EECS 293

Software Craftsmanship

2020 Spring Semester

Programming Assignment 10

Due at the beginning of your discussion session on

March 31-April 3, 2020

Reading

In addition to the following topics, the quiz syllabus includes any material covered in the lectures:

- Sections 8.2 (introduction only), 8.3, 8.4 ("Avoid throwing exceptions in constructors …", "Know the exceptions …", and "Consider alternatives …" only), and 8.5 in Code Complete.
- Items 9, 69, 73, 74, 75, 76, and 77 in Effective Java.
- Section 19.6 in Code Complete and the Quick Reference Guide on Routine Names on canvas.

Programming

The Better Software computer company has persuaded management to replace the Java Collections Framework with the Better Collections Framework. Better Software claims that the Better Collections Framework has many advantages over the original Java Collections. As a result, all teams are required to change the implementation of data structures to use the Better library. In particular, your team is tasked with replacing `Sets` (such as `HashSets` and `TreeSets`) throughout the existing code base to `BetterSets`. Although management is fully committed to the upgrade, your teammates are skeptical. The team suspects that the Better Collections may be significantly worse collections, especially in terms of reliability. Some preliminary tests showed that the `BetterSet` has a tendency of throwing exception inexplicably or of failing to follow the interface's contract. Your job is to prepare for the transition to `BetterSets` to make the upgrade as painless as possible.

Transition Sets

Create a new package `bettercollection` and with a `public final class BetterSet<T> implements Set<T>`. The idea is that your `BetterSet` is a placeholder that will eventually be replaced by the `BetterSet` provided by Better Software. Your implementation of `BetterSet` should be a simple stub that allows you to create a proof of concept of the new code base, and potentially to test it prior to receiving delivery of the new Better Collections.

As a test bed, replace all instances of `HashSet` and `TreeSet` in Programming Assignment 5 with your `BetterSet`. You should also create additional examples where `BetterSet` can be used.

Design

Develop a strategy to defend your code against potential problems in the `BetterSet`. Your design should account for any potential departure of `BetterSet` from the contract implied in the `Set` interface.

Submit a separate text file to document the error handling architecture that you will follow in your implementation. The architecture document should describe your error handling choices such as a strategy for implementing robustness and handling erroneous arguments, decisions on local or global error handling, error propagation through the code, presence and location of a barricade, and the other factors in the defensive programming checklist at the end of chapter 8.

Implementation

Implement your error handling architecture. Your implementation should be placed in a new package called `defensive` (under no circumstances it should be placed in the `bettercollection` package). You will be asked to demonstrate that your code follows your error handling architecture.

During the discussion, the discussion leader will replace your `BetterSet` with the final version, and test whether your approach is sufficiently defensive by running your test suite on Programming Assignment 5 and on any other test bed that you submit.

Submission

Create a repository called betterset.git where you will post your submission. Make small regular commits and push your revised code and test cases on the git repository. Your submission should contain:

- A separate text file to document your error handling architecture.
- Your implementation of the error handling architecture.
- A revised version of Programming Assignment 5 with `HashSet` and `TreeSet` replaced by `BetterSet`. You should also submit the test suite that you prepared for Programming Assignment 5.
- Any additional test bed that you created to evaluate `BetterSet`, along with an extensive test suite.

General Considerations

Your code should have a reasonable number of comments, but documentation is going to be the topic of a future assignment. As a general guideline at this stage of the course, comments should be similar to those accepted in EECS 132.

Discussion Guidelines

The project discussion will focus on defensive programming. In particular, you will be asked to demonstrate that your code follows your error handling architecture.