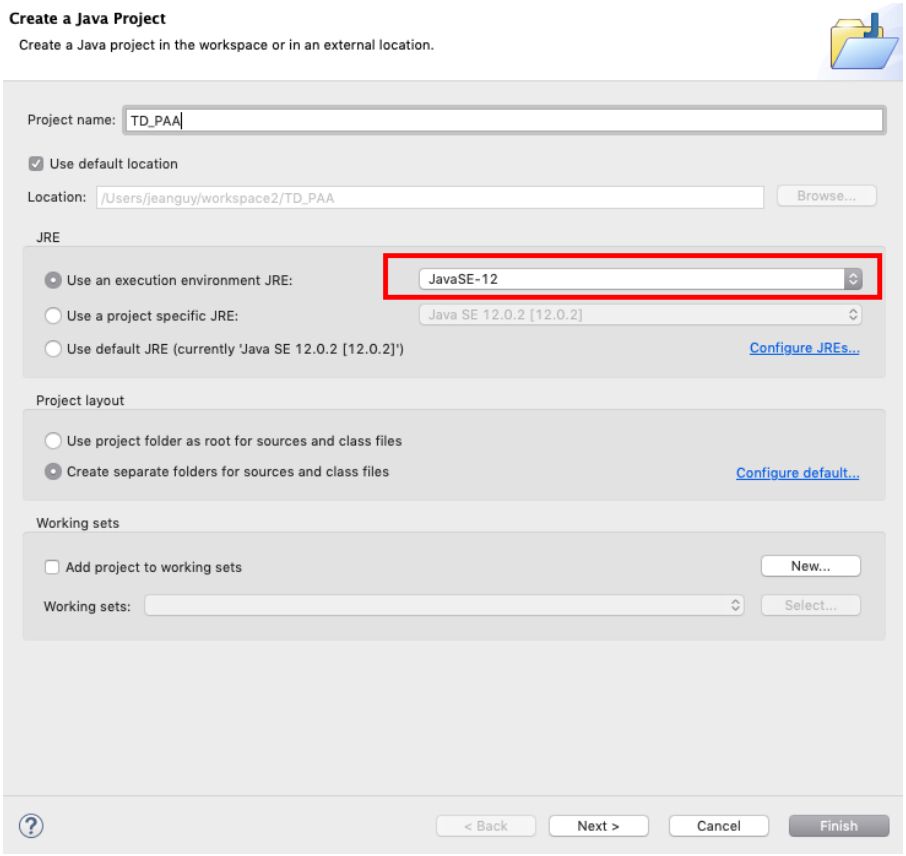
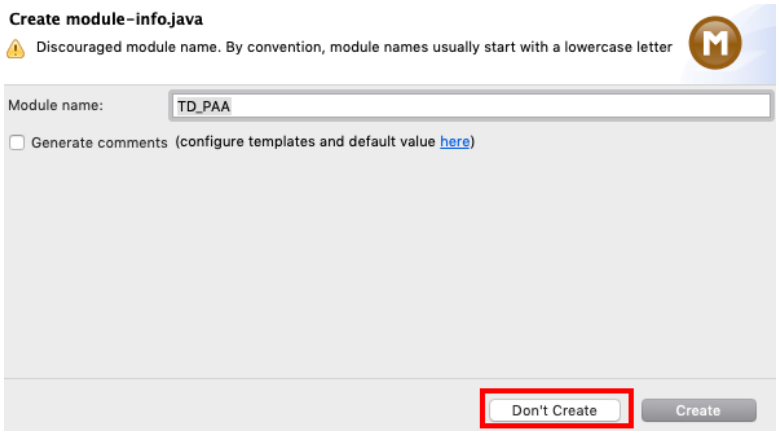
I Prise en main d'Eclipse

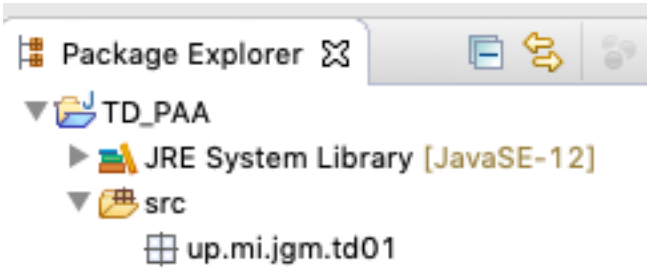
Après avoir lancé Eclipse, créez un projet appelé `TD_PAA`. Assurez vous de choisir Java 12 :



Lorsque cette fenêtre apparaît, sélectionnez `Don't Create`. Nous reparlerons plus tard de cette fonction.



Dans l'explorateur de packages, vous pouvez (et c'est très vivement conseillé) créer un package racine du projet (pour moi, up.mi.jgm), qui contient un package td01.



Quelques raccourcis clavier à retenir. Si vous utilisez MacOS, il y a de fortes chances que le raccourci soit le même, en remplaçant `Ctrl` par `cmd`.

- `Ctrl` + `Shift` + `F` : formatter automatiquement le code ;
- `Ctrl` + `Shift` + `O` : gérer les imports ;
- `Alt` + `Shift` + `R` : renommage intelligent, attention, sous MacOS : `cmd` + `Alt` + `R` ;
- `Ctrl` + `Espace` : auto-complétion.

II Premières méthodes

Rappel : On peut définir des« fonctions classiques » grâce au mot-clé **static**, c'est-à-dire des méthodes qui ne dépendent pas d'une instance de classe.
Pensez à écrire une méthode main pour vérifier le bon fonctionnement de votre code.

Exercice I (UtilMath)
Définir la classe UtilMath possédant les méthodes suivantes :

- **public static int** somme3(**int** a, **int** b, **int** c) qui fait la somme de trois entiers ;
- **public static long** fact(**int** n) qui calcule la factorielle $n!$ d'un entier positif n ;
- **public static long** comb(**int** n, **int** p) qui calcule la combinaison $\binom{n}{p}$ de deux entiers positifs p et n ;[1]
- **public static long** puissance(**int** n, **int** m) qui calcule la puissance m-ème de n : n^m, avec $m \geqslant 0$.

Exercice II (Maximum)
1. Définir une méthode **public static int** max2(**int** a, **int** b) qui retourne le maximum des deux entiers donnés en paramètre.
2. Définir une méthode qui retourne le maximum de trois entiers donnés en paramètre, avec deux version différentes :
   (a) **public static int** max3v1(**int** a, **int** b, **int** c) qui n'utilise pas max2 ;
   (b) **public static int** max3v2(**int** a, **int** b, **int** c) qui utilise max2.

Exercice III (Manipulation de tableaux)
Commencez par reprendre la classe UtilTab vue en cours.

1. Définissez la méthode max.

1. Rappel : $\binom{n}{p} = \frac{n!}{p! \times (n-p)!}$, avec $p \leqslant n$.

2. Définir une méthode qui calcule la moyenne des éléments d'un tableau de **double**.

3. Définir une méthode qui calcule la médiane des éléments d'un tableau de **double**. [2]

4. Définir une méthode qui, étant donnés un tableau de **double** et un tableau d'**int** de même longueur, calcule la moyenne des éléments du premier tableau pondérée par les éléments du second tableau.

Exercice IV (Modalités de contrôle de connaissance)
Dans de nombreuses unités d'enseignement, la note finale d'un étudiant est la moyenne d'une note de contrôle continu et d'une note d'examen, sauf si la note d'examen est supérieure à celle du contrôle continu. Dans ce cas, seule la note d'examen compte.

1. Définir une méthode qui calcule la note finale d'un étudiant à partir de deux **double** correspondant à sa note de contrôle continu et sa note d'examen.

2. On peut représenter un groupe d'étudiants par un tableau de **double**, tel que si i est un nombre pair, le i-ème élément du tableau est la note de contrôle continu d'un élément, et le $(i+1)$-ème est sa note d'examen. Définir une méthode qui calcule la moyenne d'un groupe d'étudiants. [3]

Exercice V (String et formatage de nombres)
Définir une méthode qui prend en entrée un nombre entier < 10000 et qui le formate sous forme de String sur cinq caractères, avec des espaces à gauche. Par exemple,

- formatage(5) retourne "␣␣␣␣5";
- formatage(25) retourne "␣␣␣25";
- formatage(325) retourne "␣␣325".

⚠ Cette méthode retourne un String, mais elle n'affiche rien !

Exercice VI (Interface textuelle pour UtilMath)
1. Écrire un programme qui demande à l'utilisateur s'il souhaite calculer la somme de trois entiers, la factorielle d'un entier, la combinaison de deux entiers, ou la puissance m-ème d'un entier. En fonction de sa réponse, le programme demande ensuite à l'utilisateur le (ou les) nombre(s) nécessaire(s) pour le calcul, et affiche le résultat.

2. Écrire une variante du programme dans laquelle le programme revient au menu après avoir affiché le résultat du calcul. Dans ce cas, il faut prévoir une option "Quitter" en plus du choix des opérations.

Exercice VII (Interface textuelle pour les MCC)
Écrire un programme qui demande à l'utilisateur un nombre d'étudiants dont on doit entrer les notes, puis pour chaque étudiant sa note de contrôle continu et sa note d'examen. Le programme doit ensuite afficher à l'écran la moyenne du groupe d'étudiants.

Exercice VIII (Nombres premiers)
Un nombre entier est premier s'il a exactement deux diviseurs : 1 et lui-même. [4] Pour savoir si un nombre n est premier, il suffit d'essayer de le diviser par tous les entiers $2 \leqslant k < n$, et si un d'entre

2. Rappel : la médiane d'une série de nombres est un nombre x tel que la moitié des éléments de la série sont inférieurs à x, et la moitié sont supérieurs à x.

3. On considère pour l'instant que le tableau contient un nombre pair > 0 d'éléments. On ne gère donc pas les erreurs si ce n'est pas le cas.

4. De fait 1 n'est pas premier car il n'a qu'un seul diviseur.

eux est un diviseur de n, alors n n'est pas premier. Dans le cas contraire, n est premier. On rappelle que n'importe quel nombre entier (> 1) peut être décomposé de manière unique en un produit de puissances de nombres premiers. Par exemple, $360 = 8 \times 9 \times 5 = 2^3 \times 3^2 \times 5$.

1. Écrire une méthode qui détermine si un entier n est premier. Plusieurs versions sont possibles.
   (a) Définir d'abord une méthode qui vérifie tous les entiers $2 \leqslant k < n$.
   (b) On peut améliorer l'algorithme en vérifiant uniquement pour $k = 2$ et pour les entiers impairs compris entre 2 et n.[5] Définissez cette variante de la méthode.
   (c) On peut encore améliorer l'algorithme en arrêtant les tests si $k > \sqrt{n}$.[6] Écrivez cette nouvelle version de la méthode.
2. Définir une méthode qui prend en entrée un nombre entier n, et qui retourne un String correspondant aux n premiers nombres premiers, séparés par un espace.
3. On souhaite déterminer la décomposition en facteurs premiers d'un nombre entier n.
   (a) Écrire une méthode qui prend en entrée un entier n et un nombre premier[7] p et qui retourne la puissance à laquelle p apparaît dans la décomposition de n.
   Par exemple, si on appelle cette méthode avec $n = 360$ et $p = 2$, on obtient le résultat 3. Avec $p = 7$, on obtient 0.
   (b) Écrire une méthode qui retourne un String correspondant à la décomposition en facteurs premiers d'un entier n.

Exercice IX (Jeu du nombre secret)
On souhaite implémenter un programme qui joue au jeu du nombre secret avec l'utilisateur :

- Le joueur (humain) pense à un nombre entre 1 et 100 ;
- L'ordinateur fait une proposition au joueur ;
- Le joueur indique au programme si le nombre proposé est plus grand, plus petit, ou égal au nombre secret ;
- L'ordinateur continue à chercher jusqu'à ce qu'il trouve.

Écrivez un programme qui permet de jouer à ce jeu. Voici un exemple d'exécution du programme :

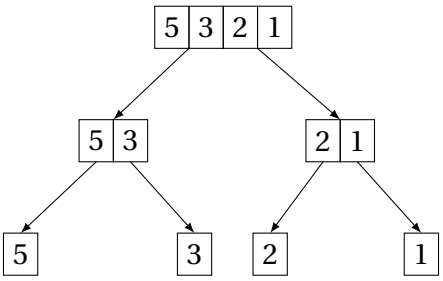
5. En effet, si n est divisible par un entier pair, il est forcément divisible par 2…
6. Si $n = k \times k'$ avec $k > \sqrt{n}$, alors nécessairement on a déjà trouvé k' plus tôt dans les itérations de la boucle…
7. On ne gère pas les erreurs si la méthode est appelée avec un nombre non premier.

```
Problems  Javadoc  Declaration  Console
<terminated> NombreSecret [Java Application] /Library/Java/JavaVirtualMachines/jdk-12.0.2.
Pensez a un nombre !
Ma proposition est 50.
Est-ce plus grand (+), plus petit (-), ou egal (=) a votre nombre ?
+
Ma proposition est 25.
Est-ce plus grand (+), plus petit (-), ou egal (=) a votre nombre ?
+
Ma proposition est 13.
Est-ce plus grand (+), plus petit (-), ou egal (=) a votre nombre ?
-
Ma proposition est 19.
Est-ce plus grand (+), plus petit (-), ou egal (=) a votre nombre ?
+
Ma proposition est 16.
Est-ce plus grand (+), plus petit (-), ou egal (=) a votre nombre ?
-
Ma proposition est 17.
Est-ce plus grand (+), plus petit (-), ou egal (=) a votre nombre ?
=
Victoire !
```

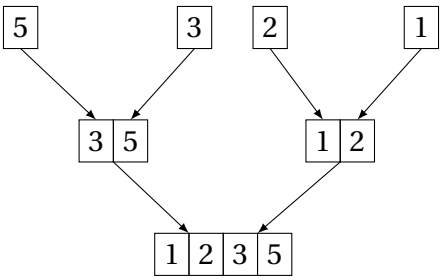
Exercice X (Tri fusion)
On peut trier efficacement des données grâce au principe du tri fusion. Cet algorithme est plus efficace que des algorithmes simples comme celui vu en cours, car le nombre d'opérations est une fonction de $n \times \log(n)$ (pour n le nombre d'éléments à trier) au lieu de n^2 pour les algorithmes simples.
Le principe de l'algorithme est assez simple :

- Un tableau qui contient un seul élément est déjà trié.
- Si un tableau contient plus qu'un élément, on peut le diviser en deux moitiés, trier séparément chaque moitié, puis fusionner les deux moitiés.
- La fusion consiste à inter-classer les éléments de chaque moitié en un seul tableau, ce qui est simple car les deux moitiés sont déjà triées à ce moment là.

Par exemple, on divise récursivement le tableau [5,3,2,1] :



Le processus de fusion consiste à comparer le premier élément de la première moitié, le premier élément de la deuxième moitié, et insérer le plus petit des deux en premier dans le résultat. Puis on réitère avec chaque élément des deux moitiés du tableau.



1. Implémenter la méthode suivante :

```
/**
 * Si les elements du tableau t son tries entre iMin et iMilieu - 1
 * d'une part, et entre iMilieu et iMax d'autre part, la methode
 * modifie t pour qu'il soit trie entre iMin et iMax.
 */
public static void fusion(int[] t, int iMin, int iMilieu, int iMax){
  ...
}
```

   Conseil : On peut créer un tableau intermédiaire dans lequel les valeurs sont rangées au fur et à mesure de la fusion, avant de les recopier dans le tableau principal à la fin.

2. Implémenter la méthode suivante :

```
/**
 * Trie par fusion les elements de t entre les indices iMin et iMax
 */
public static void triFusion(int[] t, int iMin, int iMax){
  ...
}
```