**Preliminary Information**

**Textbook**

Your textbook is: *Introduction to Java Programming, 7E.* The textbook website is:

<http://www.cs.armstrong.edu/liang/intro7e/studentsolution.html>

1. Note the location of the *Review Question Answers*.
2. Download the solutions for even numbered programming assignments. Located under *Programming Exercise Solutions*. After download, unzip into a convenient location. (If you are not familiar with *zip* files, you’ll need to get someone to show you how to unzip them. I believe windows comes with a utility to unzip files)
3. Download the code for all examples in the text. Located under *Download Source Code for Examples in the Book.* After download, unzip into a convenient location.
4. Take note of the *Common Errors*. It is organized by chapter. I would recommend reading this while we are covering each chapter.
5. We will learn the Java programming language in this class. We will use the *syntax* of the Java language to write *classes*. When you write classes you will often use *predefined classes* that do useful things. For instance, we will use the *Math* class when we want to raise a number to a power. A list of documentation of all the Java predefined classes is called the *Java API*, or just *API* for short (Application Programming Interface). It will be a while before we are ready to use the API, but for now, take note that the API can be found on the website under the link *Java API*.
6. There is an *Errata* section; however, there are no significant errors in the chapters we will cover for this class.

**Working at Home**

The Java *JRE* (Java Runtime Environment) is used to run Java programs and is usually already installed on your home computer. The Java JDK (Java Development Kit) is what allows you to write and compile Java programs. It is not on your computer. You must download this and install. You can write Java programs in any text editor. However, in this class, we will use *jGrasp*. This must be downloaded and installed. This software is installed on the computers in the CS lab, 2128 Nevins Hall. If you want to work at home, on your own computer, follow the steps below. Note, I will not be able to help you with this. The process should go smoothly. If not, seek help from a fellow student or *Google*.

1. **Download, install, and configure the JDK**

Read this document and follow the steps exactly; however, download **JDK 6, Update 7** instead of JDK 6. At one step, you will need to type something in. The document below will say to type in, “...jdk1.6.0...” You will type in, “...jdk1.6.0\_07...” The document does mention this, but I am telling you again for empahsis.

<http://www.cs.armstrong.edu/liang/intro7e/supplement/Supplement1bInstallingJDK6.pdf>

1. **Download and install jGrasp**

Go to: <http://www.jgrasp.org/> and choose: *Download*. Fill out the registration and then choose (most likely) the download that says: *jGRASP exe – Windows self-extracting exe.* Run the installation.

**Chapter 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Sections | Pages | Review Questions | Programming Exercises |
| 1.5, 1.8-1.10 | 8-12, 15-29 | 1.7-1.9, 1.14-1.25 | 1.1-1.2 |

**Section 1.5 – Number Systems**

1. Base 10 numbering system (decimal)
2. Use digits: 0, 1, ... , 9
3. $4862=4862\_{10}=4\*10^{3}+8\*10^{2}+6\*10^{1}+2\*10^{0}$
4. Base 2 numbering system (binary)
5. Use digits: 0 & 1
6. Example: $101101\_{2}$
7. Converting binary to decimal (base 2 to base 10)
8. $101101\_{2}=1\*2^{5}+0\*2^{4}+1\*2^{3}+1\*2^{2}+0\*2^{1}+1\*2^{0}=32+0+8+4+0+1=45$
9. $1000\_{2}=1\*2^{3}=8$
10. What is the largest number that can fit into a byte?
11. Count from 1 to 10 in binary.
12. Converting decimal to binary (base 10 to base 2)
13. $23\_{10}=x\_{2}, find x$
14. $813\_{10}=x\_{2}, find x$
15. An Algorithm:

DO Until val=0

 newVal = val / base (take integer part)

 rem = val % base

 print rem

 val = newVal

1. Base 16 numbering system (hexadecimal)
2. Digits

|  |  |  |
| --- | --- | --- |
| **Base 10** | **Base 16** | **Base 2** |
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

1. Examples: $60AF\_{16}, 12\_{16}, E001\_{16}$
2. Converting hexadecimal to decimal
3. $12\_{16}=1\*16^{1}+2\*16^{0}=16+2=18$
4. $B8\_{16}=11\*16^{1}+8\*16^{0}=176+8=184$
5. $60AF\_{16}=6\*16^{3}+0\*16^{2}+10\*16^{1}+15\*16^{0}=24,576+0+160+15=24,751$
6. Converting decimal to hexadecimal
7. $123\_{10}=x\_{16}, find x$
8. $1000\_{10}=x\_{16}, find $
9. Converting hexadecimal to binary
10. Convert each hexadecimal digit to its corresponding binary representation.
11. $3C\_{16}=x\_{2}, find x$
12. Converting binary to hexadecimal
13. Convert each set of four binary digits into the corresponding hexadecimal digit, starting from the right.hexadecimal digit to its corresponding binary representation.
14. $10011100111\_{2}=x\_{16}, find x$

**Sections 1.8 – A Java Program**

1. For now, every program we will write will have a structure like this:

public class MyClass

{

 public static void main( String[] args )

 {

 // Your code goes here

 }

}

The name of the file where we save this must be the name of the (public) class with the *.java* extension. For instance, the class above would be saved in the file: *MyClass.java*

1. Write *HelloWorld* program as a console application in class. Compile from command line and from jGrasp.

public class HelloWorld
{
 public static void main(String[] args)
 {
 // Display message to console.

 System.out.println( "Hello World" );

 /\* This is the most famous computer
 program and it is always the first \*/
 }
}

1. Terminology: class, class name, file name, statement, statement terminator, method, reserved words (keywords), line comment, block comment, class block, method block, nested block, case sensitive.

**Sections 1.9 – Compiling and Running a Java Program**

1. To run from the command line (a *console* application), first compile the java file:

javac HelloWorld.java

 This creates the *bytecode* file (the *class* file):

 HellowWorld.class

To run the program:

java HelloWorld

1. A few useful DOS commands for compiling and running from the command line (a *console* application).

|  |  |
| --- | --- |
| **DOS Command** | **Result** |
| dir | Shows a list of all files and folders in current folder |
| dir/p | Paginate listing of files and folders |
| dir \*.java | Display all java files in the current folder |
| dir \*. | Display all folders in the current folder |
| cd *folderName* | Move from current folder to the folder, *folderName* which is inside the current folder. You are moving *down into* the  *folderName* folder. |
| cd .. | Move *up to* the parent folder containing the current folder. |
| cd *f\** | Moves down to the first folder that begins with *f*. |
| f: | To change to the *F* drive, for instance. |

1. Example





















1. To run in jGrasp:



**Sections 1.10 – Writing a GUI Application**

1. Write *HelloWorld* program as a GUI (graphical user interface) application in class. Compile from command line and from jGrasp.

import javax.swing.JOptionPane;

public class HelloWorldGUI
{
 public static void main( String[] args )
 {
 JOptionPane.showMessageDialog( null, "Hello World" );
 }
}