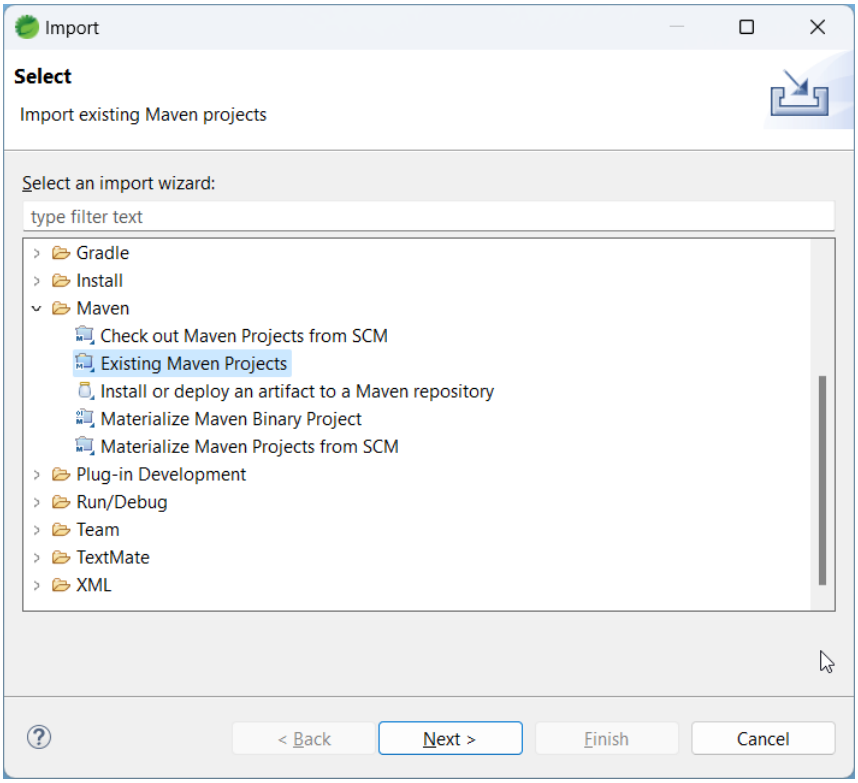
Exercices en auto-formation

TABLEAUX, BOUCLES ET CONDITIONS

MISE EN PLACE

- **Forkez** le projet https://github.com/DiginamicFormation/approche-imperative-exos
- **Clonez** ce projet en local
- **Importez** le dans votre IDE Eclipse :
    - Allez dans l'option de menu **File** puis **Import**
    - Choisissez Maven puis Existing Maven Projects



    - Cliquez sur **Next**
    - Cliquez ensuite sur **Browse** et sélectionnez le répertoire contenant le projet cloné.
    - Cliquez sur **Finish**

EXPLICATIONS

- Vous allez trouver les packages et les classes dans le répertoire **src/test/java**
- Les exercices se trouvent dans le package **fr.algorithmie**
- Pour exécuter un exercice, faites un clic droit, sélectionnez **Run as** puis **JUnit Test**

- Pour réaliser les exercices vous trouverez les consignes dans le PDF ainsi que dans le code lui-même
- **Pour LOGUER les résultats, utilisez Resultat.log car la vérification de vos résultats sera basée sur ce que vous avez logué avec cette méthode**

Exercice Ex01_AffichageIdentite (facultatif si le TP 4 a ete fait)

- Utiliser une boucle **for** pour **LOGUER** 10 fois un texte quelconque

Exercice Ex02_AffichagePartiel

- Soit le tableau suivant déclaré dans la classe:
    - **int[] array = {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;**
- **Combiner une boucle et un test** de manière à ne LOGUER que les valeurs du tableau supérieures ou égales à 3
- Combiner une boucle et un test de manière à ne LOGUER que les valeurs du tableau paires (0 est considéré comme pair)
- Combiner une boucle et un test de manière à ne LOGUER que les valeurs correspondant aux index pairs (l'index 0 est considéré comme pair)
- Combiner une boucle et un test de manière à ne LOGUER que les valeurs impaires strictement positives du tableau

Exercice Ex03_AffichageTableau

- Soit le tableau suivant : int[] array = {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;
- LOGUER l'ensemble des éléments du tableau grâce à une boucle
- LOGUER l'ensemble des éléments dans l'ordre inverse du tableau

Exercice Ex05_InversionContenu

- Soit le tableau suivant : int[] array = {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;
- Créer un tableau arrayCopy et copier tous les éléments de array dans arrayCopy **mais dans l'ordre inverse**.
- LOGUER l'ensemble des éléments du tableau arrayCopy

Exercice Ex06_RechercheMax

- Soit le tableau suivant : int[] array = {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;
- Rechercher le plus grand élément du tableau et **LOGUEZ** le

Exercice Ex07_RechercheMin

- Soit le tableau suivant : int[] array = {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;
- Rechercher le plus petit élément du tableau et **LOGUEZ** le

Exercice Ex08_CalculMoyenne

- Soit le tableau suivant : int[] array = {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;
- Calculer et **LOGUER** la moyenne des valeurs du tableau (la moyenne doit être un nombre décimal)
- Calculer et **LOGUER** la moyenne des **valeurs positives** du tableau uniquement (la moyenne doit être un nombre décimal)

Exercice Ex09_SommeDeTableaux

- tab1 : {1, 15, -3, 0, 8, 7, 4, -2, 28, 7, -1, 17, 2, 3, 0, 14, -4} ;
- tab2 : {-1, 12, 17, 14, 5, -9, 0, 18, -6, 0, 4, -13, 5, 7, -2, 8, -1} ;
- Créer un tableau **somme** de même taille que les tableaux précédents et dont chaque case d'index i contient la somme des cases d'index i des tableaux 1 et 2.
    - Exemple : somme[0]=tab1[0]+tab2[0]
- **LOGUEZ** les valeurs du tableau résultant avec une boucle.

Exercice Ex10_ComparaisonTableau

- Soit les tableaux suivants :
    - int[] array1 = {1, 15, -3, 8, 7, 4, -2, 28, -1, 17, 2, 3, 0, 14, -4} ;
    - int[] array2 = {3, -8, 17, 5, -1, 4, 0, 6, 2, 11, -5, -4, 8} ;
- **LOGUER** le nombre de valeurs communes aux 2 tableaux. On peut déjà voir que les valeurs 3 et 8 sont communes aux 2 tableaux, mais combien y en a-t-il au total ?

Exercice Ex11_FirstLast6

- Dans cette classe, plusieurs tableaux d'entiers permettront de contrôler votre algo.
- On calcule une valeur booléenne qui contrôle le tableau de la sorte :
    - elle vaut true si le tableau a au moins 1 élément et si le premier élément ou le dernier élément vaut 6.
    - elle vaut false dans les autres cas
- écrire l'algo de valorisation de cette variable avec le minimum de ligne
- **LOGUEZ** le résultat de votre algorithme pour chacun des 6 tableaux suivants :

```
int[] tab1 = {};
int[] tab2 = {2};
int[] tab3 = {6};
int[] tab4 = {1, 6};
int[] tab5 = {6, 1};
int[] tab6 = {0, 6, 1, 2};
```

Exercice Ex12_FirstLast

- Dans cette classe, on déclare un tableau d'entiers
- On calcule une valeur booléenne qui contrôle le tableau de la sorte :
    - elle vaut true si le tableau est de longueur supérieure ou égale à 1 et que le premier et le dernier élément du tableau ont la même valeur
    - elle vaut false dans les autres cas
- **LOGUEZ** le résultat de votre algorithme pour chacun des 4 tableaux suivants :

```
int[] tab1 = {};
int[] tab2 = { 2 };
int[] tab3 = { 1, 6 };
int[] tab4 = { 1, 6, 1 };
```

Exercice Ex13_Rotation

- Dans cette classe, on déclare un tableau d'entiers
- Effectuez une rotation à droite des éléments.
- Exemple : si initialement vous avez {0,1,2,3} dans le tableau alors après rotation vous obtenez {3,0,1,2}
- **LOGUEZ** tous les éléments du tableau après rotation avec une boucle for

Exercice Ex14_InteractifChiffresSuivants

Ecrire un programme qui demande un nombre à l'utilisateur puis **LOGUEZ** les 10 nombres suivants.

- Par exemple si l'utilisateur saisit 5, le programme LOGUE : 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.

<u>Instruction pour poser une question à l'utilisateur :</u>

Nous allons utiliser la classe java.util.Scanner.

```
Scanner scanner = new Scanner(System.in) ;
int nb = scanner.nextInt() ;
```

Tant que l'utilisateur ne saisit pas de valeur, cette méthode reste en attente !

Exercice Ex15_InteractifSommeArithmetique

Ecrire un programme qui demande un nombre à l'utilisateur puis calcule la somme de tous les entiers compris entre 1 et ce nombre inclus.

Exemple si l'utilisateur saisit 5, le programme affiche: 15

1) LOGUEZ ce nombre
2) LOGUEZ la somme de 1 à ce nombre inclus

Exercice Ex16_InteractifTantQue

Ecrire un programme qui demande un nombre à l'utilisateur qui doit être obligatoirement compris entre 1 et 10 :

- Tant que ce nombre n'est pas compris entre 1 et 10, le programme redemande un nombre à l'utilisateur.
- Si le nombre est compris entre 1 et 10, le programme LOGUE ce nombre et se termine.

Exercice Ex17_InteractifTableMult

Ecrire un programme qui demande un nombre à l'utilisateur un nombre **qui doit** être compris entre 1 et 10.

- Tant que l'utilisateur ne saisit de nombre compris entre 1 et 10 le programme redemande un nombre
- Si le nombre est bien entre 1 et 10, le programme **LOGUE** la table de multiplication de ce nombre puis s'arrête.
- Vous devez respecter le template de formatage suivant :

3 * 1 = 3

3 * 2 = 6

…

3 * 10 = 30

Exercice Ex18_InteractifPlusGrand

Ecrire un programme qui demande 10 nombres à un utilisateur, LOGUE chacun de ces nombres puis LOGUE le plus grand de ces nombres.

Indication : vous pouvez utiliser un tableau de longueur 10 pour stocker les 10 nombres.

Exercice FabriquerMur (Difficile - Facultatif)

- Copiez la classe **FabriquerMur** dans votre projet STS
- Dans cette classe vous devez mettre au point la méthode **fabriquerMur**
- Cette méthode doit produire un algorithme qui retourne s'il est possible ou non de fabriquer un mur avec des briques de longueur 1 et des briques de longueur 5.
- Exemples :
    - j'ai 2 briques de longueur 1 et 2 briques de longueur 5, est-il possible de créer un mur de 11m ? la réponse est oui, il suffit de prendre 2 briques de 5 et une brique de 1.
    - j'ai 3 briques de longueur 1 et 1 brique de longueur 5, est-il possible de créer un mur de 9m ? la réponse est non.
- Veuillez compléter la méthode fabriquerMur qui prend en paramètres :
    - nbSmall : le nombre de briques de longueur 1
    - nbBig : le nombre de briques de longueur 5
    - longueur : la taille du mur.
- A l'exécution les méthodes **verifier** exécutées avec diverses valeurs de paramètres permettent de dire si oui ou non votre algorithme fonctionne.

```
public class FabriquerMur {
	public static void main(String[] args) {
		// Tests de vérification
		verifier(3, 1, 8, true);
		verifier(3, 1, 9, false);
		verifier(3, 2, 10, true);
		verifier(3, 2, 8, true);
		verifier(3, 2, 9, false);
		verifier(6, 1, 11, true);
		verifier(6, 0, 11, false);
		verifier(1, 4, 11, true);
		verifier(0, 3, 10, true);
		verifier(1, 4, 12, false);
		verifier(3, 1, 7, true);
		verifier(1, 1, 7, false);
	}

	static boolean fabriquerMur(int nbSmall, int nbBig, int longueur) {
		boolean resultat = false;

		return resultat;
```

```
	}
	private static void verifier(int nbSmall, int nbBig, int longueur, boolean b)
{
		if (!fabriquerMur(nbSmall, nbBig, longueur) == b) {
			System.err.println("Test (" + nbSmall + ", " + nbBig + ", " +
longueur + ") NON passant.");
		}
	}
}
```

Exercice InteractifStockageNombre (Difficile)

Créer une classe **InteractifStockageNombre**

Faire un programme avec le menu suivant :

1. Ajouter un nombre

2. Afficher les nombres existants.

Description :

Demander à l'utilisateur de choisir une option dans le menu.

Si l'utilisateur sélectionne l'option 1, le programme demande un nombre à l'utilisateur puis l'ajoute à un tableau.

Si l'utilisateur sélectionne l'option 2, le programme affiche le contenu du tableau.

Si le tableau est plein, écrire un algorithme pour agrandir le tableau.

Exercice InteractifFibonnaci (difficile)

Créer une classe **InteractifFibonacci**

La suite de Fibonacci est une suite qui commence par 0 et 1 et dans laquelle le **nombre** de rang **N** est égal à la somme des nombres de rangs N-1 et N-2

- Créer une classe TestFibonacci
- Demander à l'utilisateur de choisir un rang N
- Ecrire un algorithme qui calcule et affiche le nombre de rang N

Exercice Interactif21Batons (difficile)

Créer une classe **Interfactif21Batons**

Le jeu est simple mais la réalisation est plus délicate. Vous allez jouer contre l'ordinateur. Celui qui prend le dernier baton a perdu.

Dans ce TP vous allez devoir imaginer vous-même le mécanisme à mettre en place, sur la base de ce que vous avez vu précédemment.