Chapter 14: Change

Currency Conversion and Money Reduction

Basic Conversion Tests
- $5 + 10 CHF = $10 if rate is 2:1
- $5 + $5 = $10
- Return Money from $5 + $5

Bank.reduce(Money)
- Reduce Money with conversion
- Reduce(Bank, String)

Test: Reduce Money Different Currency
- Create Bank instance
- Add rate CHF to USD as 2
- Reduce 2 francs to USD, expect 1 dollar

Initial Money.reduce Implementation
- Checks if currency is CHF and target is USD, rate = 2, else 1
- Converts amount by dividing by rate
- Problem: Money knows about exchange rates, which is undesirable

Refactoring to Move Exchange Rate Logic to Bank
- Pass Bank as parameter to Expression.reduce()
- Bank.reduce(Expression source, String to) calls source.reduce(this, to)

Expression Interface
- Money reduce(Bank bank, String to);

Sum.reduce Implementation
- Adds augend.amount and addend.amount
- Returns new Money with sum in target currency

Money.reduce Implementation
- Uses bank.rate(currency, to) to get rate
- Converts amount using rate

Bank Rate Calculation
- int rate(String from, String to)
- Returns 2 if from CHF and to USD, else 1

Eliminating Hardcoded Rate
- Store rates in Hashtable with Pair keys

Pair Class as Hashtable Key
- Contains from and to currencies
- Implements equals to check from and to equality
- Implements hashCode returning 0 (simple but inefficient)

Bank Rate Storage and Lookup
- Hashtable rates stores Pair to Integer rate
- addRate(String from, String to, int rate) adds to rates
- rate(String from, String to) retrieves from rates

Identity Rate Handling
- Test: rate from USD to USD returns 1
- Fix rate method to return 1 if from equals to

Techniques Introduced
- Added parameters in anticipation of needs
- Factored out code and test data duplication
- Tested Java behavior (array equals)
- Introduced private helper class without separate tests
- Forged ahead after refactoring mistake by writing new tests