

PG220

- ▶ <https://github.com/darkehis/PG220>

PG220

- ▶ <https://github.com/darkehis/PG220>
- ▶ 10 minutes par groupe.

PG220

- ▶ <https://github.com/darkehis/PG220>
- ▶ 10 minutes par groupe.
- ▶ Les slides sont dessus.

PG220: la forme

- ▶ Respecter le CamelCase.

PG220: la forme

- ▶ Respecter le CamelCase.
- ▶ Eviter d'avoir des ".class" sur le dépôt.

PG220: la forme

- ▶ Respecter le CamelCase.
- ▶ Eviter d'avoir des ".class" sur le dépôt.
- ▶ Que des caractères ASCII dans les sources (pas d'accent etc).

PG220: la forme

- ▶ Respecter le CamelCase.
- ▶ Eviter d'avoir des ".class" sur le dépôt.
- ▶ Que des caractères ASCII dans les sources (pas d'accent etc).
- ▶ Pas de méthode main = pas compilable.

PG220: la forme

- ▶ Respecter le CamelCase.
- ▶ Eviter d'avoir des ".class" sur le dépôt.
- ▶ Que des caractères ASCII dans les sources (pas d'accent etc).
- ▶ Pas de méthode main = pas compilable.
- ▶ Une classe Foo = un fichier Foo.java: sinon pas compilable.

PG220: la forme

- ▶ Respecter le CamelCase.
- ▶ Eviter d'avoir des ".class" sur le dépôt.
- ▶ Que des caractères ASCII dans les sources (pas d'accent etc).
- ▶ Pas de méthode main = pas compilable.
- ▶ Une classe Foo = un fichier Foo.java: sinon pas compilable.
- ▶ Un code pas compilable = un projet pas testable.

PG220: Staticité des variables

- ▶ Variable static = variable de classes.
- ▶ Variable final = variable immuable.

PG220: Staticité des variables

- ▶ Variable static = variable de classes.
- ▶ Variable final = variable immuable.
- ▶ POO: `Grid grid = new Grid();`

PG220: Staticité des variables

- ▶ Variable static = variable de classes.
- ▶ Variable final = variable immuable.
- ▶ POO: `Grid grid = new Grid();`
- ▶ `static int win(Grid grid);`
- ▶ `int win = Grid.win(grid) → NON!`

PG220: Staticité des variables

- ▶ Variable static = variable de classes.
- ▶ Variable final = variable immuable.
- ▶ POO: `Grid grid = new Grid();`
- ▶ `static int win(Grid grid);`
- ▶ `int win = Grid.win(grid) → NON!`
- ▶ `int win();`
- ▶ `int win = grid.win() → OUI :)`

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`
- ▶ Diminuer la restriction `uniquement si c'est nécessaire`

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`
- ▶ Diminuer la restriction `uniquement si c'est nécessaire`
- ▶ Pas d'attribut `public!!!`

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`
- ▶ Diminuer la restriction `uniquement si c'est nécessaire`
- ▶ Pas d'attribut public!!!
- ▶ Un attribut `private` avec un `getter` et un `setter` n'est plus vraiment `private`.

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`
- ▶ Diminuer la restriction `uniquement si c'est nécessaire`
- ▶ Pas d'attribut `public!!!`
- ▶ Un attribut `private` avec un `getter` et un `setter` n'est plus vraiment `private`.
- ▶ Ne pas effectuer les corrections/propositions de l'IDE aveuglément..

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`
- ▶ Diminuer la restriction `uniquement si c'est nécessaire`
- ▶ Pas d'attribut `public!!!`
- ▶ Un attribut `private` avec un `getter` et un `setter` n'est plus vraiment `private`.
- ▶ Ne pas effectuer les corrections/propositions de l'IDE aveuglément..
- ▶ `package intern` = pas de mot clef

PG220: Portée des variables

- ▶ Portée des variables/classes: commencer par `private final`
- ▶ Diminuer la restriction `uniquement si c'est nécessaire`
- ▶ Pas d'attribut `public!!!`
- ▶ Un attribut `private` avec un `getter` et un `setter` n'est plus vraiment `private`.
- ▶ Ne pas effectuer les corrections/propositions de l'IDE aveuglément..
- ▶ `package intern` = pas de mot clef
- ▶ mot clef `protected` : useless pour un tel projet

PG220: re la forme

- ▶ Eviter les méthodes trop grosses. (grand max 80 100 lignes)

PG220: re la forme

- ▶ Eviter les méthodes trop grosses. (grand max 80 100 lignes)
- ▶ Garder une langue unique pour le code.

PG220: re la forme

- ▶ Eviter les méthodes trop grosses. (grand max 80 100 lignes)
- ▶ Garder une langue unique pour le code.
- ▶ Avoir des noms de variables significatifs.

PG220: re la forme

- ▶ Eviter les méthodes trop grosses. (grand max 80 100 lignes)
- ▶ Garder une langue unique pour le code.
- ▶ Avoir des noms de variables significatifs.
- ▶ Eviter la duplication de code!