Explanation of Design Patterns

The design patterns we used:

**Information Expert**: We assigned the GameGui class as an information expert. The GameGui class updates the players’ scores and announces who is the winner at the end of the game.

**Creator**: Creator pattern has been used for several classes. For instance, objects of robot class and token class are created in the board class. Therefore, board is the creator of robots and tokens since board contains the robots and tokens.

**Low Coupling**: The creator pattern, where different objects are created, support our low coupling pattern. The classes are designed in such a way that making changes to one class will not impact another class enormously. Thus, the model maintains low coupling.

**High Cohesion**: We created several classes for game pieces. Also, assigned each class of the model with specific responsibilities in order to keep objects strongly related and focused to a particular function. Hence, high cohesion is maintained.

**Controller:** We have used a controller, which controls all the settings input and bid inputs from the players and responses accordingly.

The design patterns we could use:

**Indirection**: We could use indirection which would allow us to create an intermediate object to assign specific responsibility. This would help us in the long run during maintenance and re-use without losing low coupling and high cohesion.

**Command**: We could use the command pattern. This would allow us to assign responsibility for responses to small distinct user commands such as selecting object, starting a new game, and loading a saved game etc. This would help us to maintain high cohesion and low coupling during maintenance afterwards.