INFO 1103 Introduction to Programming
Notes Instructions(Publish version)

Format(To help you understand my notes better)

*: Not mentioned in this or before class, not necessary information.
Bold: specify a concept
Italic: emphasise
Text in Grey: Comments
Text in Red: Important notes
Text in Purple: Codes
Text in Yellow: Examples
Text in Green: A java program related

Author's reminder about this course

- Discussion board: Piazza(2016 S2), Ed(2017 S1)
- Textbooks
  - Book1: Savitch eBook: www.pearsonhighered.com/product?ISBN=0132772388
  - Book2: Horstmann eBook: http://au.wiley.com/WileyCDA/WileyTitle/productCd-EHEP002041.html

Study Strategies by author:

1. Have a careful read of lecture slides every week. They are the most important materials for this course. Textbooks are not compulsory but recommended.
2. Tutorial exercises are hard, but enough for you to understand Java. Do not hesitate to ask questions on everything. We have enthusiastic mentors(including me:) to help you.
3. And summary is necessary. But if you have my notes, it would be much more easy to understand different concepts. For example: What's the difference between instance variables and class type variables?

- If you purchased my notes:
  - Thank you very much. You have just made a wise decision. I do put a lot of effort in writing these notes, and they are nice, great, legend, awesome…
  - Do not hesitate to contact me and ask me for **OVER 50 Java programs**!!! related to my notes.
  - I am also willing to offer you help in your studies.

Information Technology is changing rapidly. I'll try to have my notes up to date. If you find a problem in my notes, tell me and maybe you will **get a reward**😊. And finally,

INFO 1103 Course Contents(2016 S2)

Week	Topic	Navigation
Week1	Introduction to Java	1. Computer Basics 2. Java Program fundamental 3. System.out.println 4. Error
Week2	2A:Variables and Expressions	1. Variables, data types and operators 2. Keyboard input and screen output 3. The class String 4. Documentation and style
	2B: Flow of control	1. The if-else statement 2. Boolean expressions and Operators 3. The switch statement
Week3	Loops	1. Loop Concepts 2. The while loop, do-while loop 3. The for loop 4. Programming with loops 5. Loop Errors
Week4	Methods	1. Method Basic 2. Method Comments 3. Parameter Passing 4. Return Value 5. Variable Scope
Week5	Array #1	1. Array Basics 2. Common array algorithms(Part 1) 3. Two-dimensional arrays
Week6	Array #2 & ArrayList	1. Common array algorithms(Part 2) 2. ArrayList
Week7	Class & Object #1	1. Object-Oriented Programming Features 2. Class & Object Concepts 3. Create a class 4. Use a class
Week8	Class & Object #2	1. Class Variables, Methods vs Instance Variables, Methods 2. The public and private modifiers 3. Constructor 4. Method Overloading
Week9	Array & ArrayLists of Objects, Inheritance and polymorphism	1. Arrays and ArrayLists of Objects 2. Inheritance 3. Polymorphism
Week11	Java Interface & Abstract class	1. Java interface type 2. The Comparable Interface 3. Abstract Class
Week12	Read and Write Files & Exception	1. Reading from files and writing to files 2. Processing Text 3. *Command line arguments 4. Exceptions and exception handling
Week13	Recursion	1. Recursion Concepts 2. Tracing Recursion Method

Please do not share this note to others without author's permission. © Copyright Reserved.

Interesting Java Programs written by myself (50 more programs!!!)

Week2:
Format.java(Helps to understand the output using different formatting in Java),
StringMethods.java(Discover those interesting methods. How can we do something with strings?)
Grader.java(A good example on if-else statement)
Weather.java(A good example on switch statement)

Week4:
isEven.java(This program tests whether the given number is even or not.)

Week5:
CommonArrayAlgorithm.java(One of the best! Explore how can we use array algorithms 1-5 to do something on arrays)
TypeArray.java(How can we type an array by ourselves.)
UseArrayInMethods.java(How can we put CommonArrayAlgorithm in methods.)
Traverse2DArray.java(Type a 2D array in console would be easy if you see this)
Additional: ReverseArray.java(I came up with a strange idea that if the array goes into reverse order…)

Week6(The best part!)
CommonArrayAlgorithm2.java, CommonArrayAlgorithmSearch.java, CommonArrayAlgorithmLoop.java(This time, I did something really **crazy** on arrays!)
ArrayList1.java(A java program that includes all the things you should know about arraylists.)
SortArray.java(How can we sort an array?)

Week7:
Account.java(A account class)
AccountTest.java(A program that tests Account class)

Week8:
Account.java(A account class but more complicated)
AccountTest.java(A program that tests Account class)

Week9:
Account.java(A account class but more complicated)
AccountTest.java(A program that tests Account class)
ChequeAccount.java, SavingsAccount.java(Good examples on subclasses.)

Week11:
A lot of programs on Java Interface and subclasses, abstract class and so on.

Week12:
A lot of programs on how to read input from files and give output.

Week13:
Countdown.java, CountDownFor.java, CountDownInfinite.java, CountDownTrace1.java, CountDownTrace2.java(These programs give a deep insight in recursive functions.)
Factorial.java, Power.java(Good examples on recursion.)

INFO 1103 Week 1
Introduction to Java

Navigation
- Computer basics
- Java program fundamental
- System.out.println
- Error

Computer Basics

- **High-level** languages are easy for people to understand and use, e.g. Java, C++, VB, Python
- **Low-level** languages are similar to machine language, e.g. the Assembly language
  - are symbolic forms of machine language that are easier for people to read.
  - require only minor additional translation before they can be run the computer.
- **Compilers**(used by C++) and **interpreters**(used by Matlab) are large *programs* that translate a program from a high-level language(source code) to low-level language(object code).
  - **Compilers** *translate the whole program. The program is then executed (run).*
  - **Interpreters** *translate each line of the program (or a block of lines) and execute it, then translate the next one and execute it.*
  - Compiled programs run faster than interpreters.
  - Java uses both a complier and interpreter.

Java Program Execution steps:
1. Write a Java program
2. **Compile**: Compiler translates a Java Program into an *intermediate* language, **bytecode**, which can be sent over the Internet and used anywhere in the world. "javac" command
3. **Java Virtual Machine(JVM)**: translates bytecode into *machine language* instructions. "java" command
4. Computer run these instructions.

Java bytecode runs on any computer that has a JVM. Every type of computer must have its own JVM. There is no need to recompile.

Two type of Java programs
- **applications**: regular programs run on the computer.
- **applets**: little applications run within a web browser.

Java Program fundamental

Eclipse is an Integrated Development Environment(**IDE**) for us to write and run the programs.

How the java program in eclipse runs:
1. Write a program in Editor in Eclipse
2. Save it to source file(src)
3. Compile by the complier
4. class files forms(include bytecode)
5. JVM
6. Run the program

Class:
1. All code in a program must go inside a **class**(a piece of software we can use in this program.)
2. **Class** name starts with capital letters.
3. The file name(e.g. HelloWorld.java) must be the same as the **class**(e.g. HelloWorld) name.

Method(Week 4):
Every time you call a method(e.g. System.out.println) in Java, you need to specify:
1. The **method** you want to use.
2. Any **values** the method needs to do its work. These **values** are called **arguments** and are enclosed in ( ) and separated by commas.
3. The compiler considers text in quotation marks as plain text not program instructions.
   - e.g. System.out.println("main"); The compiler will know that you meant the text main not the method main.

Code:
1. Codes go inside the main method.
2. A statement must end with a semicolon ;

System.out.println

- Do calculations:
  - System.out.println(3+4);
  - Output: 7
- Print a string:
  - System.out.println("Hello, World");
  - Output: Hello, World
- System.out.print doesn't start a new line.
- System.out.println("\nHello, World") start a new line before the text.

Error

1. Compile-time error(syntax error)
   - detected by complier
   - No class file is generated by the compiler
   - e.g. spelling, capitalisation, punctuation, matching of braces/parenthesis
   - e.g. System.out.println("Hello", "World!"); //not applicable for("string", "string")
2. Run-time error(logic error)
   - An error message from JVM.
   - A debugger(part of IDE) may help.
   - e.g. System.out.println("Hello, Wold!");
   - e.g. System.out.println(1 / 0); //can't divide by 0.