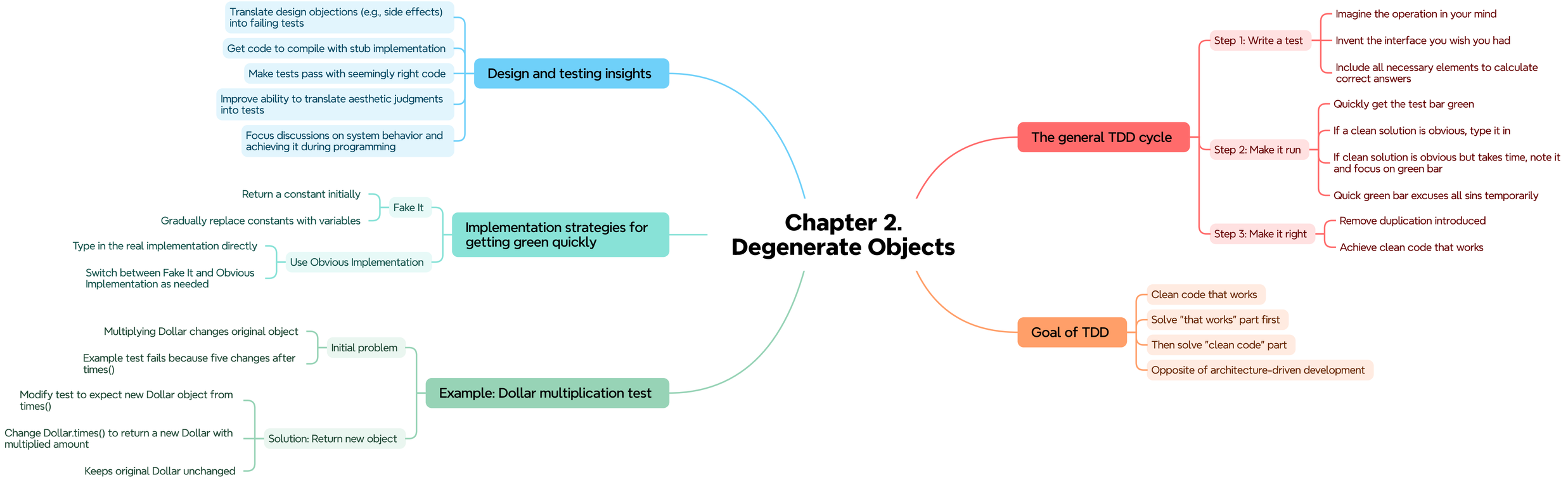
Chapter 2.
Degenerate Objects
The general TDD cycle
Step 1: Write a test
Imagine the operation in your mind
Invent the interface you wish you had
Include all necessary elements to calculate correct answers
Step 2: Make it run
Quickly get the test bar green
If a clean solution is obvious, type it in
If clean solution is obvious but takes time, note it and focus on green bar
Quick green bar excuses all sins temporarily
Step 3: Make it right
Remove duplication introduced
Achieve clean code that works
Goal of TDD
Clean code that works
Solve "that works" part first
Then solve "clean code" part
Opposite of architecture-driven development
Design and testing insights
Translate design objections (e.g., side effects) into failing tests
Get code to compile with stub implementation
Make tests pass with seemingly right code
Improve ability to translate aesthetic judgments into tests
Focus discussions on system behavior and achieving it during programming
Implementation strategies for getting green quickly
Fake It
Return a constant initially
Gradually replace constants with variables
Use Obvious Implementation
Type in the real implementation directly
Switch between Fake It and Obvious Implementation as needed
Example: Dollar multiplication test
Initial problem
Multiplying Dollar changes original object
Example test fails because five changes after times()
Solution: Return new object
Modify test to expect new Dollar object from times()
Change Dollar.times() to return a new Dollar with multiplied amount
Keeps original Dollar unchanged