Mr G's Java Jive

#15: Loops and Counting

Sometimes you want or need to do something over and over again. That's what this handout is all about.

An Old Bad Joke

My father used to tell me this one when I was a kid: Pete and Repete were sitting on a fence. Pete fell off. Who was left?

Pete and Repete were sitting on a fence. Pete fell off. Who was left?

Pete and Repete were sitting on a fence. Pete fell off, Who was left?

After about the third iteration, even a five-year-old knew enough to say "the other guy," but what we're looking at here is the basic structure of a **loop**, something that happens over and over again while a certain condition is **true.**

The while Loop

Java has quite a few ways of doing loops, and some are better suited to certain tasks than others. That being said, we're going to make things really simple here by only dealing with one kind of loop that we can twist to our own purposes if we're smart enough: the **while** loop.

A **while** statement is a lot like an **if** statement. If you remember that an **if** statement did something **if** the condition was true, then it will make sense that a **while** statement does something **while** the condition is true. The assumption here is that the **if** does it only **once** and then moves on, while the **while** statement **continues to do it** as long as the condition is true. To see how this works, write and run the little program I've shown below.

```
//StupidLoop
//a program to run a very stupid and annoying loop
//8.14.06 Clark Kent

public class StupidLoop
{//start class
   public static void main()
   {//start main
      //variables
        char ans = 'z';

      //run the loop
        while(ans!='d')
        {//start while
           System.out.print("Bet you can't figure out which key to hit ");
           System.out.print("to end the program! ");
           ans=gatling.getChar();
        }//end while

      //quit the program
        System.out.println("\nOh wow. You finally figured it out. See ya!");
   }//end main
}//end class
```

StupidLoop Explained

The condition of the loop is that it will run as long as the character variable **ans** doesn't equal the lowercase letter **d**. So each time the user enters something that's not a lowercase **d**, the condition is still true, and the loop continues to run. Enter a lowercase **d**, however, and the game's over. Now let's try something a little more sophisticated and useful.

Counting

When you really think about it, all counting is a form of a **while** loop. If I tell you to count from 1 to 20, what I'm actually saying is to start at 1, and while that number is less than or equal to 20, to say that number and then move up to the next one. In other words:

```
int num=1;

while (num<=20)
{//start loop
   System.out.println(num);
   num++;
}//end loop
```

See how simple that was? Probably even a little too simple. I might have insulted your intelligence with that one. Fine. That means we're ready to try something a little more interesting. How about a program that asks the user for two numbers and counts from one to the other? Try the code shown below and run it.

```
//Counter
//a program to count from a low number given by the user to a high number
//  also given by the user
// 8.14.06 Clark Kent

public class Counter
{//start class
   public static void main()
   {//start main
      //variables
        String name;
        int lo, hi;

      //get input
        System.out.println("Hi, welcome to my counting program.");
        System.out.print("What's your name? ");
        name=gatling.getLine();
        System.out.print("\nOK, "+name+", what number do you want to start at?" );
        lo=gatling.getInt();
        System.out.print("\nThanks. And how high do you want to count? ");
        hi=Mine.getHiInt(lo);

      //count
        while(lo<=hi)
        {//start loop
           System.out.println(lo);
           lo++;
        }//end loop

      //end the program
        System.out.println("\nThanks for playing. See ya.");
   }//end main
}//end class
```

Did you notice how I used our **getHiInt** method to prevent users from being idiots when it came time to enter the second number?

Keeping Track of Things

If we can count, then we can count **things**. And if we can count **things**, then we can keep track of things. Things like maybe **factors**. As you should know, the factors of a number are those whole numbers, from 1 up to and including itself, that go evenly into it. Here's a program that asks for a number, checks to see which numbers leading up to it are its factors, and then counts the ones that are. It's admittedly long, but I wanted it to display the entire "thought process" to the screen. Type it in and run it.

```
//Factors
//a program to show the factors of a given number
//8.14.06 Clark Kent

public class Factors
{//start class
   public static void main()
   {//start main
      //variables
        String name;
        int num;         //the user gives us this one
        int testnum = 1; //the first number we start with
        int factnum = 0; //the number of known factors so far

      //get input
        System.out.print("Hi. What's your name? ");
        name=gatling.getLine();
        System.out.println("\nWelcome to my Factors program, "+name+".");
        System.out.println("Give it a whole number, and we'll check for factors. ");
        num=Mine.getHiInt(0);
        System.out.println();

      //do stuff
        while(testnum<=num)
        {//start loop
           System.out.print("\tIs "+testnum+" a factor? ");
           System.out.print(num%testnum==0);  //display the boolean result
           if (num%testnum==0)  //if it's divisible...
           {//start if
              factnum++;        //increase the number of known factors by one
              System.out.print("  "+factnum+" factors!");
           }//end if
           System.out.println();
           testnum++;          //move up to the next test number

        }//end loop

      //end program
        System.out.println("\nSee ya!");
   }//end main
}//end class
```

The line that says **System.out.print(num%testnum==0)** is worth a little special attention here. This is printing the result of a **boolean** statement, which as you recall, is only ever **true** or **false**. So if the statement is true, it'll display that to screen, and if it's false, it'll display that as well. It was important to me that the program do this so that you could see just which numbers were factors.

By the way, did you notice the **if** statement inside of the **while** loop? You can do this. You can put **ifs** in **whiles**, **whiles** in **ifs**, **ifs** in **ifs**, and **whiles** in **whiles**. You have to be careful and make sure that all of your **curly brackets** match up (and labeling them is always a good idea), but it can be done.

Now, I'll admit that this is probably way more on-screen information than you need. So after you've run this a few times on different numbers, let's take a look at cutting this down a bit on the next page.

Cutting Factors Down To Size

If you don't need that much information, if you only want to know what the factors of a number are, and how many, then there's a much simpler way to do this. Replace the entire **do stuff** section you have right now with the much simpler one shown below:

```
//do stuff
  while(testnum<=num)
  {//start loop
     if (num%testnum==0)
     {//start if
        factnum++;    //increase the number of known factors by one
        System.out.println("\t"+testnum);
     }//end if
     testnum++;       //move up to the next test number
  }//end loop

  System.out.println("\n"+factnum+" total factors.");
```

Now compile and run it.

The Bare Minimalist Version

But suppose all you want to know is **how many** factors a number has, without needing to see what they are? Well, we can cut this down to size a little more by replacing the code in your **do stuff** section with the even simpler one shown below:

```
//do stuff
  while(testnum<=num)
  {//start loop
     if (num%testnum==0)
       {factnum++;}   //increase the number of known factors by one
     testnum++;       //move up to the next test number
  }//end loop

  System.out.println("\n"+num+" has "+factnum+" total factors.");
```

Compile and run it.

Other Little Counting Games

There are many other ways we can use **while** loops to count, but there are a few other issues we have to deal with first. One is that of wasted screen space, and the other is finally being able to run a program over and over again without having to **right-click** on the **object** every time.

Interestingly enough, that second problem can be solved with a **while** loop – and will, in handout #16.