CMPE212 – Reminders

- Assignment 1 due Friday.
- Quiz 1 Next Week, in the lab on Thursday. Everything up to and including next Monday’s lecture material. Topics will be listed later this week. Make sure to complete “Quiz 0” before writing Quiz 1.

Today

- Loops
- Bad code...
- Multi-dimensional Arrays.
- Style Guidelines for assignment 1 (if we have time).

Repetition or Using “Loops”

- Java has:
  - while
  - do/while
  - for
  - The “for each” loop

- Also, discuss the use of break and continue

“while” loop - Cont.

- while loop syntax:

```java
while ( boolean_expression ) {
    block_of_code
}
```

- As long as boolean_expression evaluates to true the statements in the block_of_code continue to execute.
- One statement inside the loop does not need { }.
- By mistake, you might write the following - what would happen?

```java
while ( boolean_expression );
```

“do/while” loop

- Syntax:

```java
do {
    block_of_code
} while ( boolean_expression );
```

- Note the “;” at the end of the while statement.
- Since the conditional test is at the end of the loop, it will always execute the loop at least once.

“for” loop

- The kind of while loop shown below:

```java
int i = 1;
while (i < 21) {
    // other statements
    i = i + 1;
}
```

is used so often, that Java has provided another looping structure that does all that is shown above, but needs only one line:

```java
for (int i = 1; i < 21; i = i + 1) {
    // other statements
    OF I++
}
```
• Syntax:

```java
for (initialization; boolean_expression; update) {
    block_of_code
}
```

• `for` loops are used when you know, in advance, the number of repetitions desired.

• If there is only one statement inside the loop you don’t need the `{ }` brackets.

• You don’t have to declare the counter inside the `for` loop, if you have declared it earlier in your program.

• But if you do declare it in the `for` statement then the scope of that variable will only be inside the loop block.

• Often, you will want to visit every element in a collection, not just a part.

• Syntax of the “for each” loop:

```java
for (type variable : collection) {
    // statements
}
```

• For example, suppose we have an array called `data`, containing a collection of `double` type numbers, and you want to add them all up:

```java
double sum = 0;
for (double e : data)
    sum = sum + e;
```

• Equivalent normal `for` loop:

```java
double sum = 0;
for (int i = 0; i < data.length; i++)
    sum = sum + data[i];
```

• The for each loop is a bit easier with arrays, but is even better suited for other kinds of collections.

• A for each loop does not allow you to change elements in the collection. Use `[]` and an index value instead in the LHS of an assignment.
Loops - Misc.

• A for each loop can use var for the element type.

• Don’t declare variables inside loops, as the repeated declaration process uses up time and memory unnecessarily.

• There is no limit in Java to how many levels you can nest loops.

• It is customary, but not necessary, to use the variables i, j, k as loop counters, when the counter has no intrinsic meaning.

Other Java Keywords Used With Loops

• break and continue

• The continue statement interrupts the execution of a loop, and returns control to the top of the loop.

• The break statement interrupts the execution of a loop, and transfers control to the first statement after the loop.

Use of continue

• For example:
   ```java
   for (i = 1; i <= 5; i++) {
       if ( i == 3 ) continue;
       System.out.println("i = " + i);
   }
   System.out.println("End of Loop!");
   ```

• Displays:
  
<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>End of Loop!</td>
</tr>
</tbody>
</table>

Use of break

• For example:
  ```java
  for (i = 1; i <= 5; i++) {
      if ( i == 3 ) break;
      System.out.println("i = " + i);
  }
  System.out.println("End of Loop!");
  ```

• Displays:
  
<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>End of Loop!</td>
</tr>
</tbody>
</table>

Use of “break” & “continue”

• Only use these keywords when it makes your code easier to read.

• Avoid the use of more than one break or continue inside a loop.

• If you use a simple condition to issue a break statement, then why can’t you put that condition in the loop test?

• Overuse of break statements can lead to “spaghetti” code - just like the use of “goto” statements!

Bad Code 1

• What is “wrong” with this code? (It compiles fine.)

```java
boolean continueFlag = true;
String userInput;
while (continueFlag != false) {
    userInput = IOHelper.getString("Enter "yes" to stop loop: ");
    continueFlag = !userInput.equals("yes");
}
```
Bad Code 2

• What is “wrong” with this kind of loop structure?:

```java
int i = 0;
while (true) {
    System.out.println("i = " + i);
    if (i >= 20)
        break;
    i = i + 1;
}
```

Bad Code 3

• What is “wrong” with this structure?:

```java
String userInput;
for (int i = 0; i < 100; i++) {
    System.out.println("i = " + i);
    userInput = IOHelper.getString("Enter " + i + " to stop loop: ");
    if (userInput.equals("yes"))
        break;
}
```

Bad Code 4

• What is “wrong” with this technique? It works:

```java
public static long factorial (long factor) {
    if (factor <= 1)
        return 1;
    return factor * factorial(factor - 1);
}
```

Bad Code 4, Cont.

• Never use recursion when you end up with a single recursive call, that is usually at or near the end of the method.

• Called “tail recursion”, and,

• It is much more efficient to use a loop instead!

• A Quicksort method is a good example of an efficient and worthwhile use of recursion.

Introduction to Multi-Dimensional Arrays

• We have discussed the declaration of 1D arrays. For example:

```java
double[] oneD = new double[100];
```

• To assign values, for example:

```java
for (int i = 0; i < oneD.length; i++)
    oneD[i] = i;
```

Multi-Dimensional Arrays, Cont.

• For each additional dimension, just add another set of [ ]:

```java
int[][] twoD = new int[4][20];
```

• Has room for 80 values. To assign, for example:

```java
int row, col;
for (row = 0; row < twoD.length; row++)
    for (col = 0; col < twoD[row].length; col++)
        twoD[row][col] = row * col;
```
Multi-Dimensional Arrays, Cont.

• It is helpful to think of a 2D array as storing tabular data, like a spreadsheet.

• For convenience (and to help you code consistently) you can think of the first dimension as the rows and the second dimension as the columns.

• To get the number of rows, use twoD.length, and to get the number of columns, grab one of the rows and use twoD[row].length.

Multi-Dimensional Arrays, Cont.

• You can use three sets of [] to get a 3 dimensional array. Using the spreadsheet analogy, the third dimension could be the sheet number, where each sheet contains a table.

• For a good analogy for a 4D array you might need to look up a Star Trek script...

Iterating 2D Arrays

• See TestLoops.java

• Can a for/each loop be used to assign or alter values in an array?

Assignment 1 Style Hints

• Let Eclipse help you with style! It will “suggest” indentation and spacing – let it!
• Separate methods with a blank line to make them stand out.
• Minimize the use of blank lines inside methods.
• Follow the {} alignment conventions discussed in class.
• Put spaces on either side of binary operators.
• Do not put spaces before , or ;
• Put a single space after , in a parameter list and after ; in a for loop.

Assignment 1 Style Hints, Cont.

• There is never any need for more than one empty line at a time.

• Once a line or comment is indented you don’t have any need for multiple consecutive spaces in the rest of the line.

• If your code lines are shorter, you can choose to use in-line comments lined up at the RHS of your code, but:

• If your lines of code are longer, then use comments above and lined up with the code they are commenting.

Assignment 1 Style Hints, Cont.

• Use descriptive method, attribute and variable names.
• Usually a name can have one or two nouns or a verb/noun combination, for a method.
• Don’t get carried away! Be concise, yet descriptive.
• Use camelCase or the underscore, _, but not both.

• Remember to capitalize constant attribute names. You can use _ in these names...
Assignment 1 Commenting

- Place a block comment at the top of your program with your name and SN. Also, in general terms describe what your program does (a few sentences only).
- Use one or two sentences before each method to describe what it does and any assumptions made. Indicate what the parameters are.
- Use in-line comments to specify units, where needed.
- A few comments inside methods might help to describe what is going on – don’t get carried away!

More Style Hints

- Leaving commented out code in your program is really irritating to someone trying to read your program – it makes it look unfinished.
- If your program is not finished – go ahead and admit this in a comment above the method or class. Note what you think the problem is. Your TA is going to find out anyways if the code is broken – so you might as well just come clean and save him some effort!

Style Demo Programs

- See:
  - NoStyle.java
  - PoorStyle.java
  - GoodStyle.java
- They all are syntactically correct, will compile without error and all do the same thing.