CIST | 1400: Intro to Computer Programming
Exam 1: Review Sheet

Content Covered

- Lectures 1 – 6
- Chapters 1 – 4 in *Java How To Program (Late Objects Edition)* by Deitel and Deitel
    - Extra Help
        - See the *Summary*, *Terminology*, and *Review Questions* at the end of each chapter.

Quiz Format

- Time Limit
    - 75 minutes
- Questions
    - Multiple Choice
    - Short Answer
    - Definition
    - Short Coding
        - Write short code to accomplish a task
    - Code Analysis
        - Find errors in code

Chapter 1 Material

1. UNIX Commands
    a. Compiling Java programs
    b. Executing Java programs
    c. Editing a Java program
2. File naming conventions for Java programs
3. Differences between types of languages
    a. Machine language
    b. Assembly language
    c. High-level language

Chapter 2 Material

1. Basic program structure
2. Output
    a. `System.out.print()`
    b. `System.out.printf()`
    c. `System.out.println()`
    d. Using escape characters
3. Input
    a. Using a `Scanner` to get user input
4. Variables
    a. Declaring variables
    b. Assigning variables
5. Math
    a. Math operators (+, `-`, `*`, `/`, `%`)

    b. Representing normal math equations in Java
6. Relational operators (`==`, `!=`, `<`, `>`, `<=`, `>=`)
7. Comments (`//` and `/* */`)
8. Escape sequences

Chapter 3 Material

1. Control structures
2. `if` statements

Chapter 4 Material

1. Boolean logical operators (`&&`, `||`, and `!`)
    a. Precedence
2. `switch` selection structure
3. `char` data type

Material Breakdown

1. UNIX Commands
    a. Compiling Java programs
        i. `javac MyJavaProgram.java`
        ii. Compiles the high-level Java code to machine code
    b. Executing Java programs
        i. `java MyJavaProgram`
        ii. Note: Don't include the `.java` extension
        iii. Executes the Java program using the Java Virtual Machine
    c. Editing Java programs
        i. `vim MyJavaProgram.java`
        ii. Edits the Java program in the `vim` editor
2. File naming conventions for Java programs
    a. Bad file names
        i. Cannot include spaces or special characters
        ii. The underscore _ is the only exception
        iii. Cannot start with a number
    b. Acceptable file names
        i. Starts with an alphabetical character (a-z or A-Z)
        ii. Uses alphabetical characters
        iii. May include numbers in the filename, as long as it doesn't start with one
        iv. May contain underscores _
        v. May contain mixed casing (e.g. `MyJavaProgram.java`)
3. Differences between types of languages
    a. Machine language
        i. Code that the computer can understand and execute
        ii. When you run the command `javac MyJavaProgram.java`, it creates a file called `MyJavaProgram.class`, which is Java bytecode, or binary.
        iii. The computer can only understand and execute the code that exists in `MyJavaProgram.class`
        iv. Example of machine code
            1. `0011010101011110`
    b. Assembly language
        i. A step up from machine code
        ii. Used for low-level processing and programming

iii. Direct interaction with the hardware
iv. Example of assembly language
1. `ADD R1, R2, #3`
2. `SUB R1, R2, R3`

c. High-level language
i. Examples of high-level languages: Java, C, C++, C#, Python, …
ii. Used for high-level processing and programming
iii. Computers cannot understand high-level languages. They must be compiled into machine code

4. Basic program structure

```
a. (1) import java.util.Scanner;
b. (2) public class MyJavaProgram {
c. (3)     public static void main ( String[] args ) {
d. (4)         code
e.         }
f.     }
```

5. Output
a. `System.out.print()`
i. Prints out something without a newline at the end
b. `System.out.println()`
i. Prints out something with newline at the end
c. `System.out.printf()`
i. Same as `System.out.print()`, but includes the ability to use format specifiers
ii. `%d` : format specifier for an integer
iii. E.g. `System.out.printf ( "I am %d years old\n", 4 );`
d. Escape characters
i. Characters that are represented by a backslash, and a single character
ii. Escape characters include tab, newline, quotation marks, etc.
iii. `\"` – Quotation mark
iv. `\n` – Newline character
v. `\t` – Tab character
vi. `\\` - Literal backslash
vii. E.g. `System.out.println ( "1.\tHere is a quote: \"\n" );`

6. Input
a. Must import `java.util.Scanner;` above the class definition in your program
b. Must create a Scanner object in the main method
i. E.g. `Scanner in = new Scanner ( System.in );`
c. To get an integer from the user and store it into the variable `x`:
i. `int x = in.nextInt();`
d. To get two integers from the user and store it into variables `x`, `y`
i. `int x = in.nextInt();`
ii. `int y = in.nextInt();`

7. Variables
a. Declaring variables
i. Examples
1. `int x;`
2. `int myInt2;`
3. `int a, b, c;`
ii. When declaring variables like these, they do not contain any value
iii. Doing something like `System.out.println ( x )` gives a compilation error that would say something like "x may not be initialized"

b. Assigning variables
    i. Examples
        1. `int x;`
        2. `x = 5;`
        3. `int y = 91;`
        4. `int a = 3, b = 2, c = 1;`
        5. `char letter = 'a';`
c. You can declare AND initialize values simultaneously, like in examples (2) and (3), order declare first, and then assign the value, like in example (1)
d. Math
    i. Math operators
    ii. Addition, subtraction, multiplication, division, modulus
    iii. `+`, `-`, `*`, `/`, `%`
    iv. Modulus operator (`%`)
        1. Examples
            a. `int x = 5 % 2;` (`x` would contain the value of 1)
            b. `int y = 20 % 8;` (`y` would contain the value of 4)
        2. Used to calculate the remainder of division
        3. We can use modulus to check to see if a number is divisible by another number
            a. E.g.
```
if ( 20 % 5 == 0 ) {
    System.out.println ( "5 is a multiple of
        20!" );
}
```

a. Order of operations (same as regular algebra)
    i. `int x = 4 + 2 / 2;` (`x` would contain the value of 5)
b. Exponents
    i. Java doesn't have the caret operator available for performing exponents
        1. E.g. `int x = 4 ^ 2` does NOT calculate 4 raised to the power of 2
    ii. Instead, you must multiply the number out individually
    iii. E.g. 4^2 is equal to `int x = 4 * 4` in Java
    iv. E.g. 4^3 is equal to `int x = 4 * 4 * 4` in Java

8. Relational operators
    a. Operators that describe a relation between two numbers
    b. `==` : Equal to
    c. `!=` : Not equal to
    d. `>` : Greater than
    e. `<` : Less than
    f. `>=` : Greater than or equal to
    g. `<=` : Less than or equal to
    h. E.g. `if ( x >= 8 )` …
9. Comments
    a. Comments are used to describe a what a block of code does
    b. Comments are not interpreted by the compiler
    c. Single line (or end-of-line) comments
        i. E.g. `// This is a single-line comment`
    d. Multi-line (or block) comments
        i. E.g.
```
/* This is a block comment.
   It can span…
   multiple lines! */
```

10. **if** statements
    a. **if** statements are one of many control structures in Java
        i. They control the flow of a program
    b. Single **if** statement
        i.
```
if ( x == 5 ) {
    System.out.println ( "x is equal to 5!" );
}
```
    c. Single **if/else** statement
        i.
```
if ( x == 5 ) {
    System.out.println ( "x is equal to 5!" );
}
else {
    System.out.println ( "x is not equal to 5!" );
}
```
    d. Nested **if** statements
        i.
```
if ( x >= 5 ) {
    if ( x <= 10 ) {
        System.out.println ( "x is between 5 and 10." );
    }
}
```
    e. Nested **if/else** statements
        i.
```
if ( x >= 90 ) {
    System.out.println ( "You got an A!" );
}
else if ( x >= 80 ) {
    System.out.println ( "You got a B!" );
}
else if ( x >= 70 ) {
    System.out.println ( "You got a C!" );
}
else {
    System.out.println ( "You got a D or lower!" );
}
```
    f. If there is only one action associated with the **if** statement, curly braces are optional
        i.
```
if ( x >= 90 )
    System.out.println ( "You got an A!" );
else if ( x >= 90 )
    System.out.println ( "you got a B!" );
```
    g. Note: The **else** case is not required

11. **switch** selection structure
    a. General
        i. Format
        ii. Data types it can compare
            1. **int**, **char**, **String**
            2. Anything else – error
    b. Semantics
        i. Use of **case**
            1. Use of **case** with/without **break** statement
        ii. Use of **default**
    c. Ability to convert **switch** to **if** statement and vice versa

12. Logical operators
    a. `&&`, `||`, `!`
    b. Precedence
13. Math operators
    a. `()`, `*`, `/`, `%`, `+`, `-`
    b. Precedence

Sample Questions

1. **For the potential variable names listed below, circle VALID if it is a valid variable name, or INVALID if it is not. For any that are INVALID, list the reason as to why it's not valid.**

   `2012IsntReal` VALID INVALID : _____

   `WeAre#1` VALID INVALID : _____

   `iAmArobot2` VALID INVALID : _____

   `variable 1` VALID INVALID : _____

---

2. **Write an output statement that would print the following text exactly as shown below on one line, leaving the cursor on the next line.**

```
"Baby, you're a firework" - Katy Perry
```

3. **Using correct operator precedence, what values would be printed by the following output statements?**

```
System.out.printf ( "%d", 42 * ( 2 / 4 ) + 18 % 4 - 12 * 2 );

System.out.printf ( "%d", 42 * 2 / 4 + 18 % 4 - 12 * 2 );
```

4. **Write the code necessary to accomplish the following. Declare any variables you use. You do not need to write a complete Java program.**

   a. Create an `int` called `number` and set it equal to 1.
   b. Use 3 output statements to produce the output below.
   c. You may only change the value of number by 1 after each output line.
   d. You may not output a value directly; i.e. `number` must be used for each value output.
   e. The output statements should include tabs to output the numbers in columns.

   Output to create:

   1 5 9
   2 6 10
   3 7 11

5. **Write code to ask for an integer called `n` and then uses `n` to evaluate the following algebraic equation. The program should display both the number read and the result of the equation. Declare any variables you use. You do not need to write a complete Java program, just enough to accomplish the requested task.**

   Equation to use:

   $$2n^4 - 10n^3 + 17$$

   Sample output (user input is underlined):

```
Enter a number: 2
Number was 2
Answer is -15
```

6. **What is the output of the following program?**

```
public class Question {
    public static void main ( String args[] ) {
        int foo;
        System.out.println ( foo );
    }
}
```

7. **What will be the output after the following Java statements have been executed?**

```
int a = 4, b = 12, c = 37, d = 51;

if ( a < b )
    System.out.println ( "a < b" );
if ( a > b )
    System.out.println ( "a > b" );
if ( d <= c )
    System.out.println ( "d <= c" );
if ( c != d )
    System.out.println ( "c != d" );
```

   a. a < b
   c != d

   b. a < b
   d <= c
   c != d

c. a > b
c != d

d. a < b
c < d
a != b

8. **Given the following segment of Java code, what is the output?**

CODE:

```
int x = -12, y = 81, z = 2112;

if ( ( x == y ) || ! ( y < z ) || ( x == z) )
{
    System.out.print ( "Heads" );
}
else
{
    System.out.print ( "Tails" );
}
```

OUTPUT: _____

Using the complete words `True` and `False` and the values for variables `x`, `y` and `z` given above, show every individual internal step required for Java to evaluate the expression in the `if` statement above, resulting with a final value of `True` or `False`. Clearly provide the result of each relational or logical operation. Rewrite the expression each time, like the following example:

Example answer:

```
x != y && y != z && x == z

 True   &&  True   &&  False

      True        &&  False

              False
```

Problem answer:

```
( ( x == y ) || ! ( y < z ) || ( x == z ) )
```

9. **Write the code necessary to accomplish the following tasks. You do not need to write a complete Java program.**
   - **a.** Create a `char` variable called `hero`
   - **b.** Ask the user to input a character and store it into `hero`
   - **c.** Use a `switch` control structure to perform the following tasks:
   - **d.** If the letter is a `B` (uppercase or lowercase), output `BATMAN IS BEST`
   - **e.** If the letter is an `S` (uppercase or lowercase), output `SUPERMAN IS BEST`
   - **f.** If the letter is a `W` (uppercase or lowercase), output `BLACK WIDOW IS BEST`
   - **g.** Otherwise, output `ALAS ALL THE SUPER HEROES ARE GONE`

CODE:

10. An uninitialized local variable contains:

a. The value last stored in the memory location reserved for that variable.
b. No value.
c. A value of zero.
d. A randomly assigned value.