CS 1331 Homework 8

Due Thursday, March 14th , 2013 8:00pm

Introduction

This homework is a lot different from anything that we have given you so far. By the end of this assignment, you will have a fully functioning game of Pacman! (Checkout this flash version of Pacman if you have never played it before – ours will be slightly different, but the concept is the same http://www.thepcmanwebsite.com/media/pacman_flash/)

Now, don't panic. That sounds like a lot of work, but we have done a bunch of it for you already. We wanted you to be able to have a homework that does something other than write things to the command line, so we decided to get to some GUI stuff early. After spring break we will actually start teaching GUI things, so you can basically ignore all the GUI components we gave you, but they are there for you to look at if you want to!

We are also giving you all the skeleton code for the project (basically we give you all the class files and method headers with javadocs and comments that tell you what everything is supposed to do). Your job will be to go through and add in your own code to all the places labeled:

```
/* Your Code Here */
```

In addition to code that you have to write, we are giving you two .class files and the documentation of the code that goes along with those classes. The documentation is in the form of javadocs (they really are useful!). Later in this description we will explain how to use the .class files and documentation, and why we are doing things this way.

8.1 Understanding the Code

The best way to approach a project like this is to first figure out how all the code and objects relate to one another. Drawing out a picture of the hierarchy is a great way to do this, so we suggest that you start there.

Draw the main hierarchy (with GameObject on the top), and show how the interfaces are implemented. We see a lot of people who seem to have an aversion to writing things on paper, but I promise you it really does help (for every single project I do, I follow this process on a whiteboard or on paper – it works!).

Also, something that you'll want to pay attention to is the fact that Location is an object – if you pass location as a parameter to something, and then make changes inside that method, you will end up changing both the parameter that you have inside your method and the Location that was actually

passed into that parameter (they are aliases of each other). To work around that, Location has a method called clone() that returns a Location that is an exact replica of the Location, but is an entirely new Location object.

8.2 Using the .class files and documentation

To use the .class files, you just need to make sure that they are located in the same directory as the rest of your code. When you compile and run, java will see them and everything should work out just fine.

When you run the javadoc command on a file, lots of html files are generated (try this on your own code if you would like to see for yourself!) Within the Pacman.zip folder, there is a folder called documentation. This is where you will find the html files for the Level and Location classes.

This might seem like a weird way of doing things – why not just give you the code like we have with other files? Well, as you get further into software development you begin to work more and more with code that comes from other people. Often times, that code is contained within its own library, and all you get to work with are the .class files (or a .jar file of the library), and some documentation. It is important that you learn early on how to work with documentation like this. Additionally, we want you to see that documentation actually is useful :]

8.3 Things that make our Pacman Different

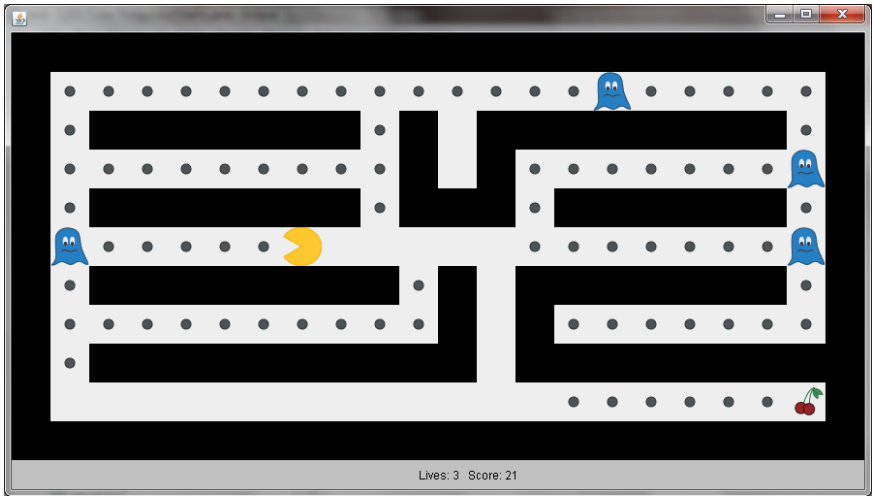
There are 3 major differences between our pacman and the real pacman.

1. Images – we are using some images that I created
2. Ghosts – there is only one color of ghost, and the ghosts do not follow the same movement patterns that traditional pacman does. If you would like to play around with how the ghosts move and use different movement strategies feel free to, but at the very least they need to move the way that we describe.
3. Pacman's movement – to control our pacman, you have to press a key to move him in a direction. If you are not pressing a key, he stops moving.

8.4 Final Product

When you are finished, you will have a working game of pacman! :D

It will look like this:



This homework will be graded almost exclusively on functionality, so make sure that things work!

A good strategy to go about doing this is to first fix all the compiler errors in the code (just make the methods return some kind of default value), and then work through making all of the things that are supposed to happen actually work.

Some important things to check are:

- move pacman
- pick up items
- move ghosts
    - first just get them to move at all
    - make sure they move correctly
- become invincible
- handle collisions with pacman and the ghosts

That is not a full list of everything that you need to do, but it is a good place to start. And as always, come ask us if you have questions!

Turn-in Procedure

Turn in the following files to T-Square. When you are ready, make sure that you have actually **submitted** your files, and not just saved them as a draft.

- GameObject.java
- GameEntity.java
- GameItem.java

- Pacman.java
- Ghost.java
- Pellet.java
- Cherry.java
- PacmanGame.java

Note** Always submit .java files - never submit your .class files. Once you have submitted and received the email from T-Square, you should download your files into a fresh folder. Compile, run, and test those exact files that you turned in. It is pretty much foolproof if you use this technique. Anything less than this is a gamble. See "safe submission" info below. File issues, non-compiling files, non-source code files, etc are all problems and will cause a 0 for the HW. Also, make sure that your files are in on time; the real deadline is 8 pm. While you have until 2 am to get it turned in, we will not accept homework past 2 am for any reason. Don't wait until the last minute!

Verify the Success of Your HW Turn-in

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. Read that email. Look at those filenames. We do not grade .class files. We require source code .java files.
3. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files and the fact that you submitted.
4. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
5. Recompile and test those exact files.
6. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files. (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Reading the email and looking at the files you download helps prevent the turn-in of bytecode (.class) file for which you will receive no credit.
    d. Helps find last minute causes of files not compiling and/or running.