The BugsZero Kata

Bugs are optional, they sneak into our code much thanks the design choices we do, or fail to do. Let's try refactoring some existing code to make it more bug-repellent!

You'll practice reading code, looking for parts where it is likely that developers would create a bug if they extended the code. Whenever you've found such a weakness in the design your challenge is to strengthen the design in order to make that kind of bug very unlikely, or even impossible!

Repo and languages

This repository comes packaged with code that is totally tested, using the Golden Master technique. Pick the language of your choice in this repository or contribute another one. In case there is none you're comfortable with you can always try to do this without tests using any of the available languages. Beware that it is not identical as we've modified the code to remove some of the noise that wasn't relevant to sources of bugs.

Procedure

1. Identify a weakness in the design - now that's where we're helping you.
2. State what the potential bug is **before explaining your solution**. This is important, simply saying another solution is better avoids thinking of **why** it is actually better.
3. Explain or refactor the code to show the new design.

The purpose is to experiment with design alternatives, so it is not necessary to refactor to perfection, and it is optional to use tests. The purpose of first explaining the problem in the design is to make sure that the new design is not just different but also adresses one or more specific problems.

Existing bugs and weaknesses (in trivia)

Beware that this list, while necessary, spoils part of the fun :)

- **A Game could have less than two players** - Could we make sure it always has at least two. Is this a runtime check or can this be enforced "statically"?
  - What can be done in a language like javascript?
- **Existing BUG**: A player that gets into prison always stays there.
  - Other than just fixing the bug, try to understand what's wrong with the design and fix the root cause

- **Introducing new categories of questions seems like tricky business**. There are business rules: categories are evenly distributed on the board, there are an equal amount of each category.
    - Could you make sure all places have the "right" question and that the distribution is always correct (i.e. same number of each category, distance always the same between categories)?
- Someone could try to create a game with 7 players, can that be made impossible?
    - Alternatively allow for 7 players or more
- Existing BUG: coins are added to the wrong player. Try to understand what made this bug likely and fix the design so that it becomes very unlikely.
- The deck could run out of questions
    - Make sure that can't happen (a deck with 1 billion questions is cheating :)
- Similarly changing the board size greatly affects the questions distribution. i.e. how do we ensure that there are as many of each category and that they repeat in a cyclic way?
    - Is there a design where it is guaranteed that the question category distribution stays the same