SWEA Iteration 6: Compositional Design and Test Spy

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

<<Date>>

1 Private Interfaces and ISP

1.1 Private Interface Design

[Describe your improved design using private interfaces to handle internal mutation of domain abstractions Game, Card, and Hero]

1.2 Refactoring

[Show code before the refactoring; and next the refactored code based upon private interfaces, for the Game domain abstraction.]

2 EtaStone

2.1 Design and UML

[Sketch a compositional design using UML for EtaStone which uses a role interface]

[INCLUDE UML DIAGRAM HERE - A GOOD PICTURE OF HANDRAWN IS OK.]
[Argue for a single Role Interface that covers both Hero Powers as well as Card Effects.]

2.2 Unit Testing using Test Spy

[Include test code for ensuring correct behavior of the effect of one of the EtaStone cards]
[Include relevant code fragments of your Test Spy to support the above test case]
[Argue that you are doing Unit testing and not Integration testing in the above shown code fragments]

3 SemiStone

3.1 Configuration Table

[Fill in Table 1., similar to the table from FRS §17.2]

Table 1: HotStone configurations

	Variability points		
Product	**Mana Prod.**	**Winning**	**...**
AlphaStone	3 every round	Findus/round 4	-
BetaStone	+1 pr round	Defeat opponent	-
GammaStone	-	-	-
...	-	-	
SemiStone	-	-	-

3.2 SemiStone Code Configuration

The SemiStone variant is configured in our code like this ...

[Provide production code fragment(s) that show the code that configure the GameImpl for the SemiStone variant]

The design of the HotStone system, with emphasis on the SemiStone variant, is shown by the following UML diagram:

[Include the UML diagram of all interfaces and classes in HotStone related to variant handling—but show only the associations between the SemiStone abstract factory implementation to its products, *not* all the other associations between concrete factory classes.]

4 Parametric 'getWinner()'

The method getWinner() in GameImpl would look like this if a purely parametric design had been employed as variant handling technique in the HotStone code:

[Provide (pseudo) code fragment(s) the show how a parametric design could be implemented]

5 Backlog

- ...
- ...