Skiddoo Coding Test

This test includes 3 questions. The first question is a simple Java code review, the second is a more indepth Java coding question which requires updating and improving an existing set of Java files and the third is a general question on the future of web development.

Question 1. Java code review

There's no need to write any code for this question. Simply take a look at the code below and critique the code as if you were reviewing a co-workers code. Mention any issues, bugs, or any other possible issues with the code that you can see.

Logger.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class Logger {

       public static final int DEBUG_LEVEL = 0;
       public static final int INFO_LEVEL = 1;
       public static final int ERROR_LEVEL = 2;

       public String fileName = "/tmp/output.log";
       private final FileWriter writer;
       private final ArrayList<String> logs = new ArrayList<String>();

       public Logger() throws IOException {
        writer = new FileWriter(fileName);
        logs.add("Initialised log\n");
       }

       private void log(String text) throws IOException {
```

```
        logs.add(text);
        writer.write(logs.get(logs.size()-1));
        writer.flush();
       }

       public void log(int level, String text) throws IOException {
        if (level == DEBUG_LEVEL) {
          log("DEBUG: " + text);
        } else if (level == INFO_LEVEL) {
          log("INFO: " + text);
        } else if (level == ERROR_LEVEL) {
          log("ERROR: " + text);
        }
       }
}
```

Question 2. Java EventService coding test

You have been supplied with a small set of Java code (question2.zip) that should include a *src*, *test* and a *libs* directory.

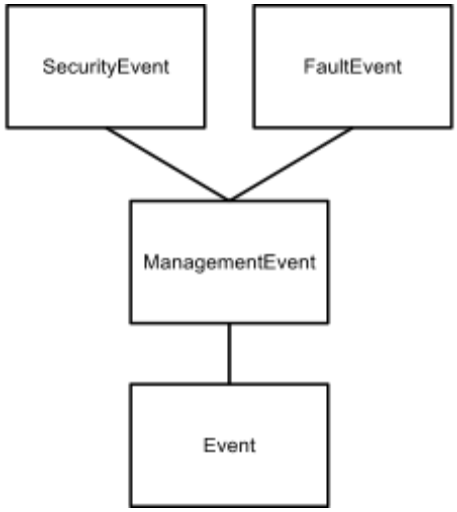
In the *src* directory there are the following files:

- Event.java
- EventListener.java
- EventService.java
- FaultEvent.java
- ManagementEvent.java
- SecurityEvent.java

In the *test* directory there is the following file:

- EventServiceTest.java

The basic principle behind this code is that there is an **EventService** class which allows classes that implement the **EventListener** interface to subscribe to Events.
There are two concrete classes of Events. They are **SecurityEvent** and **FaultEvent**.
Both of these classes implement the **ManagementEvent** interface which extends **Event**. The diagram to the right describes this relationship:



The EventServiceTest contains a test case with several tests in it. The tests are written using jUnit which is a unit testing framework for Java similar to other unit testing frameworks. The required jars that are required to run the junit tests are provided in the libs directory.

The EventService exposes 3 methods:

- subscribe
- unsubscribe
- publish

subscribe: This method is called to subscribe an EventListener to a specific Event which it can then receive notifications from the EventService when that Event is published.

unsubscribe: This method is called to unsubscribe an EventListener from a specific Event so that it no longer receives notifications from the EventService.

publish: This method is called with a specific Event and then all EventListeners that have subscribed to that event are then notified.

Task 1

Currently the unsubscribe method in **EventService** has not been filled out so events are never unsubscribed.
This task requires that you fill in the details for the unsubscribe method such that if an event is unsubscribed it will no longer receive notifications for that event.

You are also required to ensure that all tests in the **EventServiceTest** class pass. By changing the behaviour of the unsubscribe method this may change the expectations for the current tests.

Task 2

This task is a little more in depth and requires updating the *EventService* class to make it more flexible.

Currently it is only possible to subscribe concrete class event names to the *EventService* because you can only publish concrete Event Objects such as *SecurityEvent* or *FaultEvent*.

This task requires that you update the *EventService* so that if a non concrete *Event* class such as *ManagementEvent* or even the *Event* interface class are subscribed to by an EventListener then that event listener will be notified any time an event of that type is published.

For example:
If an EventListener subscribes to the EventService for management events and a SecurityEvent, which implements the ManagementEvent, is published then that event listener should be notified.

So the following code example should be possible:

```
EventService eventService = new EventService();
EventListener databaseListener = new DatabaseListener();
EventListener loginListener = new LoginListener();

eventService.subscribe(databaseListener, ManagementEvent.class);
eventService.subscribe(loginListener, Event.class);
eventService.publish(new FaultEvent(FaultEvent.EventLevel.LOW));

// the databaseListener and the loginListener should be notified of the
FaultEvent
```

It should also be noted that an Event listener should only be notified once of an event and should

not receive the same event twice in the one call to publish.

It is also important to understand that this should be a generic solution as more event types could be added later. Also think about the implication that this could make for unsubscribing events as well as it should be possible to unsubscribe non concrete class events.
eg. If a listener subscribes to a FaultEvent and then the a call to unsubscribe is called with ManagementEvent.class then that listener should be unsubscribed too.

It is also required that you update and add additional tests to check the new functionliaty that you have added.

Question 3. Future of web development

This question is a non-coding question. The idea is for you to express your ideas, opinions, and general thoughts in terms of where you see web development going. What you see as the exciting things and not so exciting things happening at the moment and what impacts they may have on future development projects.
There are no wrong answers here, just express yourself.