Lab-5 Specification: Building Defensive (AKA – Bullet-Proof) Classes

---

As we have previously discussed, the particular attribute values of an object define the current ***state*** of that object. The principles of encapsulation and information hiding assist the object in maintaining control over its state. However, the public interface of an object (i.e., constructors, accessors, mutators and other behaviors) is designed to allow users to create, access and mutate the object. Because of this necessary access, these methods (if not properly written) can allow direct access to the internal state of an object. If this occurs, and any of the attributes are mutable objects, this situation creates what is termed a ***brittle*** class. Brittle essentially means easily broken. Therefore, it is considered good programming practice to be proactive in this regard and implement what may be termed ***Defensive*** or ***Bullet-Proof*** classes.

The purpose of this exercise is to become acquainted with the concept of brittle classes and the methodology of hardening them to increase system integrity.

1. The first step is to understand the problems associated with brittle classes. The following classes are needed for this initial phase. They are contained in a package named ***lab5***:

    - Main.java
    - BrittleComputer.java
    - BrittleMotherBoard
    - BrittleCpu
    - BrittleMemory

2. Compile all of the classes. Execute the Main class with ***boolean TOGGLE = true***. Examine the output produced by the demo code. Write a detailed explanation as to why the output behaves as it does.

3. The next step is to modify the four ***Brittle*** classes into ***Defensive*** classes, using the naming convention used in the Main class. The Main class should not be modified (with the exception of the Boolean TOGGLE variable). The Shallow and Deep Cloning techniques discussed in lecture and lab should be used to implement the defensive nature of the new classes. Use the supplied Main class (with ***boolean TOGGLE = true***) to test your new classes.

4. The expected output from the Main class, once you have successfully implemented the bullet-proofing, is provided below.

===== Begin Bullet-Proof Testing =====

Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU

Model: Micron SDRAM (4 GB) - P/N: MD400SD

- Motherboard Configuration:
  * Processor - Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Computer Configuration ***
- Model: Dell PowerEdge 2100
- Motherboard Configuration:
  * Processor - Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Test #1 ***

Model: Broken (3.2 GHz) - P/N: GX9764CPU

Model: Micron SDRAM (0 GB) - P/N: MD400SD

*** Computer Configuration ***
- Model: Dell PowerEdge 2100
- Motherboard Configuration:
  * Processor - Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Test #2 ***

- Motherboard Configuration:
  * Processor - Model: Broken (0.0 GHz) - P/N: Broken
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Computer Configuration ***
- Model: Dell PowerEdge 2100
- Motherboard Configuration:
  * Processor - Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Test #3 ***

- Motherboard Configuration:
  * Processor - Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

- Motherboard Configuration:
  * Processor - Model: Broken (0.0 GHz) - P/N: Broken
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Computer Configuration ***
- Model: Dell PowerEdge 2100
- Motherboard Configuration:
  * Processor - Model: Intel 3200G (3.2 GHz) - P/N: GX9764CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Test #4 ***

*** Computer Configuration ***
- Model: Dell PowerEdge 2100
- Motherboard Configuration:
  * Processor - Model: Intel 5000G (5.0 GHz) - P/N: AZ4500CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

*** Computer Configuration ***
- Model: Dell PowerEdge 2100
- Motherboard Configuration:
  * Processor - Model: Intel 5000G (5.0 GHz) - P/N: AZ4500CPU
  * Memory - Model: Micron SDRAM (4 GB) - P/N: MD400SD

Grading Rubric: This assignment is worth 10 points and will be graded based on the following rubric:

Area	Exemplary	Competent	Developing	Points
Class Header	All header components are present, with references and comments that accurately support the state of the file.	All header components are present, but references and comments are incomplete or nonspecific.	Header is missing or only partially present, and references and comments are vague or unmeaningful.	__ / 1
Coding Style	Code implementation utilizes appropriate white space, self-documentation techniques and non-obvious comments.	Code implementation exhibits minor alignment or spacing problems, some comments are missing or redundant.	Significant alignment and spacing problems, comments are generally missing or sporadic.	__ / 1
Functionality / Creativity	All Brittle class source files have been modified to properly implement the Shallow and Deep cloning techniques.	Brittle class source files have been modified to implement the Shallow and Deep cloning techniques, but some minor errors or misconceptions exist.	Some Brittle class source files have been modified to implement the Shallow and Deep cloning techniques, but significant errors or misconceptions exist.	__ / 5
Discussion	Explanation of Brittle class operation is concise, correct and pertinent.	Explanation of Brittle class operation generally outlines the correct perspective, with some minor errors or misconceptions.	Explanation of Brittle class operation vaguely outlines the correct perspective, or contains significant errors and misconceptions.	__ / 3