

# TP1 & TP2

Prise en main de Java et premières classes. Ceci est une introduction à la création de classe, comme pour la [création de la classe Card](#) du cours.

Lancer IntelliJ en ouvrant un terminal et en tapant :

```
idea.sh
```

## Hello World!

C'est une tradition de commencer tout nouveau langage par un "Hello World!". Suivez donc les instructions de [pour faire vos premiers pas avec IntelliJ](#) & créer votre premier projet Java.

Ceci fait, vous devriez avoir un environnement de développement opérationnel.

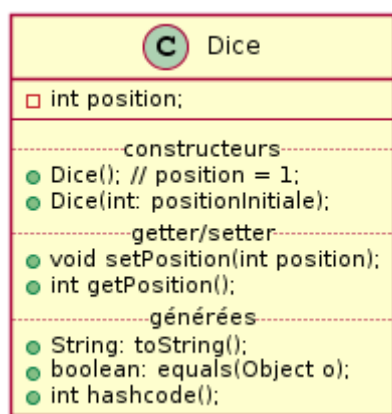
## Des Dés

On vous demande de créer une classe `Dice` qui représentera un dé à 6 faces.

### On débute

On commence par créer une première classe `Dice` avec les propriétés basiques que vous testerez dans votre programme principal.

### Modèle UML



### Tests

Créez une méthode `static` (le pourquoi du `static` est expliqué [ici](#)) dans la classe `Main` où vous testerez les différentes méthodes/attributs que vous créerez.

N'oubliez pas que les méthodes `toString`, les getter/setter et les méthodes `equals` et `hashCode` peuvent être générés automatiquement ("**code** » **generate...**").

### Dans un premier temps

- Créez un `Dé`,
- affichez son attribut à l'écran,
- Changez sa position et vérifiez que l'affichage a bien changé.

La doc utile pour la construction d'objets :

- [définir une classe](#)
  - [ses attributs](#),
  - [un constructeur](#),
- [créer et utiliser des objects](#)

### Dans un second temps

- Affichez un `Dé`
- Créez un second dé et initialisez le comme le premier dé
- Testez l'égalité des deux dés que vous venez de créer suivant:
  - l'opérateur `==` (réponse `false`),
  - la méthode `equals` (réponse `true` si les deux positions sont égales et `false` sinon).

La doc utile :

- [la visibilité](#) des méthodes et classes,
- [la surcharge](#) des méthodes d'`Object` permettant de :
  - afficher un objet à l'écran : méthode `toString`
  - égalité entre objet, méthode `equals`, qui ne va jamais sans la méthode `hashCode` (qui transforme un objet en entier).

## Le hasard

En utilisant le lien suivant : <http://www.leepoint.net/algorithms/random/random-api.html>, ajoutez une méthode `+void: roll()` à la classe `Dice` qui changera la position du dé (On suppose avoir affaire à un banal dé à six face).



Les documentations officielles du lien précédent sont :

- Classe [java.util.Random](#)
- Méthode [random\(\)](#) de la classe `java.lang.Math`

## Tests

Créez une méthode statique dans Main qui lance 100 fois le dé. Vérifiez que le nombre de fois où l'on obtient un 1 est cohérent avec la théorie.

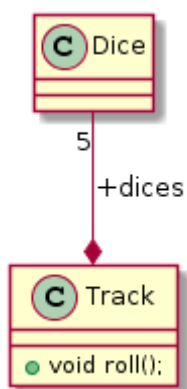
## Tapis vert

Créez une classe Track. Cette classe va nous permettre de jouer aux dés. Elle doit pouvoir lancer 5 dés.

Pour créer la classe Track, vous aurez besoin de gérer des [tableaux](#).

## UML

On propose le diagramme UML suivant :



## tests

Créez une méthode statique dans Main qui lance 100 fois les 5 dés (avec la méthode roll de Track).

Calculez la moyenne de la somme de ces 100 tirages. Est-ce cohérent avec la théorie ?

Attention, la division de 2 entiers donne un entier :

```
int un = 1;
int trois = 3;

int divisionEntiere = un / trois; // vaut 0
```



Pour obtenir un réel il faut utiliser la division réelle, et donc que l'un des deux protagoniste soit un réel :

```
float un = 1.0;
int trois = 3;
```

```
float division = un / trois; // vaut 0.3333333
```

On peut également transformer un entier en réel en utilisant le cast :



```
int un = 1;
int trois = 3;

// considere un comme un réel et on le divise par l'entier 3
float division = ((float) un) / trois;
```

From:

<https://wiki.centrale-med.fr/informatique/> - **WiKi informatique**

Permanent link:

<https://wiki.centrale-med.fr/informatique/restricted:mco-2:tp1>

Last update: **2016/03/01 10:41**

