Computer Science 204

Assignment #2

Shipping Crates of Balls

Due Date : Monday, February 21, 2011, 11:59 p.m.

40 Points

Objective

The purpose of this assignment is to extend our familiarity with the Java language. Specifically, we will become familiar with the numeric data types, the Math class and performing arithmetic operations with them. After completing this assignment you will also have a good grasp of how classes are implemented. This program will make use of many of the statements and methods in Chapters 1 – 4.

Assignment Summary

You work for a company that sells solid balls made of various materials for sports: bowling, bocce, croquet, etc. This company has to ship its balls in custom made boxes that must meet specific size and weight limitations placed on it by the shipping service and airline. For this program you may assume that the balls are packed in configurations (A) and (B) as shown in Figure 1.[1]

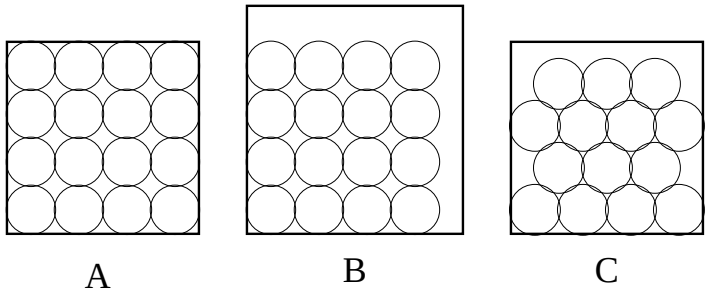

Figure 1: Shipping Crate Configurations

For this program, you will create a class to represent a rectangular solid. It should be named `ShippingCrate` and it should have the following methods:

- A constructor which takes 3 double parameters, representing the crate length, width, and height, in inches, in that order. Your constructor should set the default diameter of the balls being shipped equal to the smallest dimension of your crate. The default cost of the balls should 50¢.

[1]For 20 points of extra credit you may write an additional class specification, called `PackedShippingCrate` that uses the packing model shown in figure (C). The packed shipping crate is hard to visualize in 2D. A 3D model will be placed on the class webpage for.

- A constructor which takes 4 double parameters, representing the crate length, width, height, and ball diameter in inches, and one int parameter which is the cost of a single ball in cents.
- A method named `setMaterialDensity` which takes a single double parameter, representing the weight in **ounces** of **one cubic inch** of ball material. The default material density should be zero.
- A method named `setBallDiameter` that takes a single double parameter and sets the diameter of the balls (in inches) that are to be contained in the crate. It does not change the size of the crate.
- A method named `setBallPrice` that takes a single int parameter and sets the price of each ball in the crate in cents.
- A method called `getBallDiameter` which takes no parameters and returns a double representing diameter of each ball in in the crate in inches.
- A method called `getExteriorArea` which takes no parameters and returns a double representing the number of square feet for the exterior surface area of the crate. You have no information regarding the thickness of the wood, so assume that the inside dimensions and the outside dimensions of the crate are the same.
- A method called `getDiagonal` which takes no parameters and returns a double representing the number inches for the distance from one corner to the opposite corner of the **crate**.
- A method called `getWoodWeight` which takes no parameters, but returns the weight (in pounds) of the wood required to build the crate as a double. This method assumes that the wood is of uniform composition and thickness with a mass of 2.45 pounds per square foot. This mass to area ratio should be a constant in your method only (with appropriate storage classification).
- A method called `getBallWeight` which takes no parameters, and returns the weight of all the balls in the container in pounds as a double
- A method called `getBallWeight` which takes a single double parameter, representing the density in **ounces of a one cubic inch of ball material**, and returns the weight of all the balls in the container in pounds as a double. Note – the parameter passed in should not modify any instance variables!
- A method called `getBallCount` which takes no parameters, but returns an integer value representing the number of balls that can be stored in the crate. This assumes that balls are rigidly packed one on top of another (like in a pack of golf balls).
- A method called `getWeight` which takes no parameters, but returns a number representing the total mass of the crate and its contents as a double. This method must call the `getWoodWeight` and `getBallWeight`[2] methods.
- A method called `getCrateValue` which takes no parameters, but returns a double, rounded to two decimal places, which represents the value of all the balls in the crate. This would be the amount for which you might insure the crate contents.

[2]The one with no parameters.

- A method called `getPackingEfficiency` that takes no parameters and returns the percent of space in the crate filled with balls as a double. Higher numbers mean that the crate is packed more efficiently.
- A `toString` method which will return a String with the dimensions, formatted as:

  Solid (lll inches long, www inches wide, hhh inches high, ddd inches diag, wwwww lbs.)

  where the the final term refers to the mass of the entire crate including the balls. Inches that are returned by the `toString` method must be the minimum inches that will contain the entire crate. For example, if the length, width, and height of the crate were 12.3, 14.5, 22.6, the `toString` method should return 13, 15, 23 and the weight. The weight should be rounded to the nearest whole number and displayed as an integer.
- There is obiously going to be a lot of converting between inches and feet as well as pounds and ounces in this class definition. You need to define conversion constants in your class definition using storage methods that allow these constant to be used by all the methods in your class.
- It is HIGHLY DESIRABLE to use some private methods in your class to handle things like determining how many balls can be in the crate. A big part of this project is deciding what instance variables you need and how to keep them "up to date".

To help you verify that your code is working properly, I will provide my `ShippingCrate.class` file on the website. You can compare your results with mine. Please note – you should not use decision or loop statements in the construction of this class. Follow the coding style described in Appendix A of your text (except please place your instance variables at the top of your class file). Adequately comment your code and use Javadoc to describe the the constructors and methods in your class.

Documentation Requirement

In addition to the normal comments one might place in a program, for `ShippingCrate` class you are also required to write the corresponding *javadoc* code to comment each of your methods.[3] Please follow the examples in your text. I will generate the *javadoc* from your program and it will be counted toward your grade on this assignment.

Hand In

Send me your ShippingCrate.java file (including the javadoc) via e-mail. Be sure to send yourself a copy too. While you will have to write a tester program to verify your class is working properly, I only want the ShippingCrate.java file.

Revision Policy

All assignments handed in on or before the due date that you do not receive full credit for in either implementation or documentation are eligible for revision.

[3] This also applies to the `PackedShippingCrate` class.