Week 3 Exercises
1

Classes can implement multiple interfaces but only one abstract class

Abstract classes can declare instance variables

Abstract classes can define constructors

2

a False

eg Marker Interface

public interface Marker

b False

Any instance fields are assumed to be static and final see example

c False

Interfaces can’t declare constructor methods

3

For an class to implement an interface it has to implement it’s methods these methods do not have to take any action when called. Examples include the MouseMotionListener which when implemented has to include mouseDragged and MouseMoved. One of these methods is often ignored

4

Having an adapter class between your interface and functional classes mean that there are dummy implementations present that do not have to be redefined in subclasses

5

Make your constructor private.

6

See code

7

see code

8

see code

9

toString creates a new object of type string when called in many classes.
clone() creates a new object with copies of all the elements of the instance that it is called for.

10. What are the signs that a Factory Method is at work?
Small children working up chimneys
Rather than using a constructor a method such as getinstance() is called.
A new object is created
Returns a type defined by an abstract class of interface

11.

```
WrapFilter out =
   new WrapFilter(
      new BufferedWriter(
         new RandomCaseFilter(
            new PrintWriter(System.out))),15);
out.setCenter(true);
```

L1

L2.
a - synchronise the getInstance Method
b - it will have to obtain the key before anything is run
c -

```
package singletonpattern;
public class SingletonLazyDoubleCheck {
   private volatile static SingletonLazyDoubleCheck sc = null;
   private SingletonLazyDoubleCheck() {
   }
   public static SingletonLazyDoubleCheck getInstance() {
      if (sc == null) {
         synchronized (SingletonLazyDoubleCheck.class) {
            if (sc == null) {
               sc = new SingletonLazyDoubleCheck();
            }
} }
return sc; }
}
```