SA / Assignment 4

1. Towers of Hanoi

- Take your class diagram from the last assignment.
- Make sure, you have at least one 1..n relationship and 2 classes. Collapse classes which are associated with a 1 to 1 relationship into one class or consider upgrading the relationship to 0..1 to 0..1 or even 0..1 to n. Replace all 1s with 0..1.
- If this does not work for your diagram, talk to me (soon!) to help you fix it.
- Implement it in Java. Take special care to achieve bidirectional references and referential integrity. Ignore your methods (only implement the implicit methods: getters, setters and reference updates).
- implement Disc:move(destination: Tower) with all its check (no bigger disc on smaller). Take, adjust, or write two scenarios – one showing a disc being moved once and one showing a disc being moved to a wrong tower with a too small disc and the move being rejected. Implement the corresponding two unit tests.

2. Mancala

- Take your class diagram from the last assignment.
- Make sure, you have at least one 1..n relationship and 3 classes. Collapse classes which are associated with a 1 to 1 relationship into one class or consider upgrading the relationship to 0..1 to 0..1 or even 0..1 to n. Replace all 1s with 0..1. Make sure you have an opposite_of and successor relationship between the houses.
- If this does not work for your diagram, talk to me (soon!) to help you fix it.
- Implement it in Java. Take special care to achieve bidirectional references and referential integrity. Ignore your methods (only implement the implicit methods: getters, setters and reference updates).
- Implement methods for House:takeOppositePebbles and take, adjust, or write one scenario for this case. Implement a unit test based on this scenario to test the method.

- Implement a method House:ReDistributeCounterclockwise (this shall just take the pebbles and redistribute them, no check where the last ends up is necessary). Take, adjust, or write a scenario taking 3 pebbles out of a house and write a corresponding unit test.

For both tasks, submit the adjusted class diagrams, selected or adjusted scenarios, class implementation and unit tests as two different IntelliJ projects, and a very short manual how the tests can be started.