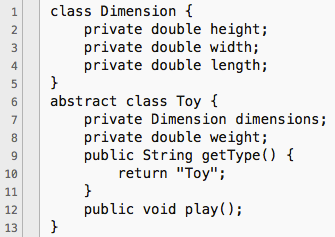
Name \_\_\_\_**Michael Pedzimaz**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Score \_\_\_\_ / 16

**CPSC 245: Object-Oriented Programming**

**Homework #2, Part 2**

Consider the following code.



1. There is something wrong with line 12. Write a corrected version below, assuming it is meant to be a function without a body.

**public abstract void play();**

1. The relationship between Toy and Dimension can be described as
   1. inheritance
   2. **ownership**
   3. association
   4. encapsulation
2. In Dimension, write a set function for the height that will ensure that it is not set negative.

**public double setHeight(int h){**

**if (h < 0){**

**height = 0;**

**}else {**

**height = h;**

**}**

**return height;**

**}**

1. What is not true about constructors?
   1. They do not have return types
   2. They have the same name as the name of the class
   3. **There can only be one per class.**
   4. They are usually used to initialize the data members and memory of a new object you are creating.
2. Write a default constructor for Dimension that will set the width, height, and length equal to 0.

**public Dimension(){**

**width = 0;**

**height = 0;**

**length = 0;**

**}**

1. Assume you have written set functions for all the private data members of Dimension that ensure that they are not set to unrealistic values. Write a non-default constructor for Dimension that uses them to set the values of Dimension’s private data members.

**public Dimension(int w, int h, int l){**

**setWidth(w);**

**setHeight(h);**

**setLength(l);**

**}**

1. Write a toString function for Dimension that will return a String containing the height, width, and length of a Dimension object, each with three digits after the decimal place.

**public String toString(){**

**return String.format(“height=%.3f width =%.3f length=%.3f”, height, width, length);**

**}**

1. To create an object, which keyword do you have to use?
   1. **new**
   2. super
   3. this
   4. void
2. Write a default constructor for Toy that will set its weight to 0 and its dimensions to a default Dimension object. In other words, for that second part, you will have to use the default constructor for Dimension to create it).

**public Toy(){**

**weight = 0;**

**dimensions = new Dimension();**

**}**

1. Write a non-default constructor for Toy that will set its width, height, length, and weight to values passed in to the constructor. Notice that setting the width, height, and length will involve creating a Dimension object. Assume you have written a setWeight function for Toy already that ensures that the weight is not set to a negative value.

**public Toy(double h, double w, double len, double wt){**

**new Dimension(h, w, len);**

**setWeight(wt);**

**}**

1. Write a toString function for Toy that will return a String containing its type (using the getType function) followed by its dimensions and weight with 3 digits after the decimal point. Use Dimension’s toString function that you wrote in #7 for this.

**public String toString(){**

**return String.format(“%s %s %.3f”, getType(), dimensions.toString(), weight)**

**}**

Suppose now I’ve created a class called ActionFigure, like so:

class ActionFigure extends Toy {

private String gender;

public String getType() {

return "ActionFigure";

}

}

1. Which of the following is not true?
   1. It has data members called *dimensions* and *weight*.
   2. **I can use the variable names *dimensions* and *weight* in ActionFigure’s code to set them.**
   3. I will absolutely have to declare a play() function for it.
   4. I can pass an ActionFigure object to System.out.println, and its dimensions and type will be printed to the screen.
2. Write a default constructor for ActionFigure that set the gender to “F” and all of its other properties to default values. Reference the appropriate super class constructor to do this.

**public ActionFigure(){**

**super();**

**gender = “F”; }**

1. Write the toString function for ActionFigure. Return the same things as Toy’s toString function, but have the gender value appear at the end. Make sure the correct toy type (“ActionFigure”) is included. Don’t write any more code than is absolutely necessary, or you will not receive credit for this. (Hint: recognize that the correct toy type will print automatically thanks to polymorphism.)

**public String toString(){**

**return String.format(“%s %s”, super.toString(), gender);**

**}**

1. Declare and create an ArrayList of Toy objects called toys.

**ArrayList<Toy> toys = new ArrayList<Toy>();**

1. Add a new default ActionFigure object to the toys list you created in #15.  
   **toys.add(new ActionFigure(args));**