000000000000000000000000000000000



java

Personal notes



[Date]

[Company name]

[Company address]

# Java lesson 1

Public class welcome

ACCESS MODIFIER KEYWORD CLASS NAME

* Sum; variable
* Sum(); function

Function Header

* Return type
* access specifier
* function name
* Parameter list

The main method of the code is strict it can’t be changed.

Once you insert a class and make it the main method the rest of the public function header will appear automatically.

# Addition file

* System.out.prinbt(””)
* System.out.println()
* System.out.printlnf() -Format specifier
* Object,nextInt gets the data from the user. This comes from the Scanner import class

Scanner input = new Scanner(System.in);

* The new keyword creates an object.
* This whole line is called instantiate a Scanner object
* Using the input keyword we can use the methods in the scanner class
* Input is an instance of the class scanner
* Dot operator is how we call the object

# OOP

* Class use to create objects that posses the attributes and behavious defined in the class
* Method defines the program statement that actually performs a task
* A objects attributes define its properties, they are instance variables in class definition
* Instance variables are specific to an object, specified when the object is created
* Classes encapsulate (i.e. wrap) attributes and methods into objects
* Implementation details are hidden within the objects themselves. Information hiding is crucial to good software engineering
* Instantiation: we need to create an object of a class before a program can perform the tasks that the class’ methods define. The process of doing so is called instantiation. An object is referred to as an instance of its class.

## Working with classes

* Object-Oriented Analysis and Design (OOAD) is a method in software development for determining requirements and creating a design that meets them, using an iterative process and object-oriented programming languages like Java.
* Unified Modelling Language (UML) : the most widely used graphical scheme for modelling objectoriented systems
* A static method is special, because we can call it without first creating an object of the class in which the method is declared
* Method invocation(object name followed by dot followed by method name and parameter list
* Class attributes and methods are usually either public or private

## Declaring method with parameter

* A method will need addition information from the calling environment to perform its task in the form of parameters
* Parameters are defined in a comma-separated list enclosed within parentheses following the method name. Each parameter must specify the data type and variable name
* When a method is defined with parameters, the call to the method (i.e. method invocation) must supply an appropriate argument corresponding to each parameter in the method definition
* The number of arguments in a method call must match the number of parameters in the parameter list of the method declaration, and the argument type must be “consistent” with the corresponding parameter type

## Instance variabels, set Methods and get methods

* Local variables are declared within a method and can only be used there. They are lost once the method ends.
* Class attributes are declared as variables in a class declaration, and outside of the bodies of the class’ method declarations.
* When each object of a class maintains its own copy of a class attribute, that attribute is referred to as an instance variable
* Whenever a class has instance variables, it is customary to provide methods that will be used to modify or access the instance variables; these are referred to as set and get methods

## Initializing objects with Constructors

* Java requires each class to have a constructor, a special method that is used to instantiate (i.e. create an instance of) the class
* Keyword new is used when instantiating an object of a class by means of the class’ constructor; this amounts to requesting memory to be allocated for storing the instantiated object
* A constructor has the same name as the class, and has no return type
* By default, the compiler provides a default constructor with no parameters, which can be used to instantiate an object when no explicit constructor has been defined for the class.
* When a default constructor is used, instance variables get their default values. To set custom values, explicitly define a constructor for the class.
* Constructors are normally declared public (since they have to be used by other classes to instantiate objects)

## Class notes

* The class that defines the main method is called the driver class
* Instanve variable must be declared on the top of the class outside
* By declaring a variable inside a class it should be declared using the private access modifier
* For every private variable to access it we use
* Set method: modify and get method: read

# String character and Regular expression

## Literal vs objects

* Literal: an exact representation of a value; usually used for primitives (e.g. int, boolean, character, etc.)
* Character literal: an integer value represented as a character in single quotes
* String literal: a sequence of characters in double quotes, treated as a String object in Java; assigned to a variable via assignment operator (=)

## Class string

* String object: created via the “new” operator
* Java treats all String literals with the same contents as a single String object with many references to it; preserves memory
* Strings in Java are represented as objects
* String Constructors: create a String object, allocating new memory and returning a reference to it in the specified variable
* Observation: String objects in Java are immutable; i.e. once created and initialized, the character contents can’t be changed; thus copying an existing String object is unnecessary (memory misuse)

A screenshot of a computer code

Description automatically generated

* equals returns true if s1, s2 contents are the same; uses lexicographic
* == on String literals means the same as equals; different on String objects created with new operator
* Attempting to instantiate a String literal that already exists in a program does not create a new one
* regionMatches checks for equality of two substrings
* StringBuilder class provided for creating and manipulating modifiable strings

## String Buider

* Strings in Java are immutable (i.e. unmodifiable)

## Class character



## Regular expression

* A specially formatted String (can’t be StringBuilder) that describes a search pattern for matching characters in other Strings
* Consists of literal characters and special symbols

## Class notes

* To get the length of a string we will use the the .lenght() method

Int n = color.lenght(); =s

* We do not need an instance of an object inorder to invoke a class method
* Class methods are declared static and is defined for the class and can also be invoked without instantiating a class of the object
* When methods has the word is its checking is something is a specific form and it will return a Boolean
* Regular expression can not be used a string builder
* First name, starting with uppercase letter, atleast three characters, and only letters(all lower case except the first)

# Files , stream and object serialization

## Class file

* Provides information about files and directories

**First** with one String argument specifying name of file/directory to associate with File object:

* Name can contain (relative/absolute) path information, as well as file/directory name
* **Second** with two String arguments specifying absolute/relative path, and file/directory to associate with File object
* **Third** with File and String arguments, uses an existing File object that specifies the parent directory of the file/directory specified by String argument
* **Fourth** uses a URI (Uniform Resource Identifier) object to locate the file

**File/Directory path specifies its location on disk**

* Absolute path contains all directories, starting with root directory (e.g. C:\ in Windows), that lead to a specific file/directory
* Relative path normally starts from directory in which the application began executing, therefore “relative” to current directory
* Separator character for separating path files/directories: varies among OS’s; use File.separator to obtain local computer’s proper separator

Sequential Access Text files  
**Sequential-accces files**

* Java stores records in order by record-key field
* programmer needs to structure files to meet application requirements
* Lets say we use filereader we still need a method to read the file as a programmer you need to specify the type of date that you will be reading

**Creating a sequential-access text file**

* Formatter used to output formatted data to text-based stream
* **Open file** Formatter constructor with one String argument receives name of file, including its path; if path is not supplied, JVM assumes file to be in directory from which program was executed; file created if not existing already; if existing file is opened, its contents are truncated
* **Write date to file**
* **Close file**

**Exceptions**

* Are abnormal conditions that cause program to execute incorrectly
* You can think of it as a graceful failure mechanism
* Gracefully as it prevents a dangerous situation
* Programmes throw exception it will tell you why the program is not executing based of certain conditions
* Java requires program to implement exceptional programming inorder to have this graceful failure you can use the try-catch construct
* Try{} to catch the errors that will stop the code from running
* SecurityException: occurs when user has no permission to write data to file
* FileNotFoundException: occurs if file does not exist, and a new one can’t be created
* FormatterClosedException: occurs on attempt to output to file when Formatter has been closed

**Other**

try{

  read from file

}

catch{could not read

  FileSystem.exit(int)

}

* System.exit(int): static method that terminates an application; argument of 0 indicates successful program termination, nonzero value normally indicates an error (exception)

**PG 12 CODE**

int Number;

Formatter output; //almost the same as the one above

output = new Formatter()//if file does not exist we cant be created or exist but cannot write to diretory

* Due to those condition we have to use them in try and catch metods

## Reading Data from sequential-Access Text file

\* Data that was written to file in previous section is read back into a program to demonstrate sequential-access character-based stream input

* **Open file** Scanner constructor takes file path of the file to be read from
* **Read data from file**: Scanner object receives data file passed to it by constructor, which can then be used by the program
* **Close**

**File output exceptions**

* File not found (does not exist and cant be created)
* Here we can create a fileif it dioes not exist
* Write accrs not available (Security exeception
* File output object closed

**File input exception**

* File does not exist and no flexibility to create a file
* File input object is closed

## Obect Serilization

* Reading an entire object from file (or writing to file) is referred to as Object serialization
* Writing to file is referred to as serialization,
* reading from file is referred to as deserialization

**Serialized Object**

* FileOutputStream used in conjunction with ObjectOutputStream for serialization
* Inputing/outputting primitive type (+string)
* For inputting/outputting object data (object: primitive and reference type)



serilization derserilaztion



Ex reference type account record

Int accNumber

String firstName

String lastName

Double balance

* When we reading the file we must do the opposite its call derilisation
* Objetcs are byte based



* Input.next() this is for reading the next set of chaert

## Object deserilization Used

## Addition java.io Classes

## Class notes

* When ever we are trying to make achannel we must create a chanel.
* Either for iuputting data from the program or input data into the code
* We must create a channel of communication for data input/output
* Associate file object with source or destination
* We read from the source either in bytes or
* We write into the destination
* To specify the path it can be relative or absolute
* Must know if ur using your own class or classes that are already predefined
* Various operating system use different ways to specify a file separator
* Lets say we use filereader we still need a method to read the file as a programmer you need to specify the type of date that you will be reading
* Classes in the same package does not to be imported
* Character based file input  
  -Scanner reader  
  -Filereader
* Characterised file output
* Formatter
* Filewriter (Allows you append but you have instantiate to the file object that you want to write)