SE2811 Exam 1 Feedback

Page 2, Problem 1.

① -1 In cohesion, a single class has a single role, rather than multiple related classes being together.

② -1 Poor cohesion is one class with many roles, not one role spread over many classes.

Page 2, Problem 2.b.

① -1 Programming to an interface happens in the PositionSubject class, not in the PositionObserver interface or the concrete observers.

② -2 The “interface” being programmed in this case is the PositionObserver, not the state being passed to the subject.

③ -2 The answer should talk about interfaces.

④ -1 Although the answer seems to discuss the PositionSubject using an interface, it is not clear.

⑤

Page 2, Problem 3.a.

① -1 Seems to assume that only one observer can be attached.

②

③

Page 2, Problem 3.b.

① -0.5 missing the observer to add in the header

② -1 Adding coupling to concrete observers

③ -3 missing body of method entirely

④ -1 Use observers.contains(o) to determine if o is already in observers.

⑤ -0.5 The name for the list of observers is specified in the UML diagram as “observers” on the line connecting the PositionSubject to the PositionObserver. This is an instance variable.

Page 3, Problem 4.a.

① -0.5 Where first thread stops not clearly stated – seems to leave open the possibility that the first thread will stop after having created the object before the second starts.

② -2 Answer is quite vague

Page 5, part a.

① -1 “Wrong” pattern presented without completely convincing argument

② -3 “Wrong” pattern presented with an argument that does not address customer’s requirements.

③ -1 Right strategy used with wrong name

④ -1 NO pattern used without completely convincing argument

Page 5, part b.

① -0.5 Missing stereotypes that describe roles in the pattern, such as <<concrete behavior>> or <<abstract behavior>>.

② -1 Wrong arrowhead for extends/implements

③ -2 Simple Factory Idiom implemented without an abstract method. Singleton without static getInstance

④ -0.5 Missing name for member variable relationship (e.g. composition, aggregation, or “plain vanilla” use). (No penalty if getBooks accepts a reference, so a member variable isn’t needed.)

⑤ -0.5 Missing implemented methods in a derived class.

⑥ -1 Illustrate multiple classes, one responsible for title search, and one responsible for author search.

⑦ -1 Program to interfaces, not concrete instances. The Library should hold a reference to the Lookup class, not to the TitleLookup or AuthorLookup classes. (-1 for lines being drawn wrong, even if the rest is OK)

⑧ -1 Arrow points wrong direction

⑨ -2 Strategy/Observer pattern with two methods, so that we cannot switch between the two methods.

Page 5, part c.

① -1 Keep the loop in Library#getBooks. Then you don’t have to repeat the loop code in all of the lookupBehavior classes. You can keep the loop by implementing the Strategy pattern with an “Equals” strategy instead of a “Lookup” strategy.

② -0.5 Keep the book. The user will still need a mechanism to provide the information about the book he/she is looking for.

③ -0.5 Missing argument to strategy method as defined in the interface in part b.

④ -2 Provided solution ***increases*** coupling with various search methods, e.g. by including the names of the concrete classes in the implementation of the method.

⑤ -2 Provided solution puts both comparisons in the same method, ***decreasing cohesion*** as new lookup strategies are added.

⑥ -3 Provided code does not actually solve what the user was looking for. (It does not allow searching by either title or author.)

Page 6, Problem 1

① -0.5 “all other” instead “the other”

-0.5 almost there but the “finish” idea not clearly communicated

-2 idea of thread stopping is there, but not of waiting for it

-3 idea of waiting or stopping other thread absent, other correct statements present

-5 answer is wrong but displays some knowledge of threads

Page 6, Problem 2

-0.5 Missing t2.join() or second wait() if needed (and if t1.join() is used).

-0 missing void return type

-1 two nested anon inner classes instead of one

① -0.5 Better to use join() than notify/wait.

② -0.5 wait() must be in a synchronize statement to avoid an IllegalMonitorStateException

③ -1.5 missing join() entirely

④ -1 making Thread into

⑤ -0.5 thread2. missing from use of join()… or “this” is used instead

⑥ -0.5 Using run instead of start

⑦ -0.5 Minor syntax errors in lambda expression