CS 116

Lab Assignment # 5: Inheritance[1]

- **Points: 4**
- **Submission**
  - **Deadline: Wednesday 03/30 11:59 PM**
  - **Submit on Blackboard under assignment "Lab5". Please make sure that you click the "Submit" button and not just "Save".**
- **Late Submission Policy**
  - **You can do a late submission until Friday 04/1 11:59PM with a 5% penalty on the total points for this assignment.**
  - **After that solutions will be posted and no submission will be accepted**
- **Early Submission**
  - **You can also get 5% extra point on your score on the assignment for early submission if you submit by Tuesday 03/29 11:59PM.**
- **Getting help**
  - **From instructor during office hours in SB228 or by email.**
  - **By seeing one of the TAs during the listed TA office hours in room SB108. (check the course website http://cs.iit.edu/~jkorah/cs116/)**
  - **By visiting the ARC (Academic Resource Center).**
- **Academic Dishonesty Policy**
  - **Working with a partner: You can work with a partner as assigned by the instructor. Otherwise this should be considered individual work.**
    - **Even if you are working with a partner, you and your partner are required to make individual submissions.**
  - **Please note: In case two submissions are declared identical (and if you are not supposed to work together) the excuse: we worked together, does not hold and both submission will be treated according to ethics rules.**

PROGRAMMING TASK A: Create a folder called TaskA in your assignment folder to store the files for task A.

Objectives:

[1] Acknowledge materials by Matt Bauer

1. Design, code and test a derived class demonstrating inheritance

Circle and Cylinder (1.5 points) - You have been hired by a popular canned soup company to help optimize production. One of the ways to optimize production is to look closely at cost: how many cans of soup can be produced from a batch of soup and looking at the cost/profit involved. The sizes of cans vary so you decide that it would be helpful to have a program that can calculate the volume of a can given its radius (or diameter) and its length (or height). Rather than coding a class to calculate volume of a cylinder, you realize that you can use a Circle class that calculates the area of a circle first, is a more flexible approach (you may have other circle related volumes other than a cylinder later). You know that the volume of a can would just be the area of the circular top of the can multiplied by the height of the can.

1. Create a Circle class which has one instance variable of radius (default=0) and methods to calculate the circumference and area of a circle in addition to the default/non-default constructors, mutators, accessors, and toString() methods (the toString should return a formatted String with the radius, circumference and area).

2. Create a new class, Cylinder, inheriting from the Circle class but using the additional data of Cylinder height (type=float, default value=0). The Cylinder class should have the following:

- a default constructor
- 2 non-default constructors
    - 1 non-default constructor will have radius and height as parameters
    - 1 non-default constructor will have a Circle object and height as parameters
- Appropriate accessor and mutator methods
- a method to calculate the volume
- a toString() method (which overrides the toString() method in your Circle class) which calls the toString() method from the super class then adds the additional data (height, volume) from the Cylinder class

Create a client file called CylinderClient.java and implement the following instructions.

- Create a Circle object circle1. Print out the object.
- Create a Cylinder object cylinder1. Print out the object.
- Create a Cylinder object cylinder2 from the circle object circle. Print out the object.
- Change the radius of circle1. Print out the object.
- Call Circle methods on Cylinder1 and print out their outputs.

PROGRAMMING TASK B: Create a folder called TaskB in your assignment folder to store the files for task B.

Objectives: Design, code and test a derived class demonstrating inheritance and polymorphism

Cylinder and Sphere (1.5 points) - A local ice cream store is trying to calculate a price to charge for its scoops of ice cream based on the number of scoops it can make from a tub of ice cream. The ice cream tubs are the usual round containers seen in most ice cream stores, essentially large cylinders. The store is assuming the scoops are spheres. We will calculate the volume of a scoop and the volume of a tub of ice cream and see how many scoops we can get from a tub. The formula for the volume of a sphere is (4/3)*PI*radius^3

Question: How do you decide if the sphere should inherit from cylinder OR inherit from circle? Hint: Think of OTHER applications where the formulas for a sphere might be used…would they always need to have a cylinder object? Would all sphere applications probably have to have a circle object? The answer to these questions should lead you to conclude it would be best to have the Sphere inherit from Circle.

Start with your Circle class and inherited Cylinder class from the previous lab. Create a Sphere class, also inheriting from the Circle class. The Sphere class should have the following:

- a default constructor
- 2 non-default constructors
    - 1 non-default constructor will have radius as parameter
    - 1 non-default constructor will have a Circle object as parameter
- a method to calculate the volume
- a toString() method (which overrides the toString() method in your Circle class) which calls the toString() method from the super class then adds the additional data (volume) from the Sphere class

Create a CylinderSphereClient.java driver program and implement the following instructions.

- Create and output a Cylinder radius 28 and height 35
- Create and output a Sphere radius 6
- Calculate and output the number of Spheres in the Cylinder
- Create and output a Circle (different radius from above)
- Assign subclass object reference (Cylinder) to a superclass object reference (Circle), then output the superclass object reference
- Assign subclass object reference (Sphere) to a superclass object reference (Circle), then output the superclass object reference
- Create an array of 3 Circle objects, assign a new Circle to one position, a new Cylinder to the second position, and a new Sphere to the third position, then use a loop to output the objects in the array

PROGRAMMING TASK C: Create a folder called TaskC in your assignment folder to store the files for task C.

Objectives: Design, code and test an abstract class and derived classes demonstrating inheritance and polymorphism.

Circle Volume (1 point) - In the previous lab we explored inheritance and polymorphism using Sphere and Cylinder classes inherting from a Circle class. We saw that this is not a perfect "is a" relationship required for inheritance. A Sphere/Cylinder "has a" circle for a horizontal cross section. However, a Sphere or Cylinder are both CircleVolume objects. A Sphere "is a" CircleVolume object and a Cylinder "is a" CircleVolume object. But such a thing as a CircleVolume object does not exist, it is an abstraction we come up with to group 3-D objects. We saw in lecture that Object-Oriented programming supports abstract objects and inheritance. Abstract objects cannot be instantiated (never can be "new"d).

Change your Circle class to an abstract class CircleVolume with the following:

- a default constructor
- 2 non-default constructors
    - a non-default constructor with a radius as parameter
    - a non-default constructor with a CircleVolume object as parameter
- setRadius/getRadius/getCircumference/getArea/toString as before
- an abstract method getVolume

Update your Sphere and Cylinder classes to inherit from the CircleVolume class.

Create a CylinderSphereClient.java driver program with the instructions below.

- Create objects of both classes and output.
- Assign each object to a reference of the abstract class and test both abstract and non-abstract methods.
- Create an array of the abstract class of size 4, and insert 2 objects of Sphere and Cylinder class each.
- Demonstrate polymorphism by printing out the objects in the array.

Submission instructions

- In your submission you must include

    a. The source code files and the compiled files for the program.
- Zip all files and name the zip file using your last name followed by your first name followed by the name of the assignment

    i.e. Doe_Jane_Lab5.zip
- Upload the file on assignment folder: Lab5 on Blackboard.