SWEA Iteration 7: Blackbox Testing and Pattern Hunting

<<Group name>>
Computer Science, University of Aarhus
8200 Århus N, Denmark
<<Names>>

<<Date>>

1 EC of RhoStone 'attack()'

1.1 Conditions

Regarding subquestion 1, the following conditions influence function attack() in RhoStone:

- is the attacking player owner of the card (boolean condition)
- is card active (boolean condition)
- *(Your group's further analysis here)*

1.2 EC tables

These leads to a first version of the *equivalence class table*:

Condition	Invalid ECs	Valid ECs
is owner of card	false [a1]	true [a2]
(more analysis)...	...	...

The found ECs have the *representation* and *coverage* properties because [argument here].

1.3 Extended test case table

We use Myers heuristics for valid and invalid ECs to generate test cases as outlined in the extended test case table:

ECs covered	Test case	Expected output
[a2][...]	Fin (in turn): Ts (active, Fin) → Hero (Ped)	STATUS_OK
(more)	...	...

Legend: The symbol → means "attack"; Fin and Ped are Findus and Peddersen respectively. Cards are from DeltaStone and abbreviations are: "Ts" for card Tomato Salad (attack 3, health 2), ...

2 Facade

[Argue whether the Game interface is a FACADE pattern or not.]

3 Observer

[Include the source code of the usePower() method in your HotStone, highligthing the code lines that trigger observer calls; shortly argue for what is going on]

[Include a JUnit test case for a scenario in which the Danish Chef uses his power to successfully field a Sovs minion; and shortly argue for the assertThats which verifies that all proper observer events are notified from the Game]

4 Transcript

4.1 Pattern identified

[Argue for a pattern that allows transcription to be made using a purely "Change by addition" technique]

4.2 Implementation notes

[Include and explain code fragments that demonstrate how A) transcription is made B) how transcription is turned on and off at run-time]

5 WizardHub Effect Library

5.1 Adapter Design

[Include an UML diagram focussing on how the library class EffectWizard, interface HotStoneGameTarget, and your private interface for your HotStone game are part of an Adapter pattern design.]

5.2 EtaStone card effect code examples

[Include and explain code fragments that demonstrate how the two or three card effects are coded using the WizardHub library.]

5.3 Adapter code examples

[Include and explain code fragments that demonstrate how two or three methods (involved in the card effects described above) of your adapter code is implemented.]

6 Role diagram

[Include your role diagram and a short explanation]

7 Backlog

The following features and requirements are still not implemented in our Hot-Stone software:

- …
- …