**TP#1 ITII\_2018\_B**

**Le premier projet**

***Mise en place de l’espace de travail***

Tout d’abord nous créons notre espace de travail. Dedans nous créons notre Projet Java : ITII\_2018\_B en sélection Java 8. Nous vérifions que nous avons bien les vues Eclipse nécessaires (Package, Navigator, Problems, Console, listes des perspectives, Project explorer).

1. ***Création des packages***

Nous créons le paquetage com.itii (commande : New -> Package) toujours en minuscule

1. ***Création de la classe Launcher***

Nous créons ensuite le package **com.itii.planning** (ou sinon on peut rajouter le package **planning** au package **com.itii** existant). Il contiendra l’ensemble du code relatif à la partie graphique de notre application

Nous créons maintenant la classe **Launcher** (commande : New -> Class) toujours en majuscule

Nous importons donc ce package (En utilisant import)

Nous créons le point d’entrée du ***fenêtre*** programme : le main.

1. ***Création de la principale de notre application dans le main principal***

Nous créons la classe graphique **MainWindow** appartenant à ce nouveau package, mais cette fois-ci nous allons la faire hériter de **JFrame** afin de l’afficher sous forme d’une fenêtre, avec barre de titre.

Nous initialisons la fenêtre en lui donnant :

* Une taille : setSize(..)
* Une visibilité : setVisible(…)

Nous ajouterons à la fin l’appel aux méthode «**validate()**» et «**repaint()**» afin de mettre à jour la liste complète des composants de notre fenêtre et de les redessiner au lancement du programme.

L’affichage se fera en instanciant notre MainWindow depuis le main de la classe.

Launcher. Rappelons que MainWindow est un singleton et que son instanciation passe par l’appel a une méthode appelée communément « getInstance() ». L’illustration ci-contre montre comment instancier ce singleton depuis le main.

Nous importons alors le paquetage qui contient MainWindow pour que la classe Launcher en ait connaissance.

Astuce : Pour importer automatiquement la bibliothèque manquante on peut utiliser le raccourci **Ctrl + Shift + O**.

1. ***Lancement du programme***

Nous lançons le programme.

**Question 1**

Ctrl + alt + F est un raccourci qui sert à formater le code écrit sur Eclipse.

1. ***Arrêt du programme***

**Question 2**

Quand on ferme la fenêtre, le programme continue de tourner. On peut voir que l’icone d’état du programme est toujours rouge à ce moment.

**Question 3**

En rajoutant setDefaultCloseOperation( JFrame.*EXIT\_ON\_CLOSE );* on peut voir que le programme s’arrête correctement à la fermeture de la fenêtre.

***Coding guideline***

Ces règles de codage sont là pour certifier que l’ensemble des différents codes écrits par la promotion respectent un certain standard.

Ainsi, l’ensemble des membres d’une classes seront privés à l’exception éventuelle des membres statiques et autres constantes. L’utilisateur de ces membres se fera au travers d’accesseurs (getter / setter). Les *getter*s seront en charge d’assurer que le membre retourné n’est jamais null.

Le constructeur vide devra être explicitement écrit (qu’il soit privé ou public).

La classe MainWindow sera un singleton puisqu’elle représente notre fenêtre principale.

***L’application de gestion de Planning***

1. ***La barre de menu***

La barre de menu est une JMenuBar contenant des JMenu qui eux-mêmes peuvent contenir des JMenuItems s’ils possèdent des sous-menus.

Nous créons une classe PMenuBar, dans laquelle nous

Nous créons une classe MainPanel, dans laquelle nous

1. ***Le menu déroulant***
2. ***La Colone de noutons***