HW0 Reflection

Jordan Ye

January 18, 2025

(1) **In retrospect, what could you have done better to reduce the time you spent solving this assignment?**

Looking back, I realize that I could have significantly reduced the time spent on the assignment by spending more time upfront to fully analyze the requirements and plan my approach. Specifically, I should have:

- Created a clear outline or flowchart of how each component (Ball, BallContainer, and Box) would interact before diving into coding.
- Invested time in reviewing similar examples and design patterns (such as composition versus inheritance) which are critical in structuring my code.
- Taken advantage of the warnings and documentation provided by the IDE to catch subtle mistakes in constructor assignments and method implementations early on.

(2) **What could the Principles of Software staff have done better to improve your learning experience in this assignment?**

In my opinion, the learning experience could be enhanced by:

- Providing more detailed examples that demonstrate the intended design patterns, especially for topics like composition versus inheritance. A few real-world scenarios illustrating these concepts would have been invaluable.
- Including a troubleshooting guide or FAQ section that highlights common pitfalls (for example, common mistakes in implementing constructors or handling collections) and how to avoid them.
- Offering a brief video walkthrough or annotated sample solutions for similar assignments could also help clarify the expectations and reduce confusion regarding key design decisions.
- The git and eclipse installation should have been more visual. It was difficult to read blocks and blocks of black and white paragraph with no visualization and was a struggle.

I think these additions would help us gain a better grasp of complex topics and reduce the learning curve.

(3) **What do you know now that you wish you had known before beginning the assignment?**

Before starting this assignment, I wish I had known more about the importance of thorough planning and the power of Java's built-in libraries. For example:

- I now understand the value of sketching out your class responsibilities and interactions on paper or using a diagram before coding. This would have minimized the time I spent iterating over poorly designed methods.
- I learned that familiarizing myself with the Java Collections API—in terms of its strengths and limitations (like handling duplicates and ensuring immutability through unmodifiable views)—can save significant debugging time.
- I also gained practical insights about effective debugging practices such as carefully analyzing IDE warnings and leveraging JUnit tests to guide incremental improvements. I also learned various commands to clean, build , test and run my project as well as some other parameters to see more information to debug.