Mr G's Java Jive

#16: The Big Loop

So far, whenever we've wanted to run a program more than once, we've had to go back to the project window, right-click on the program object, and tell it to run the main method again. That was a huge pain in the tush, so here we'll create a new class called **BigLoop** that you'll use as the basis for all future program.

The Basic Skeleton

The code shown below is the bare skeleton of BigLoop. If you've typed everything in correctly, all it should really do when you compile and run it is to say "We're finally done here. See ya!"

Check it out.

```
//BigLoop
//a program to run itself over and over again
//1.2.07 Clark Kent

import java.text.*;  //needed for number formats

public class BigLoop
{//start class
   //======== parking space 1

   //======== parking space 2

   //======== parking space 3

   //======== main parking space
     public static void main()
     {//start main
        //variables

        //big loop

        //ending
          System.out.println("\nWe're finally done here. See ya!");

     }//end main
}//end class
```

Big Deal!

So the program tells you that it's finally done. Big deal! Finally done doing what?

Well that word **finally** is what this is all about, because BigLoop assumes that you're going to be doing <u>something</u> over and over again until you've <u>finally</u> decided that you've had enough of it. In order to make that true, make the changes shown below inside of **void main**, then compile and run.

```
//variables
  char ans='y';

//big loop
  while (ans=='y')
  {//start while
     //insert method call below

     //ask to do over
       System.out.print("\nRun again (y/n)? ");
       ans=gatling.getAns();
  }//end while
```

Something New Has Been Added!

Now, when you run the program, not only should it keep asking you if you want to run it again, but it also demands an answer of either **y** or **n** (or **Y** or **N**) from you, and will not let you go until you give it a valid response. This is because of the **gatling.getAns** method that's used in the while loop. I wrote this method as a way of idiot-proofing so that the user could only enter a **y** or **n** response. If you want to see how the method works, check out code in the **gatling** class.

The **gatling.getAns** method is really quite polite and tolerant of idiots. It will let a user put in bad data indefinitely, and will keep prompting them to try again. But if you don't want your program to be quite that patient, you might try using **gatling.getAnsSurly**. This method only gives users three chances to put in valid data, and then insults them if they blow it.

But despite all this neat stuff, our program really isn't doing anything yet. We'll change that right now by making another new project and pasting the guts of this one into it.

Using BigLoop

The whole point of BigLoop is that it's the basis for other programs that you'll write in the future. So rather than adding anything else specific to it, we're going to leave it just as it is forever, but always copy its guts into whatever new thing we're going to do. With that in mind, create a new project called **RootLoop** and past the guts of BigLoop into it.

Obviously (at least I hope it's obvious to you), you'll have to change the name **inside** of the program, along with some of the comments. So make the changes I've marked below.

```
//~~BigLoop~~RootLoop
//~~a program to run itself over and over again~~a program to repeatedly get square roots
//1.2.07 Clark Kent

import java.text.*;  //needed for number formats

public class ~~BigLoop~~RootLoop
{//start class
   //======== parking space 1

   //======== parking space 2

   //======== parking space 3

   //======== main parking space
     public static void main()
     {//start main
        //variables
          char ans='y';

        //big loop
          while (ans=='y')
          {//start while
             //insert method call below

             //ask to do over
               System.out.print("\nRun again (y/n)? ");
               ans=gatling.getAns();
          }//end while

        //ending
          System.out.println("\nWe're finally done here. See ya!");

     }//end main
}//end class
```

If you've done everything right here, then when you compile it should run exactly as it did before.

Making It Specific

Inside of the while loop in **void main** is a comment line that says **insert method call below**. The method we call will be the one that gets run over and over again until we decide we're tired of it. Problem is that we don't have a method to run yet, so there's nothing to insert "below." We'll take care of that by writing a method called **rootmain** in the first parking space. Here's what it should look like:

```
//======== parking space 1
  public static void rootmain()
  {//start rootmain
     //variables
       double num, root;
       String nstring, rstring;

     //number formats
       DecimalFormat d3o = new DecimalFormat("0.###");

     //get input
       System.out.print("\n\tWhat number would you like the square root of? ");
       num=gatling.getDouble();

     //do calculations
       root=Math.sqrt(num);
       nstring=d3o.format(num);
       rstring=d3o.format(root);

     //show results
       System.out.println("\tThe square root of "+nstring+" is "+rstring+".");

  }//end rootmain
```

Compile this just to make sure there are no errors.

Having Isn't Enough

If you tried running the program after the last changes, you probably noticed that it didn't do anything different. It didn't ask you for numbers to show the square roots of. That's because having isn't enough. It's not enough to just have the **rootmain** class. You also have to use it. Using it is the final step we need to take in order to make **RootLoop** into a fully-functional repeating program. And after all we've done so far, using it is probably about the easiest thing there is to do.

Remember the comment in the while loop inside of **void main** that said **insert method call below**? Well now it's time to insert that method call. Take a look at the change shown below, put it in your own program, and then compile and run.

```
//big loop
  while (ans=='y')
  {//start while
     //insert method call below
       rootmain();

     //ask to do over
       System.out.print("\nRun again (y/n)? ");
       ans=gatling.getAns();
  }//end while
```

Dessert

This was a piece of cake, right? If you had any problem with the last step, it was probably because you forgot to put the parentheses with the method name. A lot of people forget that the parentheses are always part of the method call (especially when the method needs to bring something in).

More to Come

That's about it for the whole BigLoop thing. I hope you see how this makes running your programs a whole lot easier. Next up: random numbers.