Levi George

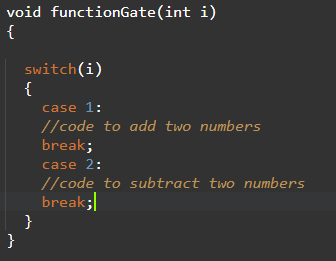
CS 36000: Software Engineering

11/21/2020

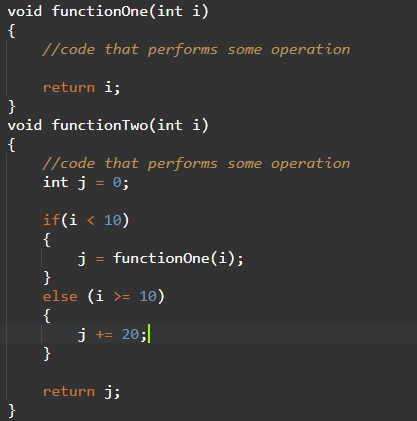
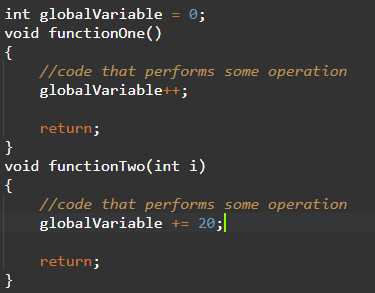
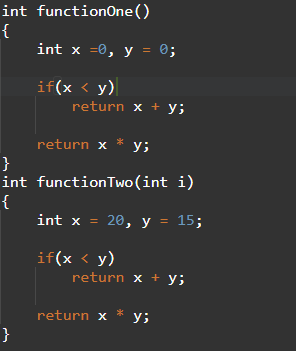
Prof. Venkata Inukollu

Homework 3

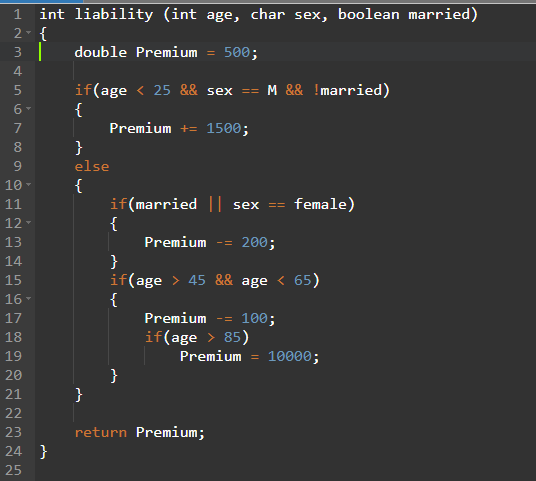
1. Functional Cohesion
   1. A form of cohesion in which a function has only one task it performs
   2. Example: A function which may perform a single action, like a basketball player having the function shootBall(). It performs a single function, this would not be true for a function like aimAndShootBall(), which performs both aiming and then shooting the ball.
   3. Pros/Cons
      1. Pros: Highly reusable and easy to maintain
      2. Cons: No downsides to usage
2. Communicational Cohesion
   1. A form of cohesion in which a single function does several actions related to the process it must perform as a software product, but it also performs all these operations on the same data.
   2. Example: A function which performs multiple related actions on a single piece of data. This could be like a function which takes data in and then displays the data and applies encryption on it all at the same time.
   3. Pros/Cons
      1. Pros: Better than Logical or Coincidental Cohesion, acceptable in some situations, but not desirable
      2. Cons: Still not as reusable as Functional
3. Logical Cohesion
   1. A form of cohesion in which a single function performs several logically related tasks, but only one is performed at a time and this is determined when the function is called.
   2. Example: This would involve a function that takes a parameter which is used to determine which part of the function that runs, this parameter could be used in a switch statement with different sections of code.



* 1. Pros/Cons
     1. Pros: None this is a bad principle to follow
     2. Cons: Hard to understand functions, not reusable

1. Content Coupling
   1. A form of coupling in which a function will directly reference another’s content
   2. Example:
   3. 
   4. Pros/Cons
      1. Pros: None, this is a bad principle
      2. Cons: Any changes made in one function must be reflected in the other, hard to maintain.
2. Common Coupling
   1. A form of coupling in which two functions reference a global variable
   2. Example:
   3. 
   4. Pros/Cons
      1. Pros: None, this is a bad principle
      2. Cons: It is hard to dissect errors in code and find source of bugs, repeated usage of a single variable that is accessible throughout program is poor for security.
3. Stamp Coupling
   1. A form of coupling in which a function will process a data structure, but only partially
   2. Example: A function which will check an integer array for items greater than 20 and then add 5 to them. This will not guarantee that all or any items are edited, thus it exemplifies a usage of a data structure where it is “stamped”.
   3. Pros/Cons
      1. Pros: None, this is bad programming
      2. Cons: It is hard to figure out where an error occurred, it isn’t reusable, more data than needed is used, allows crime, consumes computer resources
4. Mutation Testing
   1. A form of testing in which a system will undergo minor changes and these changed versions (called mutants) undergo tests in which they are tested for serious deviations in behavior from the original. This is done to ensure that the tests can determine if the software has changed and detect major changes in behavior.
   2. Example: A mutant testing system may change values, decisions, or statements. So, in the below image, a mutant system may change function one into function two. After that change, it would then proceed to run tests to check for major impacts on system behavior.
   3. 
   4. Pros/Cons
      1. Pros: Great coverage of system and tests to ensure accuracy of tests,
      2. Cons: Time expensive and resource consuming, hard to perform without automation.
5. Regression Testing
   1. A form of testing in which tests are executed twice, once to test a system and the second time to ensure that changes occurring to the system do not affect the system’s behavior negatively.
   2. Example: I might rewrite a function which adds two numbers and then test it to make sure it produces a certain number, once I change it I would run that test again to ensure that it produces that specific number again.
   3. Pros/Cons
      1. Pros: Can be used during integration testing, assisting with interface and interaction testing. Ensures bugs don’t re-emerge.
      2. Cons: Can be hard to use regression testing without automation. High-skill requirement to use effectively. High impact on budget.
6. Iterator pattern
   1. A design pattern in which one produces a way of accessing elements of a container or data structure without knowing its detailed implementation.
   2. Examples: This would be like having a map to a city, it allows you to search the individual elements of the container, relating to the city you can check each building in the city with the map.
   3. Pros/Cons
      1. Pros: Less code and mess from designing multiple container iterations
      2. Cons: Heavy-handed for systems with simple collections being used.
7. Decorator pattern
   1. A design pattern in which extensions to objects or classes are placed within wrappers and then placed around the object to be extended.
   2. Example: Much like a Russian nesting doll, the original doll is placed within other dolls. The original object is placed within or with a wrapper object around it, these wrappers extend the design of the original object.
   3. Pros/Cons
      1. Pros: It becomes easier to discover issues in wrappers by determining the problem and what operation it occurs during. Wrappers allow for extension of classes without subclasses.
      2. Cons: Numerous additions can lead to a complicated or long trail of wrappers. It can become hard to remove wrappers later in implementation while retaining certain wrappers.

Criterion Coverage Example



1. Statement Coverage
   1. Test Sets
      1. T\_1 = < age = 24, sex = M, married = false>
      2. T\_2 = < age = 25, sex = F, married = false>
      3. T\_3 = < age = 45, sex = F, married = false>
      4. T\_4 = < age = 64, sex = F, married = false>
      5. T\_5 = <age = 66, sex = F, married = false>
   2. Coverage = 1.0
      1. Sc: 8
      2. Si: 2
         1. Age can’t be greater than 85 and less than 65, thus lines 18 and 19 are inaccessible.
      3. Se: 10
   3. Is T Adequate – Yes
2. Block Coverage
   1. Test Sets
      1. T\_1 = < age = 24, sex = M, married = false >
      2. T\_2 = < age = 26, sex = F, married = false >
      3. T\_2 = < age = 46, sex = F, married = false >
   2. Coverage = 1.0
      1. Bc: 4
      2. Bi: 1
         1. Age can’t be greater than 85 and less than 65, thus lines 18 and 19 are inaccessible.
      3. Be: 5
   3. Is T Adequate – Yes
3. Branch Coverage
   1. Test Set
      1. T\_1 = <age = 24, sex = M, married = false>
      2. T\_2 = <age = 26, sex = F, married = false>
      3. T\_3 = <age = 46, sex = F, married = false>
   2. Coverage = 1.0
      1. Branches Covered = 3
      2. Branches infeasible = 1
         1. Age can’t be greater than 85 and less than 65, thus lines 18 and 19 are inaccessible.
      3. Branches Total = 4
   3. Is T Adequate - Yes
4. Condition Coverage
   1. Test Set
      1. T\_1 = <age = 24, sex == F, married = false>
      2. T\_2 = <age = 25, sex == F, married = false>
      3. T\_3 = <age = 26, sex = F, married = false>
      4. T\_4 = <age = 46, sex = F, married = true>
      5. T\_4 = <age = 46, sex = M, married = false>
      6. T\_5 = < age = 24, sex = M, Married = true>
   2. Coverage = 1.0
      1. Conditions covered = 7
      2. Conditions Infeasible = 1
         1. Age can’t be greater than 85 and less than 65, thus lines 18 and 19 are inaccessible.
      3. Conditions Total = 8
   3. Is T Adequate - Yes
5. Decision Coverage
   1. Test Set
      1. T\_1 = <age = 24, sex = F, married = false>
      2. T\_2 = <age = 26, sex = male, married = true>
      3. T\_3 = <age = 46, sex = male, married = false>
   2. Coverage = 1.0
      1. Decisions Covered = 3
      2. Decisions Infeasible = 1
         1. Age can’t be greater than 85 and less than 65, thus lines 18 and 19 are inaccessible.
      3. Decisions Total = 4
   3. Is T Adequate – Yes