Questions

1 Object-oriented design, UML

You are to design a digital music manager application. There are songs, each with a title, artist, and album. For a given artist or album, it should be possible to get the corresponding songs. Songs may be stored in audio file formats such as MP3 or AAC. Play lists of songs are also needed.

Read both parts of this question before answering. You may assume here that an album has only one artist. State any further assumptions in your design.

(a) [3] Draw a well-designed UML class diagram to represent this information, and to address the scenario in part (b). Provide the basic abstractions, attributes, methods, relationships, multiplicities, and navigabilities as appropriate.

(b) [2] Draw a corresponding, well-designed UML sequence diagram for a scenario to play all the songs in the albums of a given artist.

2 Software engineering

(a) [2] Explain clearly the difference between *incremental prototyping* and *evolutionary prototyping*. Also, provide examples (not from the lecture notes) for the two approaches.

(b) [2] Explain how the *Liskov substitution principle* may be used to decide whether or not an inheritance relationship between two classes is appropriate. Also, provide an illustrative example.

(c) [2] What role do *executor* objects perform in a model-view-controller design (such as in the Wmvc framework). Why are executor objects useful?

(d) [2] What are *user stories*? Explain how user stories are used in extreme programming.

4 [3] Testing

Consider a `Rect` class to represent a rectangle in a two-dimensional plane.

```
public class Rect {
    …
    // create rectangle with given corners
    public Rect( Point topLeft, Point bottomRight ) { … }

    // return true iff point p is in or on the rectangle
    public boolean encloses( Point p ) { … }
}
```

You need to create a reasonably thorough set of tests for the `encloses()` method. Describe the equivalence classes of tests and the expected results. You do not need to implement the method, write test code, or use JUnit.