CSCI 1301 Lecture 9

What we will cover:
Iteration
The while Loop
Number Guessing

Class Exercise 11 – Tedium.java

Iteration
Iterative or looping structures allow programs to programmatically repeat instructions. Java has three looping statements: while, do-while, and for. These structures are typically interchangeable, but each specific type of loop lends itself to certain types of problems. Common uses of loops include: batch operations on sets of data, counting loops (think summations), anything repetitive, etc.

The while Loop
The while loop allows instructions to be executed repeatedly until a condition evaluates to false. This is what is known as an event-controlled loop; it performs an indefinite number of iterations and terminates when an event has been satisfied. The general form of the while loop is:

```
initialize control objects;
while (loop-continuation-condition) {
    statements;
    update control objects;
}
```

The loop-continuation-condition, or loop test, is a logical expression tested at the start of each loop iteration. If the test evaluates to true, then another iteration, an execution of the loop body statements, is performed. After the loop body statements have been executed the program returns and performs the loop test again. The loop is repeated until the loop test is false. The loop then terminates and the program goes to the next statement after the loop body’s ending brace.

The control objects used in the loop test must be initialized before starting the while loop. The control objects used in the loop test should be updated during the loop body otherwise you may get an infinite loop. For while loops, it is possible to never execute the loop body if the loop test never evaluates to true.
We could count from 0 to 100 with a while loop with code like this:

```java
int counter;
counter = 0;
while (counter < 100) {
    System.out.println(counter);
    counter++;
}
```

We initialize a counter, use it to test a condition, and then update that counter as we execute the loop of the body. If we added a print statement in this loop, we would see that the variable is incremented by one on every pass of the loop. These types of loops, for hopefully obvious reasons, are called counter-controlled loops.

What is wrong with the following example (hint: look at the counter sequence)?

```java
int counter;
counter = 0;
while (counter < 100) {
    System.out.println(counter);
    counter--;
}
```

The value of `counter`, as this loop executes, is: 0, -1, -2, ... . This is an example of an infinite loop. The `counter--;` statement should be `counter++`. The loop body must update the loop control objects otherwise the loop will never terminate (there must be a way for the loop test to fail).

Class Exercise 12 – Modifying Tedium.java

**Number Guessing**

We’ll now write a program that continuously asks a user to guess the randomly generated number while giving them clues as to the accuracy of their entry. The program stops once the user guesses the answer.

We now introduce the ability to “randomly” generate a number. Java’s Math class has a method, `random(...)` that returns a double value greater than 0.0 and less than 1.0. Because of this, the result of `Math.random()` is usually multiplied by some multiple of 10 to get a whole number with a certain range. It is usually also cast into an integer when wanting to work with whole numbers.

In addition, we’ll be using the `Scanner` class a lot more as our output and input will get a little more complicated. Refresh yourself on the use of this class.

```java
import java.util.Scanner;

public class GuessNumber {
    public static void main(String[] args) {
        int guess = -1, answer = 0;
        String message = "";
        Scanner scanner = new Scanner(System.in);

        answer = (int) (Math.random() * 100);

        // until the guess equals the answer, keep asking for a guess
        while (guess != answer) {
            System.out.print("Please enter a number between 0 - 100: ");
            guess = scanner.nextInt();
        }
    }
}
```
message = "Your guess is ";

if (guess > answer) {
    message += "high";
} else if (guess == answer) {
    message += "correct!!";
} else {
    message += "low";
}

System.out.println(message);