

ICS 52 - Introduction to Software Engineering
Midterm Exam #2 – Fall, 2008

Last Name: KEY First Name: _____

1	2	3	4	5	Total
---	---	---	---	---	-------

1. (10 points) There are some similarities and differences between a class in Java and the Software Engineering concept of an Abstract Data Type (ADT).

Identify two similarities between Java classes and ADTs.

5 points for each part.

Both . . .

- have an interface**
- have an internal state**
- hide implementation details**
- are based on information hiding**

Identify two ways in which a valid Java class might not be a true ADT.

It might have just functions, no internal data or state, like `java.util.Math`.

It might be a user interface element, such as a `Frame`.

It might have public data members, which violates the principal of information hiding and makes it not “abstract” (in the ADT sense).

It might use inheritance.

It might be abstract (in the Java language sense), and therefore incapable of being instantiated.

2. (20 points) Here's a rule of thumb: "Module interfaces should not reveal how the module is implemented."

Suppose a Java class `Student` has a method documented like this:

```
// Returns the total number of course units for the
// student that count towards graduation.
public int getUnits();
```

Does this method follow the rule of thumb, or not? Explain your reasoning.

10 points for each part.

Either yes or no accepted; most people took the easier "yes" road.

Select a desirable software quality discussed in lecture or in the textbook, and show how following this rule leads to software possessing that quality.

Lots of different possible answers. One example:

The quality of maintainability is enhanced by following this rule. If a module does not expose its implementations to other modules, then it is likely that the module can be modified without affecting its clients.

3. (30 points) Congratulations! In an alternate iTunes-free universe, you have been hired as a Software Architect for a new system that will allow users to download music and videos and play them on a PC or portable device (just like iTunes and iPod in our universe). This product will be called iHum. Your boss asks you to sketch out two software architectures for iHum, using two different architectural styles (but not using Pipes and Filters, which your boss loathes). For each approach draw a diagram (not a UML class diagram) showing the iHum architecture following that style. Make sure your diagrams are clearly labeled and clearly show a high-level design for iHum using each architectural style.

Name of first style: _____

15 points for each part; 5 parts for naming a style, 10 for diagram.

Common problems:

- **Diagram was not iHum-specific; for instance having a box labeled “Client” without indicating what an iHum client can do.**
- **“not layers” – to be a layered architecture, there should be coherent levels, each level should have a clear and consistent role in relation to the other levels – in particular the functionality at one level should be implemented by calling on the level below, each level should consist of multiple modules/components, the top level shouldn’t be the vague “UI” or “iHUM”.**

Name of second style: _____

4. (20 points) Define and contrast the terms *software architecture* and *software design*.

5. (20 points) The textbook says that high cohesion and loose coupling are desirable properties in a design and its subsequent implementation.

Why is high cohesion desirable? Your answer should be based on at least one software quality, principle, or property discussed in lecture or the textbook.

Why is loose coupling desirable? Your answer should be based on at least one software quality, principle, or property discussed in lecture or the textbook.