GC Coding Standards

(public_version r20121021)

Používanie nižšie uvedených štandardov budem kontrolovať a vyžadovať. Slúžia hlavne pre nás, aby nám v neskoršej časti vývoja ušetrili čas lúskaním svojho a hlavne cudzieho kódu.

V prípade, že máte výhrady, označte text a pridajte komentár (výhrady typu "je to hlúposť" ignorujem, nie je to hlúposť; pre úspešnú zmenu štandardu argumentujte). V prípade, že chcete niečo doplniť, vytvorte komentár na názve sekcie. Ja to buď schválim tým, že to pridám alebo okomentujem, že prečo si myslím že nie.

Dokument je možné komentovať na
https://docs.google.com/document/d/1PsJG90sRK9Df2ES0CzmuaxPfJ_Lhpgnsc3-kwQolxhI/edit

Global

- Žiadne "random" čísla a stringy a error cody v kóde; v prípade, že chcete niečo také použiť, zadefinujte public static final premennú v hlavičke triedy nad prvou metódou
- Jedna deklarácia premennej na riadok
- Každá premenná bude mať self-explanatory názov (žiadny String s). Máte IDE, ktoré to aj tak doplní za vás, omnoho lepšie sa to číta po niekom; jediná výnimka je pre iteračné premenné (i).
- Premenné sa deklarujú až vtedy, keď ich naozaj potrebujete.
- Telo metódy by sa malo vôjsť na jednu obrazovku. V prípade, že sa nevôjde ani na dve obrazovky, treba to rozbiť.
- Ternárne operátory používajte len v jedno-dvoj riadkových metódach - hlavne v metódach typu isSomething, kedy ide o vyhodnotenie a vrátenie true/false alebo krátkych metódach, ktoré niečo vracajú na základe vyhodnotenia jednej podmienky..

  ```
  return ($player->getAwesomeness() > Awesomeness::average) ? new AwesomeBadge() : new StandardBage();
  ```
- Instance variables sú výhradne private, zadefinujte gettre a prípadné settre. (okrem static final premennych, tie vacsinou robite preto, aby boli public)
- Metóda isNieco() musí vracať true/false. Metóda vracajúca true/false by mala mať v názve isNieco()
- Všetko, čo by nemali používať ostatní deklarujte ako protected/private (private naozaj len výnimočne, budeme chcieť aj dediť).
- Ak meníte public metódu, skontrolujte si kde všade sa používa aby ste niečo neznefunkčnili (toto by mali zabezpečiť testy v prípade, že ich napíšete správne a pokryjete nimi všetko).

- Komentujte len vtedy, keď je to naozaj nevyhnutné. Všetky komentáre sa dajú vyriešiť správnym nazvaním premenných a metód a refaktoringom. Navyše ten kód sa omnoho ľahšie debuguje a znovupoužíva.
- Metóda by mala robiť len to, čo má v názve. Ak mení iné objekty ako by sa očakávalo, refaktorujte.
- Na odsadenie používajte 4 znaky (teda tabulator s odsadenim 4 znakov, alebo medzery) - v IDE sa to dá nastaviť
- Statické konštanty píšte upper case a slova oddelujte medzerou: CLASS_CONSTANTS.
- V prípade, že v if bloku je len jeden riadok, obaľte ho zátvorkami - aj tak sa časom ukáže, že tam treba niečo pridať a tie zátvorky tam budete musieť dať; len čo budete zbytočne bohovať na autora pôvodného kódu

Štruktúra kódu

- Používajte exceptiony a setujte exception message (kľudne si narobte vlastných exceptien koľko potrebujete, omnoho lepšie sa debuguje zo stack tracu keď hneď viete kde je chyba). Nie je nič horšie ako “ifovať” niečo 5 úrovní do hĺbky. Exception flow to spraví za vás.
- Žiadny YODA jazyk v kóde

```
if (true == condition) // ZLE!
```

- Blokové zátvorky {} začínajú a končia samostatne na novom riadku (ide proti Java štandardu, ale naozaj zlepšuje čitateľnosť)

```
if (condition == true)
{
      // robte si co chcete...
}
else
{
      // a dorobte si zbytok...
}
```

- Používajte nad vlastnými metódami anotácie (PHPdoc, JavaDoc), pomáhajú IDE lepšie zobraziť autocomplete - hlavne v PHP, kvôli jeho netypovosti.

Testovanie

- Výstupné podmienky každej funkcionality budú zašpecifikované v Greenhopperi na zadnej strane kartičky; vaše testy budú pokrývať všetky výstupné podmienky a pri odovzdaní funkcionality budú všetky testy svietiť na zeleno
- Názvy testov robte podľa štandardov jednotlivých unit-test frameworkov (viem ze phpUnit ocakava nazvy testOdozvyServera - teda keyword test na zaciatku a pokracuje sa normalnym camel caseom)
- V prípade, že neviete test nazvať, máte problém pretože neviete čo idete robiť. Kontaktujte zodpovedného člena teamu o vysvetlenie a pomoc.

- Počet assertov na jeden test je neobmedzený
- Používajte pokročilé asserty, nie len základné 4 (typu len assertEquals)
- Ak to váš framework dovoľuje, zapnite si striktné testovanie - rovnako aj jazyk, v ktorom vyvíjate (ak PHP vyhodí notice, ošetrite ho)

Java / Javascript

Vychádzal som z Java Coding Standards - Oracle a zo skúseností zo všetkých projektov doterajších, zhrniem tu tie najpodstatnejšie. Píšem štandardne pre Javu, to čo sa dá aplikovať pre Javascript (a príp. jQuery), to tam aplikujte.

Premenné

- Názov premennej v camel case first lower formáte (String successResponseMessage)

Triedy

- Názov triedy v tvare prvé písmenko slova veľké: GpsLocation (v prípade skratiek je možné použiť aj GPSLocation).
- Ak nechcete, aby sa od triedy dedilo, dajte jej atribút final.
- Ak to je možné, vyhýbajte sa statickým metódam (pracujte nad inštantciou objektu)

PHP

Premenné

- Názvy premenných v camel case first lower ($successResponseMessage)

Metódy

- Volanie metódy, kde je ako parameter asociatívne pole rozbite kvôli čitateľnosti na pár riadkov (práve tam, kde je pole) - to isté platí aj pre napĺňanie samotného poľa

```
$this->strangeFunctionWithManyParams(10, 5, 'ahoj', array(
     'param1' => $val1,
     'param2' => 'val2',
     ....
));
```

HTML/CSS

- Používajte doctype HTML a v čo najväčšej možnej miere HTML5 a CSS3, nevyžadujem podporu starých prehliadačov (neverím, že človek čo má androida používa IE7 a otvára nám to kopu nových možností)
- V CSS používajte vendor prefixy (alebo zavedieme modernizr, nech to robí za nás)
- Máme kopu tagov (section, article, blockquote), nepoužívajte len divy a spany
- Názvy id a tried píšte lowercase a slová oddeľujte pomlčkou (nie podtržníkom)
  `<section id="user-settings" class="grid box-with-shadow"></section>`
- Akékoľvek štýlovanie patrí do CSSka, inline je povolený len dynamicky javascriptom a len ak to je nutné (keď sa obsah mení za behu)

VCS (version control)

- Na vývoj funkcionality si vytvorte vlastnú branchu
- Hotová funkcionalita sa bude zhromažďovať v devel branchi, kde sa bude testovať
- Otestovaná funkcionalita pôjde do stable branche, z ktorej sa spraví release
- Dojebaný commit alebo zabudnutý kus kódu v devel a stable branchi znamená pivo pre ostatných členov teamu
- Odporúčaný formát commitov je (v angličtine): GCTEAM-{ticket_number} [+|*|-] module: commit message
    - ticket_number: číslo príslušného ticketu v Jire
    - + pridaná funkcionalita
    - * opravená funkcionalita
    - - odstránená funkcionalita
    - module) špecifická časť projektu, ktorej sa to táka
    - commit message) jednou vetou (pripadne odrazkami) zhrnuté o čo v commite ide