**CS 245: Object-Oriented Programming**

**Homework 1**

**Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Score \_\_\_\_\_ / 22**

1. The relationship between Laptop and Screen can best be categorized as
   1. Inheritance
   2. Association
   3. Ownership
   4. Private
2. What is the 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. All of the above
   5. Just choices a and b
3. The relationship between Bayonet and Weapon could best be described as
   1. Inheritance
   2. Association
   3. Ownership
   4. Private
4. 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.
5. In the relationship between Hammer and Tool, which is the superclass and which is the subclass?
6. 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.
7. 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.
8. 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.
9. 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.
10. One problem associated with software development is rigidity, which means
    1. It is difficult to move code from one project to another.
    2. It is difficult to change the code to meet changing or additional needs.
    3. Changing the code tends to break it.
    4. It is difficult to use.
11. Which pattern tries to make it easier to create a variety of related objects?
12. Model-View-Controller
13. Singleton
14. Factory
15. Delegation
16. Which pattern tries to avoid mixing the storage of data and how it is displayed?
17. Model-View-Controller
18. Singleton
19. Factory
20. Delegation
21. Which pattern often uses a private constructor and static functions as part of its design?
22. Model-View-Controller
23. Singleton
24. Factory
25. Delegation
26. Which pattern often uses aggregation or composition to enable an owner to perform a task by asking from help from the things it owns?
27. Model-View-Controller
28. Singleton
29. Factory
30. Delegation
31. Which pattern is used particularly to ensure consistency across multiple threads?
32. Model-View-Controller
33. Singleton
34. Factory
35. Delegation
36. Which concepts enable the Factory pattern to be particularly useful?
37. Aggregation and Composition
38. Information Hiding and Encapsulation
39. Inheritance and Polymorphism
40. Abstraction and Ownership
41. Which of the following is a false statement?
42. An interface is a data type that consists entirely of abstract functions.
43. A class can implement multiple interfaces but extend only one class.
44. “extends” (i.e. inheritance) means “is a” whereas “implementing an interface” means “acts like”
45. Using interfaces helps avoid violating the Liskov Substitution Principle.
46. Which design principle does the following code violate?  
    class SoundPlayer {  
     public void playSound(String type, Object obj) {  
     if (type.equals(“horn”)) {  
     (Horn)obj.play();  
     } else if (type.equals(“whistle”)) {  
     (Whistle)obj.play();  
     }  
     }
47. Interface Segregation Principle
48. Dependency Inversion Principle
49. Open-Close Principle
50. Single-Responsibility Principle
51. Which design principle does the following code violate?  
    interface ISoundPlayer {  
     public abstract setEncoding(Sting encoding);  
     public abstract setDestination(String destination);  
     }
52. Interface Segregation Principle
53. Dependency Inversion Principle
54. Open-Close Principle
55. Single-Responsibility Principle
56. Draw a UML diagram for the following system: A cell phone is a type of computing device. It has a screen, usb jack, power button, and wake button. Include data members and methods for the classes you define, and show the appropriate relationships (3 points)