CIS 201 Fall 2008: Lab 9
Bouncing Bumpers

- Reading a file

Consider the process of writing a pinball game. Given a screen and a ball that bounces, how do different levels or pinball games differ? They differ in how their bumpers (and other things) are laid out.

What we really want, then, is to build a pinball *engine*, a program that can simulate any pinball machine described in an appropriate way. You will begin with a simulator that has ball physics but no bumpers. Your job is to extend the program so that it prompts the user for the name of a bumper file (format described below) and then creates the appropriate sprites in the machine before setting the ball(s) loose.

Phase 1. Download the files `Pinball.java` and `PinballGame.java` into a directory for `Lab9`. The two files are linked through Moodle (in the description for the Lab where you will turn it in) and are located at `http://cs.potsdam.edu/facult`
and `http://cs.potsdam.edu/faculty/laddbc/Teaching/CS1/Week11/PinballGame.java` respectively.
Also in that directory are several `.pnb` files; `.pnb` is an extension I coined to hold pinball description files. It is a text format described further down.
Download the Java and pinball files into your lab directory. Compile and run the pinball game. It should compile but it should not do anything because there is no description of a pinball game.

Phase 2. Look at the `PinballGame` class. Find all of the stub functions (they are the ones with no code in them). You are going to need to replace the stubs with working code.

Phase 3. Modify `getFileName` so that it displays the prompt that is passed into it followed by a single space and then reads a string from the user, returning that value. The returned value should be permitted to include spaces. Modify the `connectScannerToFile` method so it prints out the name of the file it was passed (this is still stub code).
Checkpoint 1:: Compile and run your program and show one of the instructors your program running (prompt, read, echo name of file).

Phase 4. Now you will need to replace the method `ensureExtension`.
`ensureExtension` is passed a file name and returns that file name with a given extension. That is, if it were passed `something` and `txt`, it would return the string `something.txt`. The method is called *ensure*Extension because it should only add the extension if the file name does not already end with it. Thus calling the method with `something.txt` and `txt` should also return `something.txt`.
Your method will take two `String` parameters. You will want to look at the documentation for the `String` class (the `substring`, `equals`, `endsWith` methods in particular). The first parameter is the file name, the second the extension. If the file name does not end with a dot and the given extension, it needs fixing; otherwise return it unchanged. To fix a file name, if it ends with a dot, just return file name plus extension; otherwise return file name plus dot plus extension.
After you get the file name (from `getFileName` above) but before passing it to `connectScannerToFile`, you should ensure that it has the extension for a pinball description file; something like this:

```
fname = ensureExtension(fname, ”pnb”);
```

Phase 5. Now, read the whole file. You will write a loop that reads the next token from the file (a token is, in this case, a word). Then, if the word is "Ball", "Oval", or "Rectangle" you will call a routine that reads the rest of the information for that type of sprite and adds one to the scene.
That is, in `readBufferFile` you will use the provided code to prompt the user and get a `Scanner` hooked up to a file. Then, while there is another word to read, read it. If the word is "Ball", call the `handlePinball` method (and so on for the other two types). Lines are of the form:

```
Ball <color> <x> <y> <w> <h> <dX> <dY>
Oval <color> <x> <y> <w> <h>
Rectangle <color> <x> <y> <w> <h>
```

Where `<color>` is a word that can be passed to `getColor` to return a color and all the rest are double values. x and y are the location of the bumper (or ball); w and h are width and height. The dX and dY are the velocity of the ball. You should use the `Scanner` to do most of the work of reading all the fields of a given bumper.
Checkpoint 2:: Show your code reading files to one of the instructors.

Phase 6. Write a pinball file that describes a pinball level with two balls (starting near the left and right edges of the screen heading toward the center of the screen) and two oval sprite eyes and one rectangle sprite "mouth" to make a "smiley face".

Phase 7. Upload your Java and text files.
Make sure both authors' names are in all of the header comments!
Go to Moodle; you will find an assignment in week 11 labeled **Lab 9**. Go there and upload all of the Java files you created/modified in lab today. Note that there will be other files in your `Lab8` directory. You want to make sure you upload just the `.java` file.
Also make sure that both partners have copies of the program. This program has some bearing on the current assignment so you probably want to have this code around. Comments will also be very helpful in this program for exactly that reason.