Ask me anything

0 questions
0 upvotes
Why do people like the waterfall model?

- easier to "understand" -- more intuitive

- It's supposed to show how far into the project we are

- Because they misread the paper it was described in

Test cases shouldn't have that many asserts
How would you refactor the `testState()` method?

```java
@Test public void testState() {
    assertEquals(game.get('a', '1'), ' ');
    assertEquals(game.get('c', '3'), ' ');
    game.set('c', '3', 'X');
    assertEquals(game.get('c', '3'), 'X');
    game.set('c', '3', ' ');
    assertEquals(game.get('c', '3'), ' ');
    assertFalse(game.inRange('d', '4'));
}
```

- **reorder**: NO
- **Test cases shouldn't have that many Asserts, split it into different tests**: YES
- **It tests multiple things at once, maybe divide it into multiple tests**: YES
- **Separate into multiple test cases, one test cases to check if the game is initialized properly, one to set it, and one test case to check if the board is out of the range.**
- **in the assertEquals the "actual" and "expected" parameters are the wrong way around**: NO
- **seperate one for range**: NO
Why is it preferable to build up a String using a StringBuffer rather than simply concatenating String instances?

- because of the memory..
- concat always produces a new string instance
- Stack overflow
public Player() {
    this();
}

- It doesn't do anything
- The constructor calls itself
- Nothing...
- Infinite loop
- Calling this(); like a method is weird
- This is a constructor that calls itself, which will then call itself -> infinite recursion
- Runs itself but there is no such function except maybe the constructor
What's wrong with this code?

- Seems good to me
- player doesn't have a parent, so super is not defined

```java
public Player() {
    super();
}
```
The TicTacToe invariant says nothing about the gameState variable. What could we add?

protected boolean invariant() {
    return (turn == X || turn == 0)
        || (!this.notOver() ||
            this.winner() == player[X]
            || this.winner() == player[0]
            || this.winner().isNobody())
        || squaresLeft < 9
        || turn == X && this.winner().isNobody());
}
How can we make clear in the code that new Player() represents “nobody”?

- **new Nobody class inheriting from Player**
- **use a method that produces a null object**
  `winner = new Player(); // = nobody();`
- **We could maybe introduce a NullPlayer-class that does absolutely nothing apart from being a placeholder for the future winner**
- **how would we implement the nobody() method?**

```java
class TicTacToe {
    protected Player winner = new Player(); // = nobody();
}
```
Last chance for questions