK (định nghĩa thông thường)**: The JDK is a software package you download in order to create Java-based applications.

Graphical user interface, text

Description automatically generated

* JDK versions and package

Graphical user interface, text, application

Description automatically generated

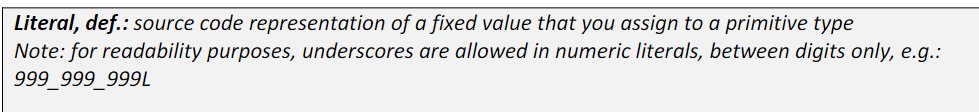
Graphical user interface, text

Description automatically generated

* How to install the JDK and Download for Java SE: <https://www.infoworld.com/article/3296360/what-is-the-jdk-introduction-to-the-java-development-kit.html>
* Two key Java commands: java and javac
  + The JRE inside your JDK adds the java command to your command line. You can verify this by dropping into a command shell and typing java -version, which should return the Java version you've just installed. (In some cases you'll have to restart your system for this change to your system path to fully take.)
  + You'll need javac which is JDK component to compile your Java files
  + The javac command lives inside the /jdk directory, and in recent versions of the installer will automatically be added to the path .... Some IDEs include a Java compiler by default. It is usually possible to configure them to use a specific installed version if you wish.
* Compile and run a Java program: <https://www.infoworld.com/article/3296360/what-is-the-jdk-introduction-to-the-java-development-kit.html>

1. Primititve Types (Built into the language):

* byte (8 bits/1 byte), range: -128 -> +127; default value: 0
* short(16 bits/2 bytes), range: -32,768 ... +32,767; default value: 0
* int (32 bits/4 bytes), range: -231... +231-1; -2147483648 to 2147483647, default value: 0; literals: decimals, hexdecimal (0x1a), binary (0b11010)
* long(64 bits/8 bytes), range:-263... +263-1; default value: 0L; use capital L at the end of an integer literal, default int
* float (32 bits/4 bytes), 32-bit IEEE 754 floating point, use it to save memory, but use BigDecimal for precise values; default value: 0.0for 0.0F; e/E for scientific notation accepted, e.g.: 1.234e2
* double (64 bits/8 bytes), 64-bit IEEE 754 floating point, default choice for decimal, but not for precise values; default value: 0.0dor 0.0D
* boolean(1bit-of-information/undefined-size, JVM-dependent), values: true, false; default value: false. Size of boolean: <https://stackoverflow.com/questions/383551/what-is-the-size-of-a-boolean-variable-in-java>
* char(16 bits/2 bytes), range: ‘\u0000’ ... ‘\uffff’; default value: ‘\u0000’Unicode: unique code for characters. char represent numbers ranging from 0 to 65,535. Unicode: <https://docs.oracle.com/javase/tutorial/i18n/text/unicode.html>
* Default values are automatically assigned to (class) attributes, but local variables need to be explicitly assigned.



* Naming Convention: [https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html#naming](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html%23naming)
  + Variables names are case-sensitive
  + A variable's name can be any legal identifier - an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "$", or the underscore character "\_". However, the convention is to always begin your variable names with a letter, not "$" or "\_". The dollar sign character, by convention, is never used at all
  + You may find some situations where auto-generated names will contain the dollar sign, but your variable names should always avoid using it
  + A similar convention exists for the underscore character; while it's technically legal to begin your variable's name with "\_", this practice is discouraged
  + White space is not permitted
  + Subsequent characters may be letters, digits, dollar signs, or underscore characters. Conventions (and common sense) apply to this rule as well
  + When choosing a name for your variables, use full words instead of cryptic abbreviations (chữ viết tắt khó hiểu). Doing so will make your code easier to read and understand. In many cases it will also make your code self-documenting; fields named cadence, speed, and gear, for example, are much more intuitive than abbreviated versions (trực quan hơn nhiều so với các phiên bản rút gọn), such as s, c, and g
  + Also keep in mind that the name you choose must **not be** a keyword or reserved word
  + If the name you choose consists of only one word, spell that word in all lowercase letters
  + If it consists of more than one word, capitalize the first letter of each subsequent word. The names gearRatio and currentGear are prime examples of this convention
  + If your variable stores a **constant** value (declare the constant variable by using keyword final) such as static final int NUM\_GEARS = 6, the convention changes slightly, capitalizing every letter and separating subsequent words with the underscore character. By convention, the underscore character is never used elsewhere.

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

1. StringTokenizer

* The string tokenizer class in java is legacy code and it is replaced with string.split() method
* StringTokenizer class in Java is used to break a string into tokens. A StringTokenizer object internally maintains a current position within the string to be tokenized. Some operations advance this current position past the characters processed
* A token is returned by taking a substring of the string that was used to create the StringTokenizer object
* The String Tokenizer class allows an application to break strings into tokens
* To use String Tokenizer class we have to specify an input string and a string that contains delimiters. Delimiters are the characters that separate tokens. Each character in the delimiter string is considered a valid delimiter. Default delimiters are whitespaces, new line, space, and tab.

Diagram, timeline

Description automatically generated

* Overload constructors of StringTokenizer:

Graphical user interface, text, application

Description automatically generated

* Syntax:

Create an object of class StringTokenizer

StringTokenizer object\_name = new StringTokenizer(“The-string-you-want-to-separate”, ”-”);

=> output: it will return the array of [“The”, “string”, “you”, “want”, ”to”, “separate”]

* If we don’t put anything for the delim, it will consider you separate by the whitespaces, new line, space, and tab.
* The last parameter of StringTokenizer will include the the white space as the token too

Graphical user interface, text, application

Description automatically generated

* Some methods in StringTokenizer

Graphical user interface

Description automatically generated with medium confidence

* Vietnamese Document: Click [here](https://t3h.com.vn/tin-tuc/stringtokenizer-trong-java#:~:text=M%E1%BB%99t%20l%E1%BB%9Bp%20StringTokenizer%20l%C3%A0%20m%E1%BB%99t,th%C3%A0nh%20c%C3%A1c%20m%C3%A3%20th%C3%B4ng%20b%C3%A1o.)
* Một lớp StringTokenizer là **một lớp có trong java.util gói** và nó được sử dụng để chia một Chuỗi thành các mã thông báo. Nói cách khác, chúng ta có thể chia một câu thành các từ của nó và thực hiện các thao tác khác nhau như đếm số lượng mã thông báo hoặc chia một câu thành các mã thông báo. Có các hàm tạo và phương thức trong StringTokenizer này giúp chúng ta chia một câu thành các mã thông báo. StringTokenizer, mã hóa chuỗi trên cơ sở các dấu phân cách được cung cấp cho đối tượng lớp String tokenizer.
* Các dấu phân cách chung là khoảng trắng, tab, dòng mới, ký tự xuống dòng và nguồn cấp dữ liệu biểu mẫu. Các dấu phân cách này được coi là mặc định và nếu người dùng muốn cung cấp dấu phân cách riêng thì anh ta có thể cung cấp bằng cách xác định dấu phân cách trong tham số làm đối số.
* Nhớ phải import gói java.util.StringTokenizer; trước khi dùng StringTokenizer



* The Constructors of StringTokenizer:
  + Constructor has 1 parameter:
    - Syntax:

StringTokenizer object\_name = new StringTokenizer(“The string you want to separate”); => output [“The”, “string”, “you”, “want”, ”to”, “separate”]

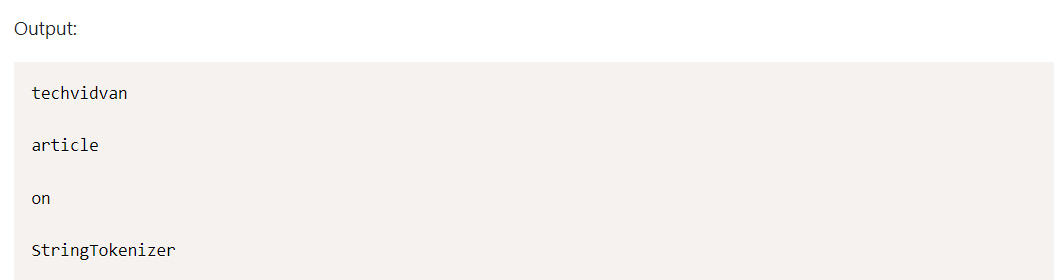
* + - Hàm tạo này được triển khai để thực hiện mã hóa một chuỗi cụ thể đang được cung cấp trong tham số. Hàm tạo này nhận tất cả các dấu phân cách mặc định đã được định nghĩa trong định nghĩa của lớp StringTokenizer.
    - Vì không có delim (parameter thứ 2) nên dấu phân cách mặc định là: Khoảng trắng, dòng mới, tab, dấu xuống dòng “\ r”, Nguồn cấp dòng “\ n” và nguồn cấp biểu mẫu “\ f”.
  + Constructor has 2 parameters which the second one is delim (dấu phân cách):
    - Syntax:

StringTokenizer object\_name = new StringTokenizer(“The-string-you-want-to-separate”, “-”); => output [“The”, “string”, “you”, “want”, ”to”, “separate”]

* + - Example:

A picture containing graphical user interface

Description automatically generated



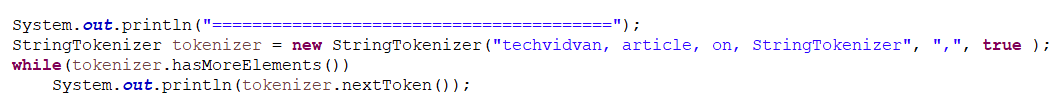
* + Constructor has 3 parameters which is the third parameter is flag has the datatype is boolean:
    - Syntax:

StringTokenizer object\_name = new StringTokenizer(“The string you want to separate”); => output [“The”, “string”, “you”, “want”, ”to”, “separate”, true]

* + - Ham tạo này được triển khai để thực hiện mã hóa chuỗi dựa trên dấu phân cách và có chức năng bổ sung để hiển thị dấu phân cách.
    - Với 3 parameter nếu là false thì nó sẽ không cho phép delim là một token, nhưng khi flag = true thì delim sẽ là một token
    - Example with flag = false

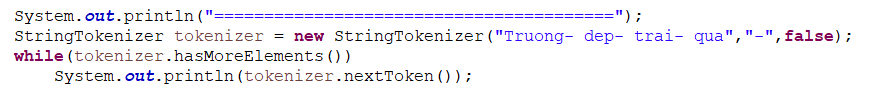
Graphical user interface, text, application

Description automatically generated



Text

Description automatically generated with medium confidence



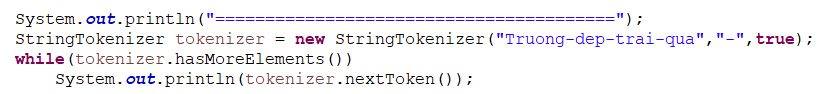
Text, table

Description automatically generated

* + - Example with flag = true

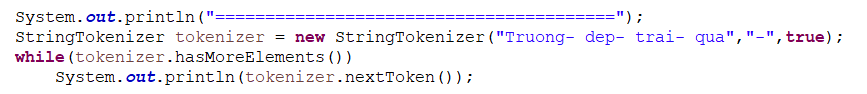
Graphical user interface, text, application, email

Description automatically generated



Text

Description automatically generated with medium confidence



Graphical user interface, application

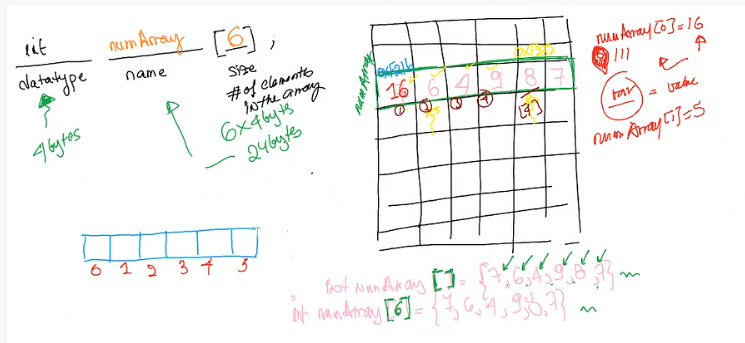
Description automatically generated with medium confidence

1. Split() method in class String

* Document: https://www.geeksforgeeks.org/split-string-java-examples/
* Split() is better than StringTokenizer because it can decide the separated element

1. Array:

* Initializing the size of array means reserving the space in memory
* Element of array is also a variable, and variable has the memory location



Text, whiteboard

Description automatically generated

1. Question:

* String[] args? And its example?