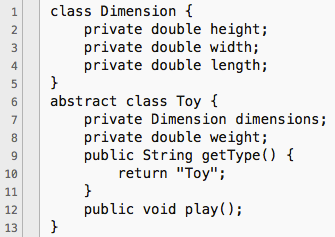
Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Score \_\_\_\_ / 38

**CPSC 24500: Object-Oriented Programming**

**Homework #9 – Object-Oriented Concepts  
Due Wednesday, November 29 at 6pm**

Consider the following code.



1. There is something wrong with line 12. Rewrite line 12 below so that it is correct.

**Answer:**

public abstract void play();

1. The relationship between Toy and Dimension can be described as
   1. inheritance
   2. ownership
   3. association
   4. encapsulation

**Answer:** (C)

1. In Dimension, write a set function for the height that will ensure that it is not set negative.

**Answer:**

public void setHeight(double height) {

if (height >= 0) {

this.height = height;

} else {

System.out.println("Height cannot be negative");

}

}

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.

**Answer:(**C)

1. Write a default constructor for Dimension that will set the width, height, and length equal to 0.

**Answer:**

public Dimension() {

this.height = 0;

this.width = 0;

this.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.

**Answer:**

public Dimension(double width, double height, double length) {

setWidth(width);

setHeight(height);

setLength(length);

}

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.

**Answer:**

@Override

public String toString() {

return String.format("Dimension [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

**Answer:** (A)

1. Write a default constructor for Toy that will set its weight to 0 and its dimensions to a default Dimension object.

**Answer:**

public Toy() {

this.weight = 0;

this.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. Assume you have written a setWeight function for Toy already that ensures that the weight is not set to a negative value.

**Answer:**

public Toy(double width, double height, double length, double weight) {

this.dimensions = new Dimension(width, height, length); // Assumes Dimension's non-default constructor is defined.

setWeight(weight);

}

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. Leverage Dimension’s toString function that you wrote in #7 for part of this. (2 points)

**Answer:**

@Override

public String toString() {

return getType() + " " + dimensions.toString() + String.format(" Weight=%.3f", 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 directly 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.

**Answer:(**C)

1. 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.

**Answer:**

public ActionFigure() {

super();

this.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 necessary. (2 points)

**Answer:**

@Override

public String toString() {

return super.toString() + " Gender=" + gender;

}

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

**Answer:**

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

1. Add a new default ActionFigure object to the toys list you created in #15.

**Answer:**

toys.add(new ActionFigure());

1. Draw a UML diagram that shows the Toy, Dimension, and ActionFigure classes. Make sure you show the data and functions for each class with their proper visibility. Also make sure you show their relationships correctly. (4 points)

A black and white cube with text

Description automatically generated

**Multiple choice questions**

1. The relationship between Laptop and Screen can best be categorized as
   1. Inheritance
   2. Association
   3. Ownership
   4. Private

**Answer:(**B)

1. Which one of these is not a purpose of information hiding in object-oriented programming?
   1. To protect the values of variables from being changed inappropriately
   2. To free other programmers from having to know too many details of the inner workings of the class.
   3. To protect people's personal information from hackers.
   4. To provide an opportunity to format or convert data before returning it.

**Answer:(**C)

1. The relationship between Bayonet and Weapon can best be described as
   1. Inheritance
   2. Polymorphism
   3. Ownership
   4. Private

**Answer:(**A)

1. Which of the following is not true of the relationship *composition*?
   1. The owner is responsible for creating and destroying the object it owns
   2. Other objects of the owner type can own the very same object.
   3. It involves exclusive ownership, meaning that no other object could own the owned object.
   4. It, like aggregation, is a form of ownership.

**Answer:(**B)

1. In the relationship between Hammer and Tool, which of the following is correct?
   1. Hammer is the superclass, and Tool is the subclass.
   2. Hammer is the subclass, and Tool is the superclass.
   3. Hammer is the owner, and Tool is the owned.
   4. Hammer is the owned, and Tool is the owner.

**Answer:(**B)

1. Which of the following is a false statement?
   1. A class is like a blueprint, and an object is something built according to that blueprint.
   2. A class can contain data members and methods.
   3. A class can contain objects as data members within it.
   4. Usually, only one object can be built for any given class.

**Answer:(**D)

1. Which of the following is false about an abstract class?
   1. It contains one or more abstract functions.
   2. You cannot create an object of an abstract class.
   3. It cannot have any non-abstract functions.
   4. It is the most generic version of a family of related types of things.

**Answer:(**C)

1. Why is polymorphism powerful?
   1. It enables us to refer to related objects generally but still access specific functionality for each object.
   2. It enables us to build more complicated objects starting from simpler ones.
   3. It helps us protect data from being inadvertently changed.
   4. It helps us build objects that contain other objects.

**Answer:(**B)

1. Why do we write public get and set functions?
   1. Object-oriented languages require us to do so.
   2. If we didn't, there would be no way to read or write the values of the data members of a class.
   3. In combination with making data private, these public functions give us a way to read and write the values of these variables in a controlled way.
   4. They make a program run more efficiently.

**Answer:(**C)

1. Which pattern tries to make it easier to create a variety of related objects?
   1. Model-View-Controller
   2. Factory
   3. Delegation
   4. Singleton

**Answer:(**B)

1. Which pattern tries to avoid mixing the storage of data and how it is displayed?
   1. Model-View-Controller
   2. Factory
   3. Delegation
   4. Singleton

**Answer:(**A)

1. Which pattern often uses aggregation or composition to enable an owner to perform a task by asking for help from the things it owns?
   1. Model-View-Controller
   2. Factory
   3. Delegation
   4. Singleton

**Answer:(**C)

1. Which of the following is a false statement?
   1. An interface is a data type that consists entirely of abstract functions.
   2. A class can implement multiple interfaces but extend only one class.
   3. "extends" (i.e. inheritance) means "is a" whereas "implementing an interface" means "acts like"
   4. Using interfaces helps a program run more efficiently.

**Answer:(**D)

1. Which software principle does this example violate? Explain how you’d improve it. (3 points)  
   public class City {  
    private ArrayList<Apartment> apartments;

private ArrayList<Home> homes;  
 private ArrayList<Office> officeBuilding;  
}

**Answer:**

public class City {

private ApartmentManager apartmentManager;

private HomeManager homeManager;

private OfficeBuildingManager officeBuildingManager;

}