Com S 227
Fall 2014
Miniassignment 1
30 points

Due Date: Friday, February 6, 11:59 pm (midnight)

"Late" deadline (25% penalty): Monday, February 9, 11:59 pm

General information

This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus, http://www.cs.iastate.edu/~cs227/syllabus.html **, for details.**

You will not be able to submit your work unless you have completed the *Academic Dishonesty policy acknowledgement* on the Homework page on Blackboard. Please do this right away.

If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Note: This is a miniassignment and the grading is automated. If you do not submit it correctly, you will receive at most half credit. See the section "What to turn in" at the end of this document

Please start the assignment as soon as possible and get your questions answered right away.

Overview

For this miniassignment you will implement one class, called `Taxi`, that is a simple model of taxi. It is slightly more complex than the `Atom` or `Basketball` classes that you saw in Lab 2, so be sure you have done and understood Lab 2.

Specification

The specification for this asssignment includes

- this pdf,
- any "official" clarifications posted on Piazza, and
- the online javadoc:

http://www.cs.iastate.edu/~cs227/homework/mini1/doc/mini1/Taxi.html

The javadoc contains the exact descriptions of the required methods and their behavior. Read the paragraphs at the top of the javadoc for a general description the class and some usage examples.

Where's the main() method??

There isn't one. This isn't a complete program and you can't "run" it by itself. It's just a single class, that is, the definition for a type of object that might be part of a larger system. To try out your class, you can write a test class with a main method, analogous to **`AtomTest`** in Lab 2. For example:

```
import mini1.Taxi;

public class MyTaxiTest
{
  public static void main(String[] args)
  {
    // new taxi with a base fare of $2.00 and a per-mile rate of $3.00
    Taxi t = new Taxi(2, 3);

    t.drive(10);    // drive 10 miles
    t.drive(5);     // drive 5 miles

    // total miles driven should now be 15
    System.out.println(t.getTotalMiles());
    System.out.println("Expected 15.0");
  }
}
```

There is also a specchecker (see below) that will perform a lot of functional tests, but when you are developing and debugging your code at first you'll want to have some simple test cases of your own as in the **`main`** method above. See the javadoc for more examples of test code (you can copy and past that code into Eclipse).

Getting started

*Smart developers don't try to write all the code and then try to find dozens of errors all at once; they work **incrementally** and test every new feature as it's written. Here is a rough guide for some incremental steps you could take in writing this class.*

1. Create a new, empty project and add a package called `mini1`.

2. Create the `Taxi` class in the `mini1` package and put in stubs for all the required methods and the constructor, as shown in the javadoc. Remember that everything listed in the javadoc is declared `public`. For methods that are required to return a value, just put in a "dummy" return statement that returns zero.

3. Download the specchecker, import it into your project as you did in labs 1 and 2, and run it. There will be lots of error messages appearing in the console output, since you haven't actually implemented the methods yet. *Always start reading from the top.* All you really want to check at this point is whether you have a missing or extra public method, or if something is really wrong like the class having the incorrect name or package. Any such errors will appear first.

4. Now start working incrementally on functionality. The simplest operation to start with is `getTotalMiles()`, which just tells you how many total miles have been driven (by calls to the `drive()` method). You'll need to add an instance variable to keep track of this value. It should be initialized to zero in your constructor and updated in `drive()`. Try it out using a test class as shown above in the section, "Where's the main() method?"

5. A couple of other easy ones are `getMeter()` and `getTotalCash()`. You'll need another instance variable to keep track of the amount of money on the meter, and one to keep track of the total amount of money collected in fares. Both should have initial value zero.

6. The two parameters in the constructor were not needed to implement `getTotalMiles()`, but they will be needed for the other methods. In order to be used in other methods, those values will have to be saved into instance variables in your constructor (kind of like the protons, neutrons, and electrons in your `Atom` class from Lab 2).

7. What about `getCurrentRate()` ? This value is initially zero, but after a call to `startRide()` it should return the per-mile rate that was given in the constructor. This suggests that you need another instance variable to keep track of which value to return for the current rate.

8. Next think about **`startRide()`**. A good way to think about this is to try writing some usage examples and test code to be sure you understand what it is actually supposed to do. For example,

```
Taxi t = new Taxi(2.0, 3.0);
System.out.println(t.getCurrentRate());// what value should we get?
t.startRide();
System.out.println(t.getCurrentRate());// what value should we get?
System.out.println(t.getMeter());      // what value should we get?
```

9. Now, when we call **`drive()`** after **`startRide()`**, the value on the meter should be going up. This requires a slight modification to **`drive()`**.

```
Taxi t = new Taxi(2.0, 3.0);
t.startRide();
t.drive(10);
System.out.println(t.getMeter());      // what value should we get?
```

(Note that if you implemented **`getCurrentRate()`** already, this is easy.)

10. Next you might move on to **`endRide()`**. Again, write some test code first, for example,

```
Taxi t = new Taxi(2.0, 3.0);
t.startRide();
t.drive(10);
t.endRide();
System.out.println(t.getCurrentRate());// what value should we get?
System.out.println(t.getMeter());      // what value should we get?
System.out.println(t.getTotalCash());  // what value should we get?
```

Notes

1. You do NOT need conditional statements ("**`if`**" statements) for this assignment. We will start covering conditional statements later next week, and you won't be penalized if you use them, but you would just be making things more complicated.

2. Your code needs to be in a package named **`mini1`**. Do not add any additional Java classes.

The SpecChecker

You can find the SpecChecker online at
http://www.cs.iastate.edu/~cs227/homework/mini1/speccheck_mini1.jar .

Import and run the SpecChecker just as you practiced in Labs 1 and 2. It will run a number of functional tests and then bring up a dialog offering to create a zip file to submit. Remember that error messages will appear in the console output. There are many test cases so there may be an overwhelming number of error messages. *Always start reading the errors at the top and make incremental corrections in the code to fix them.*

When you are happy with your results, click "Yes" at the dialog to create the zip file.

See the document "SpecChecker HOWTO", which can be found in the Piazza pinned messages under "Syllabus, office hours, useful links" if you are not sure what to do.

Documentation and style

Since this is a miniassignment, the grading is automated and in most cases we will not be reading your code. Therefore, *documentation is not required*. However, we recommend that you document each method as you create it. It is usually easier to write a method correctly after you have written down what it is supposed to do!

If you have questions

For questions, please see the Piazza Q & A pages and click on the folder **`miniassignment1`**. If you don't find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag **`miniassignment1`**. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Java examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have Java code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it. Be sure you have stated a specific question; vague requests of the form "read all my code and tell me what's wrong with it" will generally be ignored.

Of course, the instructors and TAs are always available to help you. See the Office Hours section of the syllabus to find a time that is convenient for you. We do our best to answer every

question carefully, short of actually writing your code for you, but it would be unfair for the staff to fully review your assignment in detail before it is turned in.

Any posts from the instructors on Piazza that are labeled “Official Clarification” are considered to be part of the spec, and you may lose points if you ignore them. Such posts will always be placed in the Announcements section of the course page in addition to the Q&A page. (We promise that no official clarifications will be posted within 24 hours of the due date.)

What to turn in

Note: You will need to complete the "Academic Dishonesty policy questionnaire," found on the Homework page on Blackboard, before the submission link will be visible to you.

Please submit, on Blackboard, the zip file that is created by the SpecChecker. The file will be named **`SUBMIT_THIS_mini1.zip`**. and it will be located in the directory you selected when you ran the SpecChecker. It should contain one directory, **`mini1`**, which in turn contains one file, **`Taxi.java`**. Please LOOK at the file you upload and make sure it is the right one!

Submit the zip file to Blackboard using the Miniassignment 1 submission link and verify that your submission was successful by checking your submission history page. If you are not sure how to do this, see the document "Assignment Submission HOWTO" which can be found in the Piazza pinned messages under “Syllabus, office hours, useful links.”

*We strongly recommend that you just submit the zip file created by the specchecker. If you mess something up and we have to run your code manually, you will receive **at most half the points**.*

> We recommend that you submit the zip file as created by the specchecker. If necessary for some reason, you can create a zip file yourself. The zip file must contain the directory **mini1**, which in turn should contain the file **Taxi.java**. You can accomplish this by zipping up the **src** directory of your project. The file must be a zip file, so be sure you are using the Windows or Mac zip utility, and not a third-party installation of WinRAR, 7-zip, or Winzip.