Question Sheet 1

Name: _____ ID: _____

These questions do not have a formal, definitive answer. They are meant to be food for thoughts. Feel free to seek answers on browsing the Internet, talking to other software developers or reading books.

1. We have seen that we use a class as a unit to group similar concepts. Is there a need for a larger granule of organization, such as a package?

2. There is no formal rule to partition classes among packages. Which criteria would you use?

3. What is a Java package? How are they named, stored, created?

4. Create a package `com`, which has a single class `B`, with the method `getOne()`. Next, create a class `C`, that has a `main` method that uses `B`.

5. Java uses an environment variable `CLASSPATH` to make classes stored in packages visible to other clients. This variable can be redefined, during compilation, via the tag `-cp`. Show the commands necessary to compile the class `C`, defined above.

6. What is the UML notation used to represent packages?

7. Java packages are released as `jar` files. What are these files? Show how to build a `jar` file containing package `com`. Show how to run class `C`, that is a client of `com`.

8. What are the advantages of using archives to release packages, such as the `jar` files used in the Java language?

9. The Reuse/Release Equivalnce Principle (REP) says that "The granule of reuse should not be smaller than the granule of release". What is reuse in this case? Is copy'n paste reuse? Is file copying reuse?

10. The Common Reuse Principle (CRP) says that "Classes that are reused together should be packaged together". Illustrate this principle with an example.

11. Classes describing buttons, labels and windows tend to be reused together. Can you give further examples of classes that should be reused together?

12. According to the CRP, the classes in the previous question should be packaged together. Why is it good to store them together in the same package?

13. The Common Closure Principle (CCP) says that "A change that affects a package should affect as many classes in the package as possible". Well, the best kind of package, in terms of maintainability, is a package that does not depend on anything, and that nothing depends on; however, giving that this scenario is unrealistic, the classes in the package should depend on the same interfaces and classes. Why?