

R2.01 - Développement Orienté Objets (DOO)

EX 1 : Programmation Structurée en Java et Machine Virtuelle

Organisation des séances de TP

- Relisez votre poly de cours correspondant (ici le chapitre 1)
- Les exercices supposent que vous êtes allés en cours et que vous avez lu votre poly de cours
- Si besoin reportez-vous au poly du cours lorsque vous faites les exercices
- Vous n'avez pas à rendre vos sources à la fin de la séance
- Le corrigé des exercices vous sera envoyé par Jean-Claude Martin en temps voulu
- Vous pouvez télécharger ce fichier doc localement et y insérer vos réponses aux questions / schémas / code Java ...

EXA : Introduction à Eclipse

- Module info : "Dont creat"
- Parcourir la brochure Eclipse.pdf « Introduction à Eclipse » (ignorer la partie JUnit pour l'instant).
- Allumez un PC sous Windows
- Lancez Eclipse sur votre PC
- Spécifiez comme workspace votre dossier réseau afin de pouvoir récupérer vos projets d'une séance sur l'autre
- Sous Eclipse, créez un projet Java "EX-01" dans lequel vous stockerez toutes les classes développées dans cette feuille d'exercices.
- Cliquez sur le dossier src de ce projet, puis créez une classe AfficherSaisir
 - Ecrire une méthode main() qui affiche « Bonjour ».Exécutez la (vous devez toujours tester vos programmes au fur et à mesure !)
- Afficher ensuite le nombre pi (Math.PI) de manière formatée avec 3 chiffres après la virgule.
- Saisir un nombre n ; afficher un nombre aléatoire entre 0 et n
- En utilisant l'explorateur Windows, parcourez le contenu de votre dossier Workspace Eclipse. Trouvez où est stocké le fichier *AfficherSaisir.class* résultant de la compilation de votre classe AfficherSaisir

EX B : Tableaux de types de base, type boolean, type char, boucle foreach

- Créer une classe *Stock* ayant le squelette suivant que vous devrez compléter comme indiqué plus bas
ATTENTION AU FORMAT DES COMMENTAIRES : Pour chaque bloc d'instructions
 - Sauter une ligne
 - Commenter : *// Allouer ...*
 - Bloc d'instruction

```
public class Stock {
```

```

public static void main(String[] args) {
    //-----
    // Déclarer les constantes
    //-----
    ...

    //-----
    // Déclarer les tableaux
    //-----
    ...

    //-----
    // Allouer les tableaux avec une taille max définie en constante
    //-----
    ...

    //-----
    // Mettre des valeurs dans les cases
    //-----
    ...

    //-----
    // Parcours des deux tableaux avec deux boucles foreach
    //-----
    ...

    //-----
    // Parcours des deux tableaux en parallèle avec une boucle while et un
boolean
    //-----
    ...

    //-----
    // Parcours des deux tableaux en parallèle avec une boucle for "classique"
(pas une boucle for each)
    //-----
}
}

```

- Déclarer dans la méthode main () un tableau de caractères (char) pour y stocker des codes d'articles et un autre tableau de double pour les prix
- Allouer NB_ARTICLES cases dans ces tableaux avec NB_ARTICLES déclarée en constante égale à 3
- Y mettre les valeurs indiquées ci-dessous dans l'exécution
- Afficher le contenu des tableaux en utilisant deux boucles "for each" (cf cours 01)
- Afficher le contenu des tableaux en parallèle en déclarant un booléen et faire une boucle while
- Idem avec une boucle for "classique" (pas une boucle for each)

EXECUTION DU MAIN :

Parcours successif des tableaux avec deux boucle foreach :

A	B	C
10.5	2.5	21.5

Parcours des deux tableaux en parallèle avec une boucle while et un booleen :

A - 10.5 Euros

B - 2.5 Euros

C - 21.5 Euros

- Dessiner le schéma de la mémoire à la fin du main (**comme montré en cours**)

SCHEMA DE LA MEMOIRE **MODE BOITES ET FLECHES**

SCHEMA DE LA MEMOIRE **MODE RUBAN**

EX C : Méthodes et passage de paramètres

Quel est le type de passage de paramètre permis en Java ?

.....

Ecrire un programme et un sous-programme qui testent les points suivants (vous pouvez faire une classe avec **différentes méthodes static comme la classe Methodes du poly du cours 01** :

1. essayer d'échanger les valeurs de 2 variables de type char passées en paramètre (afficher leur valeur avant l'appel, pendant l'appel et après l'appel)
2. essayer de mettre 150.7f dans la case d'indice 0 d'un tableau de float passé en paramètre (idem afficher avant, pendant et après l'appel)
3. essayer de modifier l'adresse d'un tableau de nombres réels passé en paramètres (idem affichage) en stockant null dedans
4. Optionnel :
 - a. Ecrire une fonction `echangerDeuxFloat(float f1, float f2)` puis `echangerDeuxFloatDansTableau(float[] tabFloat)`
Comparer et analyser la différence.

OPTIONNEL : Tests Unitaires

Pour aller plus loin dans l'utilisation d'Eclipse et pour vérifier vous même votre travail, voici une activité optionnelle :

Le but est de vous montrer sur des exemples simples, le principe de "tests unitaires".

Il est en effet important de tester vos programmes et de vérifier qu'ils produisent bien les résultats escomptés.

Nous allons voir ici comment vérifier qu'un appel de méthode produit bien le résultat demandé.

Cela nécessite d'utiliser un framework avec des classes et des instructions que vous n'avez pas à comprendre pour l'instant.

Dans le menu «File» -> «Import», sélectionnez le menu Git «Projects from Git» (Next)

Sélectionnez «Clone URI» et remplissez la case «URI» avec

«<https://gitlab.dsi.universite-paris-saclay.fr/travaux-pratiques-doo-iut/exstudents01.git>» (Next, Next, Finish)

Toutes les autres informations seront renseignées automatiquement.

Vous aurez ainsi récupéré un projet EXSTUDENTS01 dans lequel chaque classe que vous avez définie jusqu'à maintenant est accompagnée d'un test.

Par exemple, ce projet comporte une classe `AfficherBonjour` et une classe `AfficherBonjourTest`. La classe `AfficherBonjourTest` sert à vérifier que la classe `AfficherBonjour` fait bien ce qui était demandé (afficher "Bonjour").

Dans la classe `AfficherBonjour` du projet EXSTUDENTS01

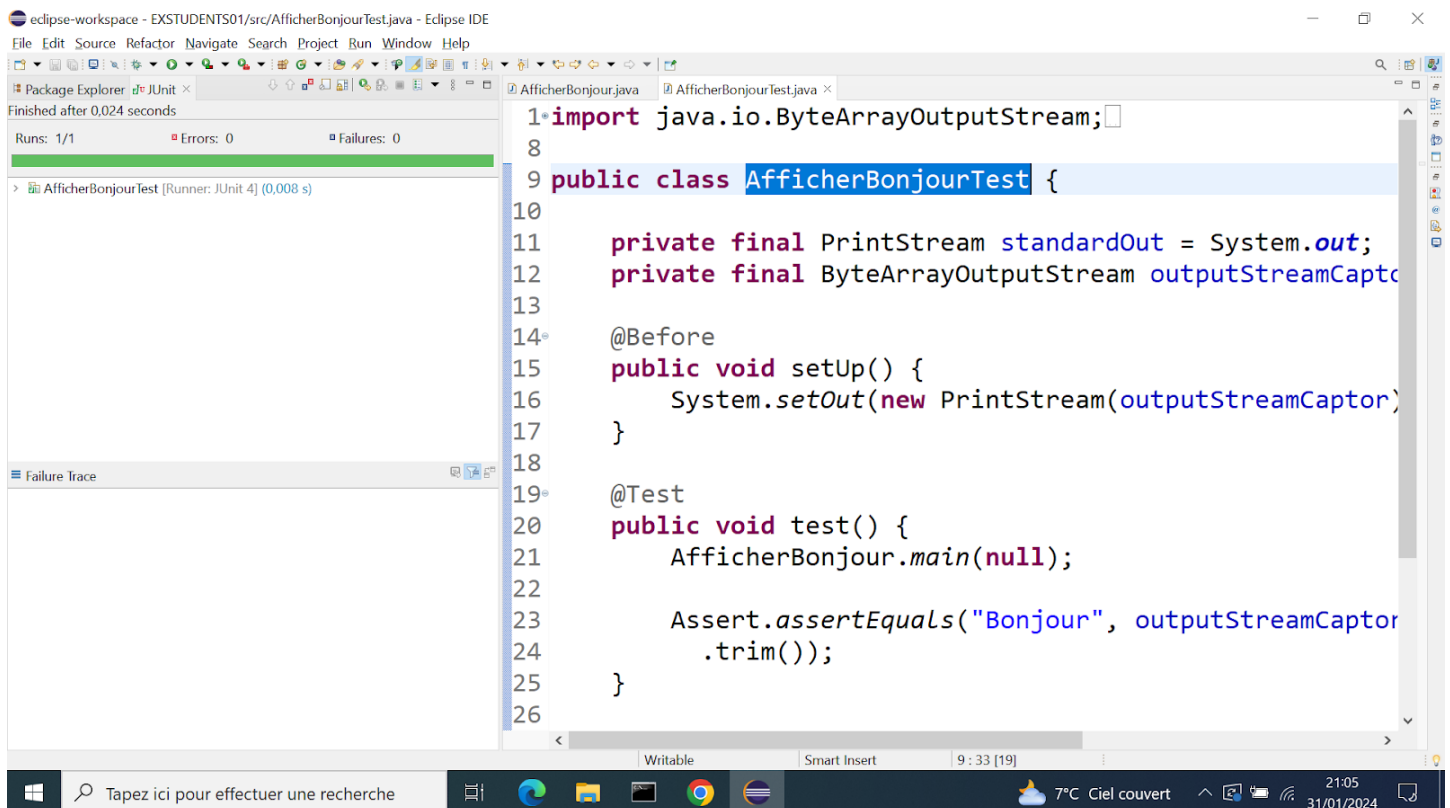
ajoutez dans le main () : `System.out.println ("Bonjour");`

Et supprimez l'instruction

`throw new UnsupportedOperationException("Not implemented yet");`

Exécuter le main de la classe `AfficherBonjourTest`.

Vous devriez voir s'afficher l'écran suivant qui indique que le test s'est bien passé (il y a un barre verte en haut à gauche et 0 erreurs) :



The screenshot shows the Eclipse IDE interface. The main editor displays the `AfficherBonjourTest.java` file with the following code:

```
1 *import java.io.ByteArrayOutputStream;
8
9 public class AfficherBonjourTest {
10
11     private final PrintStream standardOut = System.out;
12     private final ByteArrayOutputStream outputStreamCaptor
13
14     @Before
15     public void setUp() {
16         System.setOut(new PrintStream(outputStreamCaptor)
17     }
18
19     @Test
20     public void test() {
21         AfficherBonjour.main(null);
22
23         Assert.assertEquals("Bonjour", outputStreamCaptor
24             .trim());
25     }
26 }
```

The left sidebar shows the Package Explorer with the `JUnit` package selected. The bottom status bar indicates the test results: `Runs: 1/1`, `Errors: 0`, and `Failures: 0`. The bottom of the IDE shows the Windows taskbar with the date `31/01/2024` and time `21:05`.

Examinez la méthode `test` de la classe `AfficherBonjourTest`:

La 1ère instruction appelle la méthode `main()` de la classe `AfficherBonjour`.

La 2ème instruction compare ce qui a été affiché avec la chaîne de caractères "Bonjour".

Cette manière de tester une méthode par rapport à ce qu'on attend d'elle, s'appelle un test unitaire. Nous avons utilisé ici JUnit qui est un framework pour tester les méthodes en Java.

Faites de même avec les autres classes de cette feuille d'exercice.

Vous pouvez trouver plus d'informations sur les tests unitaires et JUnit, par exemple ici :

<https://www.jmdoudoux.fr/java/dej/chap-junit.htm>