AI Mancala Code Review

Are the variable and method names meaningful and descriptive? Give specific examples to support your observation. How could the variable and method names be improved?

The AI used the exact method names and param names I provided it from the javadocs. This is evident throughout the whole generated solution as it is 1-to-1 with the naming convention. A great example of this is the Board.java class where it uses the exact names provided by the javadocs. No improvements would be needed.

```java
// Method to set up pits
public void setUpPits() {
    // Implementation to establish 12 empty pits
    // ...
}

// Method to get the list of pits
public ArrayList<Pit> getPits() {
    // Implementation to return the list of pits
    // ...
    return pits;
}

// Method to get the list of stores
public ArrayList<Store> getStores() {
    // Implementation to return the list of stores
    // ...
    return stores;
}
```

Does the code follow coding conventions and formatting standards? Has it made appropriate use of includes? Are static members used properly? Give specific examples of how you would improve the coding conventions used.

Yes it follows the standard coding standards when it comes to comments, tab spacings, bracket placements and also arithmetic spacing. It imported all needed libraries to complete it's needed tasks. All access modifiers are appropriate for their usage. The AI all around did very well when it came to sticking to the standards of coding.

```java
package mancala;

public class Pit {
    private int stoneCount;

    public Pit() {
        this.stoneCount = 0;
    }

    public void addStone() {
        this.stoneCount++;
    }

    public int getStoneCount() {
        return this.stoneCount;
    }

    public int removeStones() {
        int removedStones = this.stoneCount;
        this.stoneCount = 0;
        return removedStones;
    }

    @Override
    public String toString() {
        return "Pit: " + this.stoneCount + " stone(s)";
    }
}
```

Are the classes properly encapsulated? Are member variables private? Are accessor and mutator methods used? How could you improve the encapsulation of this code?

All classes, member variables and even accessor and mutator methods have proper access modifiers. A great example of this is how all the classes are public but the member variables in side each class are private. The accessors and mutators in all classes are also public as they need to be accessed by other files. The following is a code example.

```
public class MancalaGame {

    private ArrayList<Player> players;
    private Player currentPlayer;
    private Board board;

    // Constructor
    public MancalaGame() {
        this.players = new ArrayList<>();
        this.currentPlayer = null;
        this.board = null;
    }

    // Method to set players for the game
    public void setPlayers(Player onePlayer, Player twoPlayer) {
        // Implementation to set players for the game
        // ...
    }
...
```

Is there any duplication of code in this project? Are there methods that do essentially the same thing, or parts of the same thing that could be made into smaller methods?

No duplication of methods or any other variables. It references the javadocs really well and provided great skeletons for methods it had trouble implementing.

Does each class and method have a single, obvious purpose or responsibility? Are there any long methods that should be broken up into smaller methods? Give specific examples of how you could improve the code with respect to responsibilities.

Each class has a very distinct purpose and responsibility as provided by the javadocs. No long or repeating methods. Personally, when I began implementation of methods I decided to break multiple methods up into their own as the single responsibility principle was getting blurry. This is evident in my own solution.