*Esig-1*

**Hors série 1 - Bikes Repair**

# Open ...

# 

# 

# 

# 

# Objectifs:

1. Consolider votre compréhension de:
   1. la distinction entre classe utilisée et classe à utiliser,
   2. l’utilisation de la méthode toString( ),
   3. du fonctionnement du constructeur,
   4. l’utilité des getter et setter,
2. Créer et utiliser des tableaux d’objets,
3. Utiliser la classe Scanner pour lire un fichier texte externe et en traiter le contenu,
4. Travailler avec des dates (classe LocalDate),
5. Réviser les algorithmes Moyenne et Recherche et savoir les implémenter en Java.

# Mais aussi:

1. Produire du code à partir d’une situation problème,
2. Chercher des solutions en équipe et à l’aide de Google,
3. Envisager la suite du programme pour le cours ALO,
4. Prendre du plaisir à coder.

# Mise en situation

La société *Mybikes Sàrl* offre un service de réparation de vélos. En tant que développeur pour cette société, vous proposez de mettre en place les prémices d’un programme en Java permettant de traiter informatiquement les réparations afin d’en extraire (pour commencer) des statistiques.

### *Nota Bene:*

*Pour l’instant, le programme ne traite que 10 réparations, pas plus, pas moins. Ce point sera modifié plus loin.*

# Matériel à disposition:

Vous disposez de:

* la classe “Mybikes” qui contient la méthode main (c’est votre classe de test),
* la classe “Reparation” (vide pour le moment),
* un fichier texte “reparations.txt” contenant les informations des 10 réparations. Les informations dans ce fichier texte sont organisées comme suit:
* Chaque ligne correspond à une réparation,
* Une ligne contient dans l’ordre:  
   “sa date” “sa durée” “sa description” “son statut”.

# Caractéristiques (règles de gestion + informations)

1. Une réparation possède: une date, une durée (en minutes) une description et un statut.
2. Une réparation doit contenir au moins une description pour être créée. Par défaut le statut est “ouverte” à la création. La date, le statut et la durée peuvent être modifiés par la suite.
3. Le prix d’une réparation pourra se faire en fonction d’un tarif horaire, pour l’instant fixé à CHF. 120.- / heure (quelle que soit la réparation). Le prix d’une réparation étant calculable, il n’est pas nécessaire de le stocker dans la classe Reparation.

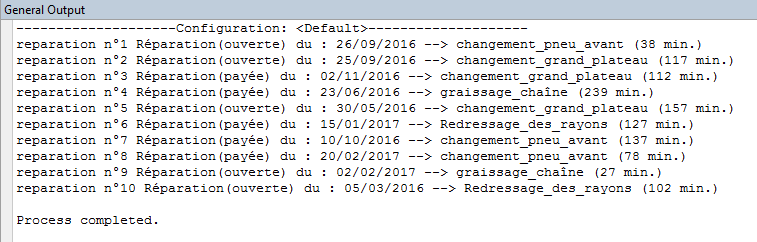
Les accès et modifications aux données d’une réparation doivent être potentiellement soumises à un contrôle, vous penserez donc à créer les “getter” et “setter” utiles.

# Fonctionnalités du programme

Le programme lit en entrée le fichier txt contenant les 10 réparations à l’aide d’une méthode lireDonnees( ) qui prend en paramètre le nom du fichier txt et qui retourne un tableau contenant les réparations. Ce tableau doit être ensuite exploité pour obtenir les affichages suivants:

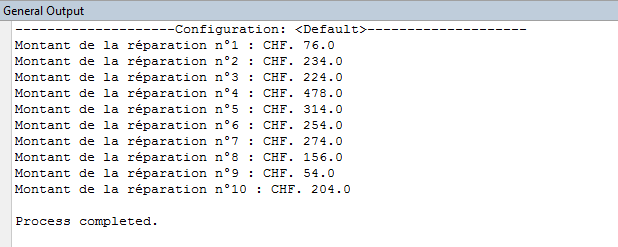
### Affichage des réparations

Une première méthode statique “afficherReparations( )” dans la classe Mybikes produira dans la console la sortie suivante:

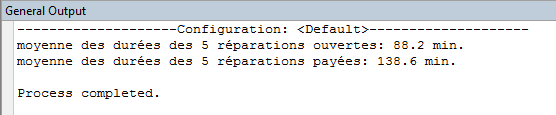


### Affichage des statistiques

1. Une méthode statique “afficherMontants( )” dans la classe Mybikes produira dans la console la sortie suivante:



1. Une méthode statique “afficherMoyennes( )” dans la classe Mybikes produira dans la console la sortie suivante:



# Tips pour le développement:

1. Pensez à effectuer un découpage en plusieurs méthodes statiques appelées depuis le main.
2. Pour convertir une date contenue dans une String en LocalDate, vous pouvez utiliser la méthode parse( ) de la classe LocalDate selon l’exemple suivant:  
   *LocalDate date = LocalDate.parse("2017-03-02").*
3. Pour tester l’égalité de deux String, il est nécessaire d’utiliser la méthode  
   .equals( ) de la classe String (plutôt que le “==” habituel).

# Pour continuer (et corser un peu l’affaire)

1. Faîtes en sorte que des durées négatives ne puissent pas être saisies par un utilisateur de la classe Reparation.
2. Modifiez la classe Reparation pour faire en sorte que toutes les durées soient arrondies au quart d’heure supérieur. En effet, le patron de l’entreprise vous a informé que la facturation ne s’effectuait pas à la minute près.
3. Créez une méthode statique afficherReparationApres( ) qui prendra en paramètre une date, affichera la liste des réparations postérieures à cette date.
4. Créez une méthode statique reparationExisteOnDate( ) qui prendra en paramètre une date et qui retournera vrai si une réparation existe à cette date (algorithme de recherche).
5. Ajoutez à vos statistiques le calcul de la durée minimale et maximale d’une réparation.
6. Essayez de modifier votre code pour gérer un nombre variable de réparations dans le fichier texte.
7. Créez une méthode statique trierReparations( ) qui prend en paramètre:
   1. le tableau de réparation
   2. une String qui peut avoir les valeurs “date” ou “duree”
   3. un booléen indiquant croissant (si vrai) ou décroissant (si faux).

Cette méthode triera donc le tableau par date ou par duree, selon les valeurs des paramètres ci-dessus et retournera le tableau trié. (affichez-le ensuite pour vérifier le fonctionnement).

1. Créez une interface de gestion des réparation en ligne de commande en utilisant un Scanner sur System.in. L’utilisateur entre des codes (à vous de les définir) qui exécutent les méthodes statiques que vous jugez intéressantes.  
     
   *Par exemple, l’affichage dans la console pourrait donner:*

