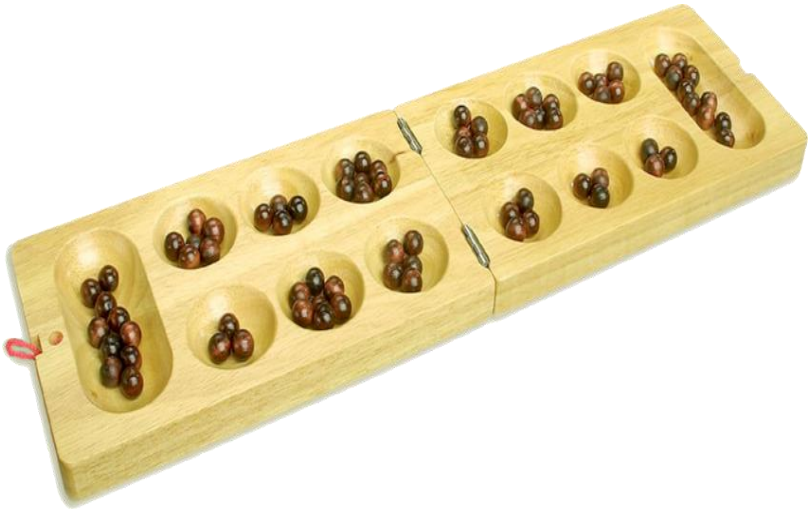
Mancala is the generic name for a family of two-player turn based strategy board games whose origin goes as far back as Ancient Egypt. The game is known under many different names. It is played with small beans, stones, or seeds and two rows of holes in a board, or in the earth, where the objective is to capture as many of the beans as possible and win the game if you have more then your opponent when the game ends. So, the game is played with two players and in complexity can be compared to (US) checkers.



Rules of the game:

- Start with 4 stones in each bowl except the Kalaha.
- Players take turns in making a move.
- A move is: pick a bowl that belongs to you (bowls on your side), take all its stones and distribute them one at a time, anti-clockwise.
- While distributing, each players include his/her own Kalaha, but not the opponent's Kalaha.
- If a move ends in that player's Kalaha, he/she can move again.
- If the last stone is laid in an empty bowl on your own side, the player can take that stone and all stones in the opposing bowl and put them in your Kalaha.

- The game ends when all bowls of the player who has the turn are empty (so this player is unable to make a turn) and all remaining stones left on the board go to the opponent.
- The winner is the player with the most stones.

Domain Exploration:

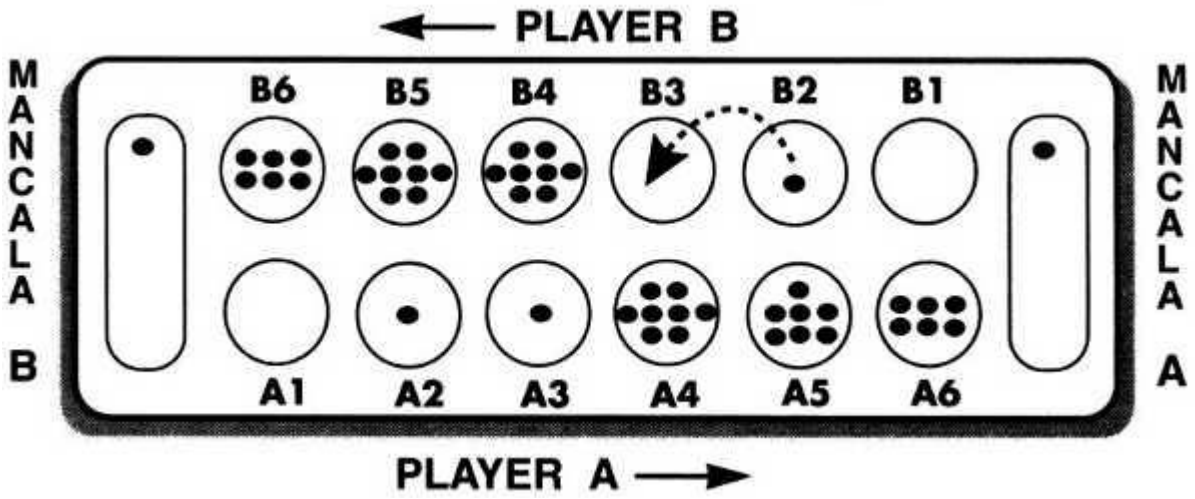
- Play some games to really understand the domain.

Group Assignment:

- Create the needed scenarios/stories needed to implement the domain (business logic) part of this Mancala game.
- Create your initial long list of Class Candidates.
- Optionally create a set of CRC-cards (Class-Responsibility-Collaboration) for these Class Candidates.
- Create Sequence Diagrams for each of your scenarios/stories.
- Create a Class Diagram that represents the structure of your domain.
- If you think it adds value you can also think of creating an Object Diagram here.

Individual Assignment:

- Implement the domain of Mancala in a Test-Driven Development manor.



Some TIPS to think about:

- Think about scenarios first
- Play scenarios (CRC cards)
- Rules of inversions (Active/Passive & Time (backtracking))
- Expert Pattern? NO GOD object that knows everything!
- Take time to think about **naming** objects and responsibilities!

Some OO rules to keep in mind:

- Objects manage their own state
- Objects do what they are asked to do
- Objects are selfish (don't do things that are not their responsibilities, no GOD objects)
- Objects collaborate with other objects to get the job done
- Objects don't expose their internals
- Objects should separate different concerns (high cohesion and low coupling)