SCALE FOR PROJECT PISCINE PHP (/PROJECTS/PISCINE-PHP) / DAY 08 (/PROJECTS/42-PISCINE-C-FORMATION-PISCINE-PHP-DAY-08)

Guidelines

- Is everybody present?
- Git clone the repository.
- Begin the evaluation.

If the repository is empty stop the evaluation and put 0.
Discuss what went wrong (not necesseraly on the technical side) and how to avoid same situalion tomorrow.

The following points stop the evaluation and give a 0:

- The submitted project isn't writing in PHP server side (off topic)
- The project must work on the Chrome version installed on the Imacs.
- If the evaluation takes too long because of the installation or configuration of a dependency, it's up to you to decide if it's worth giving 0.
- A public method or attribut without explanation. If there is an explanation but you're not convinced it's up to you to decide if it's worth giving 0.
- If for an inheritance relationship ("extends") it's not possible to say that the child Class "is a" parent Class, it's up to you to decide if it's worth giving 0.
- Failing one of the following instructions is eliminatory:
  - Only one unique Class per file.
  - A file that has the definition of a class cannot have any other except for require or require_once if necessary.
  - A file containing a class must ALWAYS be named ClassName.class.php.
  - A class must ALWAYS be accompanied by a documentation file which MUST be named ClassName.doc.txt.
  - The documentation of a class must ALWAYS be useful and correspond to the implementation.
  - A class must ALWAYS have a static method called doc that returns the documentation in a string.

The following aren't eliminatory:
- Change some aspect of the game as long as they don't misrepresent the rules or the Warhammer 40000 universe.

Bare minimum

To be able to move to the next part and check the used notions, the project must at least implement the following:

- Display the game grid
- Display at least 2 spaceships (can be 2 times the same one)
- It's possible to move spaceships
- It's possible to shot the other spaceship

If those features are implemented, you can move on and check notions used.

d08's notions

For a notion to be accounted for, the relevance of the use in the contexte MUST convince the corrector. You're encouraged to debate. It a specific notion implementation doesn't convince the corrector, it's possible to show another implementation somewhere else in the project to try to convince the corrector again.

d08's notions

Each d08's notion is worth 1 point.

- Traits
- Multiple traits ("insteadof" and "as") - Traits and abstract methods
- Finalize
- Exceptions
- Reflection API

Rate it from 0 (failed) through 5 (excellent)

d06's notions

For a notion to be accounted for, the relevance of the use in the contexte MUST convince the corrector. You're encouraged to debate. It a specific notion implementation doesn't convince the corrector, it's possible to show another implementation somewhere else in the project to try to convince the corrector again. Lacking some d06's notions does not prevent from checking the d07's notions.

d06's notions 1/2

Each d06's notion is worth 1 point.

- Classes
- Attributes and methods (both)

- $this - Visibility
- Accessors- __get et __set (both)
- __toString

Rate it from 0 (failed) through 5 (excellent)

d06's notions 2/2

Each d06's notion is worth 1 point.

- Invoke - Comparisons
- Clone and __clone (both even if some will not require __clone)
- Class constants
- Static methods - Static attributes
- __call and __call_static (both)

Rate it from 0 (failed) through 5 (excellent)

d07's notions

For a notion to be accounted for, the relevance of the use in the contexte MUST convince the corrector. You're encouraged to debate. It a specific notion implementation doesn't convince the corrector, it's possible to show another implementation somewhere else in the project to try to convince the corrector again. Lacking some d06's notions does not prevent from checking the d08's notions.

d07's notions

Each d07's notion is worth 1 point.

- Inheritance
- Method ("override")
- Protected - Self - Static
- Abstract Classes - Abstract methods
- Interfaces

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok

Empty work | Incomplete work | No author file | Invalid compilation

Norme | Cheat | Crash | Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!