Homework 4

CS 1331

Due Friday October 4th , 2013 11:59 PM

1 Introduction

Hello everybody! Hope you all had fun manipulating pictures last week. This week, we'll be covering inheritance and a bit of polymorphism.

2 Problem Description

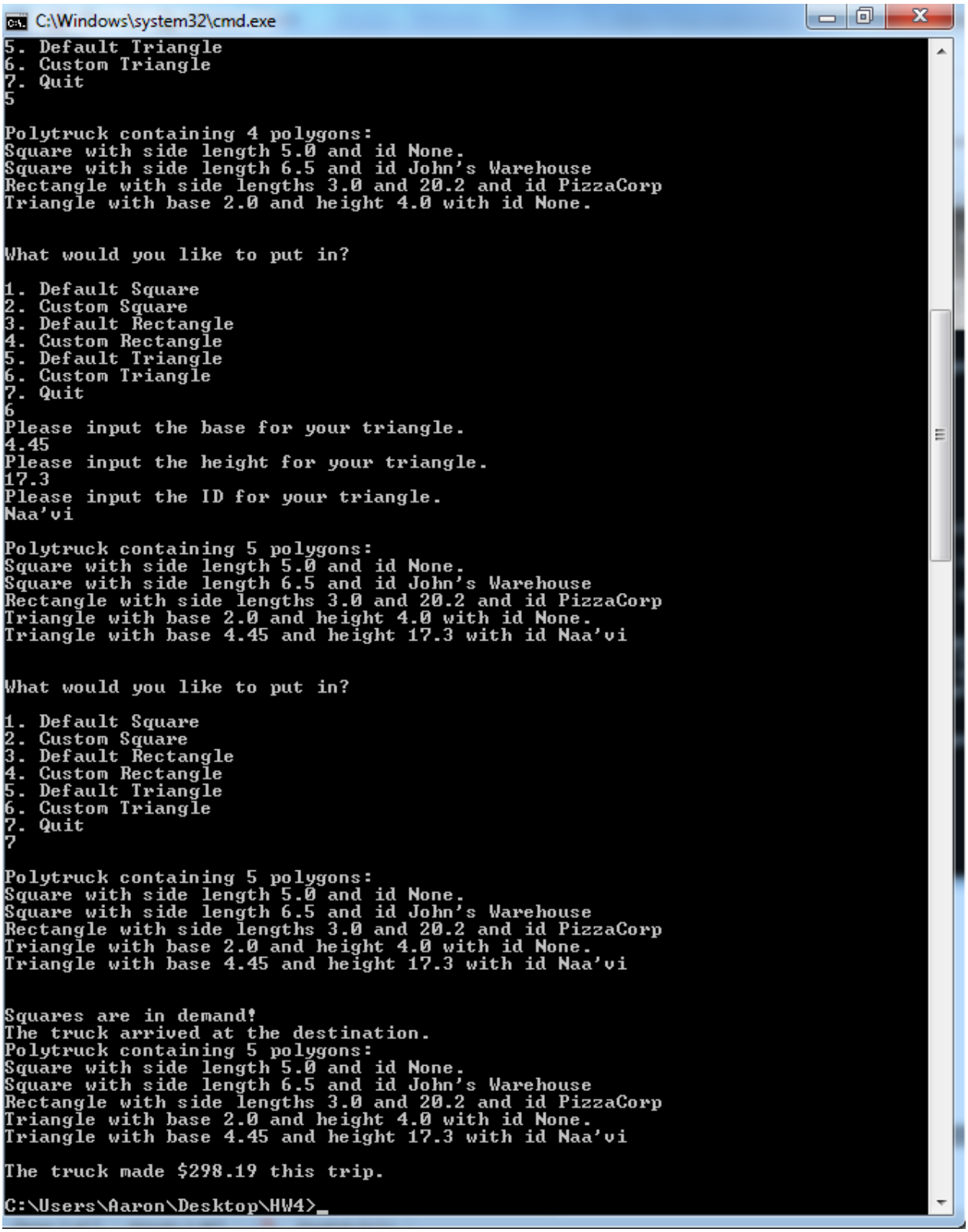
You are an employee of the company called PolyGone, the largest shipper of domestic Polygons in the continental US. Right now, your company specializes in shipping Squares, Rectangles, and Triangles, though soon you expect to branch out into new territory.

Your job is to write the classes for Square, Rectangle, Triangle, and two other polygons of your choosing. These classes must extend the abstract class Polygon that your boss wrote in order to function properly after being loaded into the PolyTruck. The user may either specify the exact dimensions of their Polygon along with an ID to keep track of it, or use the Default version with predetermined dimensions.

Finally, you must write the driver class PolyDriver that asks the user about what types of polygon they would like to load into the truck. After the user is done loading Polygons into the PolyTruck, the truck drives off to deliver the products. Normally a Polygon costs a number of dollars equal to its area – however, at any given time, a single type of Polygon is "in demand", and thus all instances of that type cost twice as much. After randomly determining what Polygon type is in demand, the total value of the truck should be output to the screen.

Here is an example of what your program should output(not including your two custom Polygons you write):


C:\Windows\system32\cmd.exe
Time to load up the truck!
What would you like to put in?
1. Default Square
2. Custom Square
3. Default Rectangle
4. Custom Rectangle
5. Default Triangle
6. Custom Triangle
7. Quit
1
Polytruck containing 1 polygons:
Square with side length 5.0 and id None.
What would you like to put in?
1. Default Square
2. Custom Square
3. Default Rectangle
4. Custom Rectangle
5. Default Triangle
6. Custom Triangle
7. Quit
2
Please input the side length for your square.
6.5
Please input the ID for your square.
John's Warehouse
Polytruck containing 2 polygons:
Square with side length 5.0 and id None.
Square with side length 6.5 and id John's Warehouse
What would you like to put in?
1. Default Square
2. Custom Square
3. Default Rectangle
4. Custom Rectangle
5. Default Triangle
6. Custom Triangle
7. Quit
4
Please input the first side length for your rectangle.
3
Please input the second side length for your rectangle.
20.2
Please input the ID for your rectangle.
PizzaCorp
Polytruck containing 3 polygons:
Square with side length 5.0 and id None.
Square with side length 6.5 and id John's Warehouse
Rectangle with side lengths 3.0 and 20.2 and id PizzaCorp
What would you like to put in?
1. Default Square
2. Custom Square
3. Default Rectangle
4. Custom Rectangle
5. Default Triangle
6. Custom Triangle
7. Quit


C:\Windows\system32\cmd.exe
5. Default Triangle
6. Custom Triangle
7. Quit
5
Polytruck containing 4 polygons:
Square with side length 5.0 and id None.
Square with side length 6.5 and id John's Warehouse
Rectangle with side lengths 3.0 and 20.2 and id PizzaCorp
Triangle with base 2.0 and height 4.0 with id None.
What would you like to put in?
1. Default Square
2. Custom Square
3. Default Rectangle
4. Custom Rectangle
5. Default Triangle
6. Custom Triangle
7. Quit
6
Please input the base for your triangle.
4.45
Please input the height for your triangle.
17.3
Please input the ID for your triangle.
Naa'vi
Polytruck containing 5 polygons:
Square with side length 5.0 and id None.
Square with side length 6.5 and id John's Warehouse
Rectangle with side lengths 3.0 and 20.2 and id PizzaCorp
Triangle with base 2.0 and height 4.0 with id None.
Triangle with base 4.45 and height 17.3 with id Naa'vi
What would you like to put in?
1. Default Square
2. Custom Square
3. Default Rectangle
4. Custom Rectangle
5. Default Triangle
6. Custom Triangle
7. Quit
7
Polytruck containing 5 polygons:
Square with side length 5.0 and id None.
Square with side length 6.5 and id John's Warehouse
Rectangle with side lengths 3.0 and 20.2 and id PizzaCorp
Triangle with base 2.0 and height 4.0 with id None.
Triangle with base 4.45 and height 17.3 with id Naa'vi
Squares are in demand!
The truck arrived at the destination.
Polytruck containing 5 polygons:
Square with side length 5.0 and id None.
Square with side length 6.5 and id John's Warehouse
Rectangle with side lengths 3.0 and 20.2 and id PizzaCorp
Triangle with base 2.0 and height 4.0 with id None.
Triangle with base 4.45 and height 17.3 with id Naa'vi
The truck made $298.19 this trip.
C:\Users\Aaron\Desktop\HW4>

3 Class Outline

All subclasses of Polygon.java:

- Need to fill out getArea(), getTotal(), and override toString()
- Should have two constructors – one which takes in input to make a custom version of that object with specific data, and one which gives "default" values to those variables. **These should use calls to super() and this() as needed! Make sure your constructors do not have duplicate code if at all possible.**

Rectangle.java:

- **Rectangle should extend Polygon.**
- Rectangle should have two double variables which correspond to the two side lengths.
- Rectangle's constructors should take in both side lengths and ID, using super() and this() accordingly.
- The area of a Rectangle is the product of its side lengths.
- Rectangle should have a static boolean which is true if Rectangles are inDemand. If they are, then getTotal() should return double the area.
- Rectangle should have a static method to set inDemand to true.

Square.java:

- **Square should extend Rectangle.**
- **Square's constructors should take in a single side length and pass that up appropriately to Rectangle's constructor.**
- You should not need to override getArea() or getTotal() for Square, as the methods in Rectangle.java should function correctly if you have written the Square constructor well.

Triangle.java:

- Triangle should extend Polygon.
- Triangle should have a base and a height, both as double instance variables.
- Triangle's constructors should take in the base, height, and ID.
- Triangle, like Rectangle, should have an inDemand static variable with a static setter to modify it for all instances of Triangle.
- The area of a Triangle is ½ the base times height.

PolyDriver.java:

- The program should prompt the user similar to how the example prompts (i.e. each number corresponds to a type of Polygon child class instantiation). It should then add the created Polygon object into a PolyTruck object, printing out the truck between each addition.
- The program should prompt the user for Polygons until they quit or until the truck is full.
- The program should randomly select one of the available Polygon classes to be "in demand", set the static variable in that class to true and display a message stating such to the console. Available Polygon classes include Rectangle, Triangle, and either of

your two classes you've written. *(Note that Square is not a valid Polygon to be in demand, because it is using the in-demand status of its parent, Rectangle. If your class does not **directly** extend Polygon and instead extends some other subclass of Polygon, you should use the in-demand status of that parent class appropriately.)*

- After selecting and displaying the type in demand, print out the total cost to the console.

3.1 Useful Tips

- Polygon is an abstract class. You haven't learned about them too much yet, but for now just know that you can't instantiate objects of an abstract class – don't try to say Polygon p = new Polygon(). In addition, any child classes **must** override methods declared as abstract to compile.
- PolyTruck.addPoly() method takes in a Polygon as a parameter, but your classes will all be Squares, Triangles, Rectangles, etc. Think about how to pass these in to the add method.
- Make sure you understand how calls to the super() constructor work. You'll be relying on them to set up the ID in the Polygon class correctly. In addition, avoid duplicating code by having your default constructors consist of a call to your non-default constructors using this().
- You are **not** to modify Polygon.java or PolyTruck.java, so if your program is relying on changes you've made then your grade will suffer.

3.2 Program Requirements

In summary, your program should…

1. Extend Polygon.java with 5 subclasses, Square, Triangle, Rectangle, and two of your own choosing.
2. Write a driver class named PolyDriver.java which allows the user to pick what Polygon is added to the PolyTruck next, until the user quits or the truck is full.
3. Pick a random subclass of Polygon to be in demand, set its static variable, then get the modified total price of the truck with the demand accounted for. Use a Random object and a NumberFormat object to set these, respectively.
4. Javadoc all methods and classes (except main()).

3.3 Turn-in Procedure

Turn in the following files to T-Square. When you're ready, double-check that you have submitted and not just saved a draft.

- Square.java
- Rectangle.java
- Triangle.java
- PolyDriver.java
- Two of your own subclasses for Polygon.java.

Make sure you are submitting .java files and not any .class files.

4 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. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. 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. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight Friday. Do not wait until the last minute!