Name _____

The following might be names of patterns: Abstract Class, Abstract Factory, Adapter, Bridge, Builder, Chain of Responsibility, Client, Collaborator, Command, Command Processor, Composite, Context, Decorator, Façade, Factory Method, Flyweight, Interpreter, Mediator, Memento, Observer, Prototype, Proxy, Singleton, Specification, State, Strategy, Template Method, Visitor.

The purpose of many of the design patterns is to make it easy to change some property of the system. In the first 7 problems give one pattern even if more than one pattern may apply.

1. (3 points) What design pattern would you use to make it easy to change the way a set of objects interact with each other.

2. (3 points) What design pattern would you use to make it easy to change the implementation of an abstraction.

3. (3 points) What design pattern would you use when you change the internal state of an object, but you want to be able to remember this state and then later tell the object to go back to that state.

4. (3 points) What design pattern would you use when you need to evaluate an expression in a small language.

5. (3 points) What design pattern would you use when you want to reduce the coupling between classes.

6. (3 points) What design pattern should you think of when you want to hide how you construct a complex object?.

Suppose you were designing the file system of an operating system.

7. (3 points) Your file system has many classes in it, but you want to hide them from the application programmers and present a simpler interface to them. What design pattern lets you hide the complexity of a subsystem like a file system?

8. (10 points) Proxy and Strategy are similar because they both contain multiple nested classes. In other respects, they are quite different. Explain the differences between Proxy and Strategy.

9. (10 points) The Builder pattern would be simpler to implement if the director and builder were combined into one class. What advantages do we gain by keeping them separate?

10. (10 points) Decorator and Chain of Responsibility are similar because they both contain multiple nested classes. In other respects, they are quite different. Explain the differences between Decorator and Chain of Responsibility.

11. (10 points) What is the difference between intrinsic and extrinsic state?

12. (10 points) The memento pattern states that it does not violate encapsulation but yet accesses an object's internal state. Explain how the memento does not violate encapsulation and discuss your language's support for this feature of the memento.

13. (10 points) In object-oriented programming one is advised to avoid case (and if) statements. Select one design pattern that helps avoid case statements and explain how it helps.

14. Each part is 5 points.

    a. Explain how the Abstract Factory promotes consistency among products that it can create.

    b. Give the most common two ways in which an Abstract factory actually creates objects.