Bryan Peterson
CSCI 360
MW 2:00 – 3:15
Professor Bowring
HMWK 3
January 30, 2007

P.250
8.8
a) Principle of Simplicity
b) Principle of Simplicity
c) Principle of Simplicity
d) Principle of Design with Reuse
e) Principle of Design with Reuse
f) Principle of Simplicity
g) Principle of Simplicity
h) Principle of Simplicity
i) Principle of Simplicity

8.9 Having a package that contains several classes, where a variable is accessible throughout the packages is not acceptable. Modules should shield the details of its internal structure and processing from other modules and (Principle of Information Hiding) and should not have access to unneeded resources (Principle of Least Privilege). Therefore, the variable should be made private with getter and setter methods. This way no module has direct access to the variable and the integrity of the data will be protected.

8.10

	Definition	**Basic Design Principle**
Modularity Principles		
Principle of Small Modules	Designs with small modules are better.	Principle of Changeability
Principle of Information Hiding	Each module should shield details of its internal structure and processing from other modules.	Principle of Changeability
Principle of Least Privilege	Modules should not have access to unneeded resources.	Principle of Changeability
Principle of Coupling	Module coupling should be minimized.	Principle of Changeability
Principle of Cohesion	Module cohesion should be maximized.	Principle of Changeability
Implementability Principles		
Principle of Simplicity	Simpler designs are better.	Principle of Economy
Principle of Design with	Designs that reuse existing	Principle of Economy

Reuse	assets are better.	
Principle of Design for Reuse	Designs that produce reusable assets are better.	Principle of Economy
Aesthetic Principles		
Principle of Beauty	Beautiful (simple and powerful) designs are better.	Principle of Changeability

8.11

Principle of Small Modules and Principle of Design for Reuse

This pair is mutually supportive because small modules can be designed for reuse. In fact, most modules that are designed for reuse will be small because they typically perform one action.

Principle of Design for Reuse and Principle of Beauty

This pair is mutually supportive because a module of code can ideally be designed to for reuse and have a solution that is simple and powerful such as the quicksort algorithm.

Principle of Cohesion and Principle of Simplicity

This pair is mutually supportive because cohesive modules are easier to understand because they have a single clear, logically independent responsibility or role. Everything that needs to be change din a highly cohesive module should be present in the module, so other modules are unaffected. Thus, highly cohesive modules are easier to code, test, debug, maintain, explain, and document all of which makes for a simpler design.

Principle of Coupling and Principle of Design for Reuse

This pair is mutually supportive because modularity is enhanced when coupling is minimized; thus, making solution easier to code for reuse.

Principle of Simplicity and Principle of Beauty

This pair is mutually supportive because an ideal solution to any problem should be simple, powerful, and easy to understand.