

Chapter 5. Franc-ly Speaking

Initial Problem

- \$5 + 10 CHF = \$10 if rate is 2:1
- \$5 * 2 = \$10
- Need to make "amount" private
- Concerns about Dollar side effects and Money rounding
- Methods: equals(), hashCode()
- Considerations: Equal null, Equal object
- 5 CHF * 2 = 10 CHF

Approaching the First Test

- Challenge
 - Big leap to write test implementable in one step
 - Need an object like Dollar to represent Francs
- Proposed Solution
 - Copy and edit Dollar test for Franc
 - Example test method: testFrancMultiplication()
 - Create Franc instance
 - Assert multiplication results
- Development Phases
 - 1. Write a test
 - 2. Make it compile
 - 3. Run to see it fail
 - 4. Make it run
 - 5. Remove duplication
- Design Philosophy
 - Early phases prioritize speed over design
 - Final phase enforces removing duplication and good design
 - Emphasis on completing all phases to maintain code quality

Summary of Progress

- Skippile "make it compile" due to short step to running code
- Duplication exists and must be eliminated before next test
- Invented smaller test to represent progress towards big test
- Used duplication and editing to write tests and model code
- Commit to removing duplication before finishing work

Implementation Details

- Franc Class
 - Private int amount
 - Constructor sets amount
 - times(int multiplier) returns new Franc with multiplied amount
 - equals(Object object) compares amounts after casting
- Duplication Issues
 - Dollar and Franc code duplication
 - Common equals and times methods needed