REGLES DE DOCUMENTATION

A l'intention du groupe de T3

Règles générales

Il faudra documenter chaque classe et chaque fonction que vous avez conçue (les attributs ne seront pas documentés. Cela se fait dans un champ commentaires comme ceci :

```
/**
* Description de l'objet
*
* Liste des tags
*/
```

A savoir que toutes les descriptions supportent l'HTML. Ainsi, il est possible de mettre en gras, en italique, etc…

Listes des tags à inclure

Tags génériques

@see

Le tag *see* permet de faire référence à une autre classe/méthode.

Tags propres aux classes

@author

Le tag *author* permet de renseigner le ou les auteurs de la classe.

@version

Le tag *version* permet de renseigner le numéro de version de la classe.

Tags propres aux méthodes

@param

Le tag *param* permet de renseigner un paramètre d'entrée d'une méthode. Dans un premier lieu, on renseigne le nom du paramètre, ensuite, on inscrit une courte description.

@return

Le tag *return* permet de renseigner la variable de retour d'une méthode. De la même manière que la tag *param*, il faut renseigner le nom de la variable, suivi d'une courte description.

@throws

Le tags *throws* permet de renseigner les exceptions susceptibles d'être levées. (Ce qui exclu ceux qui sont capturées dans un try/catch). Il faut renseigner le nom de l'exception, suivi d'une explication de ce qui pourrait la déclencher.

@since

Le tag *since* permet d'indiquer depuis quelle version de la classe la méthode est implémentée. La version doit forcement être inférieure ou égale à la version de la classe.

@deprecated

Le tag *deprecated* permet d'indiquer qu'une fonction est obsolète (et donc plus utilisée). Il faut indiquer depuis quelle version la fonction est obsolète, et quelle est la méthode à utiliser à la place (utiliser le tag *see)*.

Exemple

```
/**
* <b> <i>Personne</i> est une classe que définit un individu. </b>
* <p>
* Un individu est caractérisé par les informations suivantes :
* <ul><li>Un nom et un prénom</li>
* <li>Une sexe</li>
* <li>Une date de naissance</li></ul>
* </p>
* <p>
*  Cette classe permettre de définir une personne. Elle sera crée par la classe Fabrique
<i>Naissance</i>
* </p>
*
* @see Naissance
*
* @author nderousseaux
* @version 2.3.2
*/
public class Personne {

    private String _nom;
    private String _prenom;
    private Sexe _sexe;
    private LocalDate _dateDeNaissance;
}

    /**
    * Constructeur de Personne
    * <p>
    * A la construction de l'objet, on prend le nom, le prénom, le sexe et la date de naissance
de la personne.
    * </p>
    *
    * @param nom Nom De l'individu
    * @param prenom Prenom de l'individu
    * @param sexe Sexe de l'individu
    * @param dateDeNaissance Date de naissance de l'individu
    *
    * @throws IllegalArgumentException Se déclenche si la date de naissance est supérieure à la
date actuelle.
    *
    * @see Personne#_nom
    * @see Personne#_prenom
    * @see Personne#_sexe
    * @see Personne#_dateDeNaissance
    */
    public Personne(String nom,String prenom,Sexe sexe, LocalDate dateDeNaissance){
        _nom = nom;
        _prenom = prenom;
        _sexe = sexe;
        _dateDeNaissance = dateDeNaissance;

        if(dateDeNaissance.isAfter(LocalDate.now())){
            throw new IllegalArgumentException();
        }
    }

    /**
    * Méthode qui retourne l'age du l'individu
    *
    * @return age Age de l'individu
    *
    * @see Personne#_dateDeNaissance
    *
    * @since 2.0
    */
    public int age(){
        return _dateDeNaissance.until(LocalDate.now()).getYears();
    }
}
```