Computer Science 204

Assignment #2

Shipping Crates of Balls – REVISED

Due Date : Tuesday, February 24, 2009, 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. 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 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 ball diameter should also be stored in inches.
- A method named `setMaterialWeight` which takes a single double parameter, representing the weight in **ounces** of **one cubic inch** of ball material. The default material weight should be zero.
- A method named `setDiameter` that takes a single double parameter and sets the diameter of the balls (in inches) that are to be contained in the crate.
- A method called `squareFeet` which takes no parameters and returns a double representing the number of square feet of the exterior of the crate. I have given you 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 `diagonal` which takes no parameters and returns a double representing the number inches of the distance from one corner to the opposite corner of the **crate**.
- A method called `woodWeight` 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 `ballWeight` which takes no parameters, and returns the weight of all the balls in the container in pounds as a double
- A method called `ballWeight` which takes a single double parameter, representing the weight 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 `ballCount` 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 `weight` 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 `woodWeight` and `ballWeight`[1] methods.
- A method called `packingEfficiency` 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.
- 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.
- 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. The weight should be rounded to the nearest whole number and displayed as an integer.

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. Adequately comment your code and use Javadoc to describe the the constructors and methods in your class.

Hand In

Send me your ShippingCrate.java file via e-mail. Be sure to send yourself a copy too.

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.

[1]The one with no parameters.