Exercises on Design Patterns

1, Write down three differences between abstract classes and interfaces in Java 8.

- interface can inherit from multiple sources
- abstract class cannot inherit from multiple sources
- abstract class can have constructors
- interface inherits from nowhere while abstract class inherits from object
- differences when using reflection
- An abstract class can have non-abstract methods, while all the methods of an interface are abstract
- An abstract class can have protected, private methods, but in interface everything must be public

2, Are the following true or false?

a, Every interface must have at least one method. – FALSE (interface with no methods is a marker interface)

b, An interface can declare instance fields that an implementing class must also declare. – FALSE

c, Although you can't instantiate an interface, an interface definition can declare constructor methods that require an implementing class to provide constructors with given signatures. – FALSE

3, Provide an example of an interface with methods that do not imply responsibility on the part of the implementing class to take action on behalf of the caller or to return a value.

- observer – observables
- public void mouseDragged(MouseEvent e)
- public void mouseMoved(MouseEvent e)
- implementing class may need to take an action in response to a method call

4, What is the value of a stub class like WindowAdapter which is composed of methods that do nothing?

This is abstract adapter class for receiving window events. The methods in this class are empty. This class exists as convenience for creating listener objects. WindowListener must implement by default implement all the methods the interface declares, even if the class will ignore these methods when called. The WindowAdapter class ignores all the methods in WindowListener. This lets a subclass implement only the methods that it needs to/want to!

5, How can you prevent other developers from constructing new instances of your class? Provide appropriate examples to illustrate your answer. singleton?

To prevent other developers from instantiating your class, create a single constructor with private visibility. Singleton?

6, Why might you decide to lazy-initialise a singleton instance rather than initialise it in its field declaration? Provide examples of both approaches to illustrate your answer.

We might not have all the informations/data required during the first initialisation time during field declaration.

7, Using the java.util.Observable and java.util.Observer classes/interfaces show how one object can be informed of updates to another object.

see ObservableDemo.java for the example!

8, "The Observer pattern supports the MVC pattern". State if this statement is true or false and support your answer by use of an appropriate example.

True, model is the observable.

9, Provide examples of two commonly used Java methods that return a new object.

clone() method
some methods in swing
new Date object returned as string using toString() method

10, What are the signs that a Factory Method is at work?

- usually more classes used
- one class variable contains other class

```
MessageRenderer mr = new StandardOutMessageRenderer();
MessageProvider mp = new HelloWorldMessageProvider();
mr.setMessageProvider(mp);
```

11, If you want to direct output to System.out instead of to a file, you can create a Writer object that directs its output to System.out: