Anonymous code review

1. Grading

1.1. Source code quality

- hashCode()'s that return just 42 are not good for any reason. Replace them with Objects.hashCode(field1, field2, field3)
- A couple of methods are really large and have a high cyclomatic complexity.
- Good use of the state pattern and other patterns.
- Getters and setters are often placed in the beginning instead of at the end. Since getters and setters are almost always the same it is better practice to put them at the bottom of a file and put the more important methods first. This will create better code readability.

1.2. Testing

Element	Class, %	Method, %	Line, %
board	100% (28/28)	90% (126/140)	91% (607/665)
exceptions	100% (1/1)	100% (1/1)	100% (2/2)
extensions	100% (6/6)	40% (18/45)	65% (91/138)
logger	87% (7/8)	87% (21/24)	77% (58/75)
player	100% (5/5)	100% (31/31)	100% (86/86)
states	100% (11/11)	72% (34/47)	78% (187/237)
utility	90% (10/11)	88% (45/51)	84% (106/125)
Game	100% (1/1)	53% (7/13)	35% (13/37)

The codebase is tested very well. It has a line coverage of 84% (> 75%). Also, all tests pass on Windows as well as on OSX. The test classes look good. Usually there is only one assert of verify per method, there is a good use of class hierarchy and everything looks very structured and clear.

1.3. Code readability

1.3.1. Formatting

Most of the code is formatted correct. Indentation is correct everywhere, but there are a lot of comments which line length is greater than 80 characters. This is not problematic, but it could be better.

1.3.2. Naming

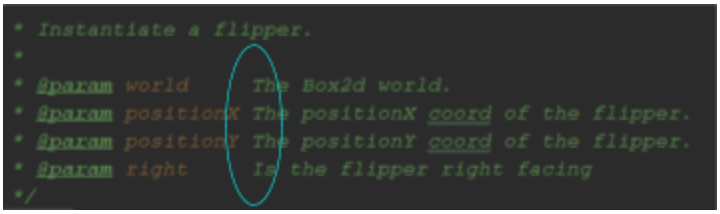
Remarks:

- Some static final fields are written in all caps, others are not, this is no good practise.
- Some fields have names equal to method names.
- We would prefer the Abstract prefix on the name of abstract classes.
- Methods which return a boolean should be called is... or has...

1.3.3. Comments

Remarks:

- Use @Override instead of writing purposeless javadoc above overridden methods.
- Private fields do not contain comments. In most cases this is OK, but in some cases this makes code hard to read.
- Some methods miss comments.
- Overall, methods that are commented are commented good and extensive.



1.4. Results

Overall code looks pretty sophisticated, this result in the following grade:

Topic	Weight	Grade
Source code quality	30%	7
Testing	20%	9
Code formatting	2%	7
Code naming	3%	7
Code comments	2%	6
Subtotal	57%	4,37
Total	100%	**7.67**

2. Enhancement proposal

The game as it is now, is very static. The scores that you get are always the same and the board always looks the same. This is not problematic, as the game just works, but it would add a lot to the dynamics of the game.

Our proposal is therefore to make a reasonable implementation of point streaks using the state pattern.

3. Additional notes

- Logging output is not very useful. A log does not indicate from where it was created.
- The game is not buildable instantly when loaded into Intellij. Fix this.
- Watch your code quality, the long methods can be implemented in a better way.