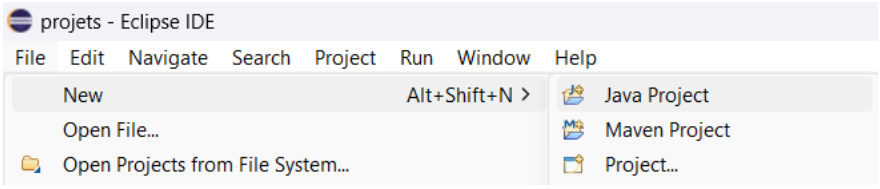
# TP N°1

Types primitifs, tests, boucles, chaînes de caractères, tableaux

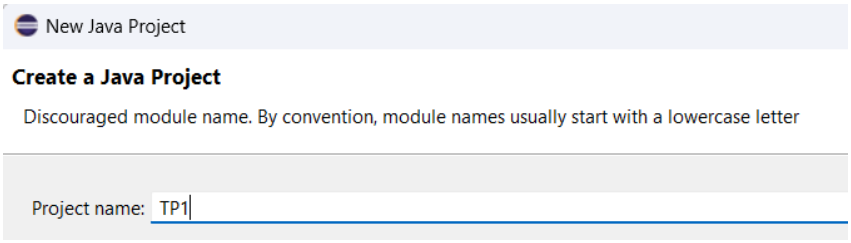
## Exercice 1 :

### ➔ Partie A :

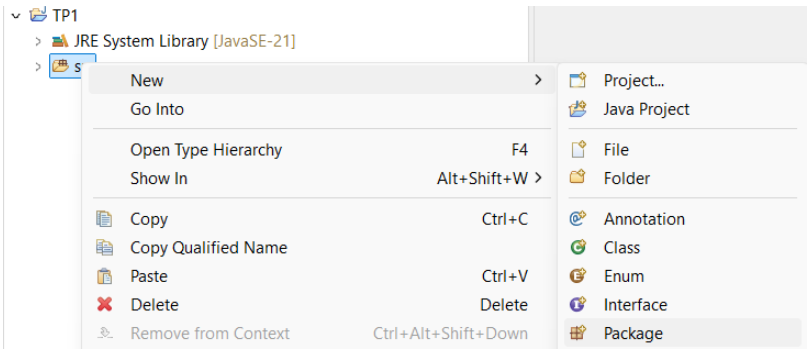
- Ouvrez **Eclipse IDE** et créez un projet Java : **File ->New->Java Project**



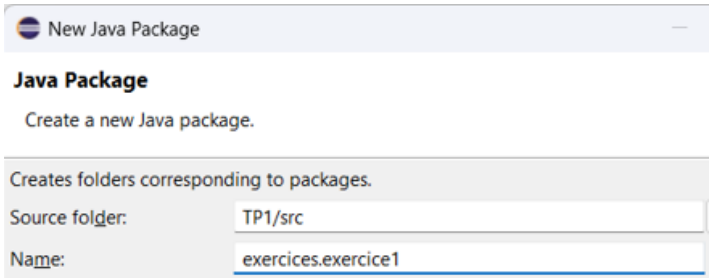

- Nommez votre projet **TP1**



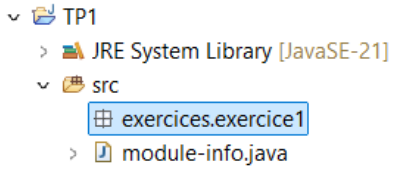

- Cliquez avec le bouton droit sur le dossier **src** de votre projet et choisissez : **New -> Package**




- Nommez votre package **exercices.exercice1**

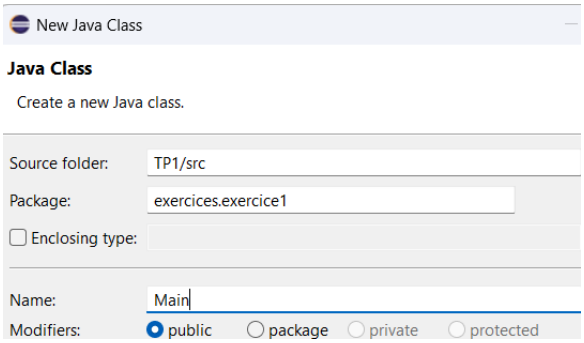


Vous obtenez l'arborescence suivante :

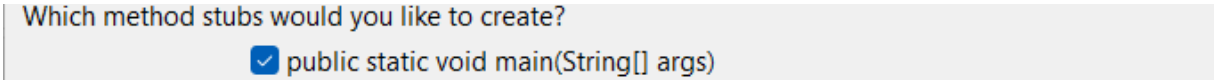



- Cliquez avec le bouton droit sur le package **exercices.exercice1** et choisissez : **New -> class**

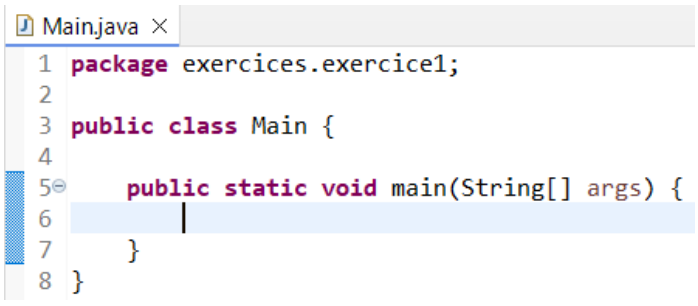

- Nommez votre classe **Main**.



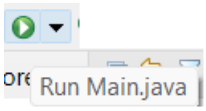

- Cocher l'option **public static void main(String[] args)** pour ajouter la méthode main :



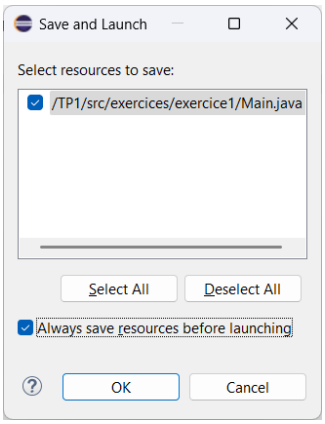

- Cliquez sur **Finish,** vous obtenez le fichier **Main.java** suivant :



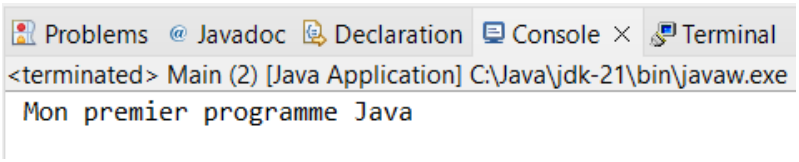

- Tapez **`System.out.println(" Mon premier programme Java ")`** ; puis exécuter votre programme en tapant **CTRL +F11** ou en appuyant sur le bouton Run



- Vous obtenez la fenêtre suivante.




- Cliquez sur OK, vous obtiendrez la sortie suivante sur la console de sortie :




## ➔ Partie B :

1. Dans la méthode **main**, écrire le code Java qui :
    a. Demande à l'utilisateur de saisir deux nombres de type double.
        - Utilisez **Scanner** et sa méthode **nextDouble()**
    b. Affiche le résultat des opérations arithmétiques +, -, *, / et % appliquées sur ces deux nombres.
        - Traitez le cas où la division est impossible.
    c. Affiche le maximum, le minimum des deux nombres.
    d. La racine carrée de chaque nombre. Si le nombre est négatif donnez la racine carrée de sa valeur absolue.
    e. La puissance de l'un par l'autre.

**Remarque : Pour les questions c, d et e, utilisez les méthodes statiques de la classe Math, fournie par Java.**

| Syntaxe | Description |
|---|---|
| `Math.max(a,b)` | Retourne le plus grand des deux nombres `a` et `b` |
| `Math.min(a,b)` | Retourne le plus petit des deux nombres `a` et `b` |
| `Math.abs(x)` | Retourne la valeur absolue de x. |

| | |
|---|---|
| `Math.pow(a,b)` | Calcule a puissance b |
| `Math.sqrt(x)` | Retourne la racine carrée de x |

➔ **Partie C :**

Nous allons maintenant diviser notre programme en deux classes : la classe **Main** (déjà existante) qui contient la méthode **main** et une autre classe qu'on va créer dans le package **exercices.exercice1** et qu'on appellera **Calculatrice**.

1. Cliquez avec le bouton droit sur le package **exercices.exercice1** et choisissez : **New -> class** . Nommez la classe **Calculatrice.**
2. Ajoutez à la classe **Calculatrice** la méthode suivante :

```
public static void Operation(double num1, double num2, char operateur)
{
   // à faire
}
```

   - La méthode affiche en tenant compte de l'opérateur (+, -, *, /, %) le résultat de l'opération arithmétique effectuée=> **utilisez switch/case**
   - Pourquoi la méthode **Operation** doit être **static** ?

3. Dans la méthode **main** :
   a. Ajoutez un menu pour permettre à l'utilisateur de sélectionner une opération arithmétique parmi : addition, soustraction, multiplication, division et reste de division.
      - Utilisez la méthode **Operation** de la classe **Calculatrice**.
   b. Utiliser une boucle **do … while** pour permettre à l'utilisateur d'effectuer plusieurs opérations sans quitter le programme.

## Exercice 2 : Trouvez le nombre secret !!

1. Dans le projet **TP1,** créez un nouveau package que vous nommerez **exercices.exercice2**
2. Dans le package **exercices.exercice2,** créez une nouvelle classe que vous nommerez **Jeu.**
3. Ajoutez à la classe **Jeu** une méthode statique appelée **`trouverNombreSecret()`**

```
public static  void trouverNombreSecret()
{
  // à faire
}
```

Cette méthode doit :

a) Générer un nombre entier aléatoire compris entre 0 et 99 (inclus). Ce sera le nombre secret à deviner.

➔ **Pour obtenir un nombre aléatoire compris entre 0 et 99. Utilisez le code suivant :**

```
Random random = new Random();
int nombreAleatoire = random.nextInt(100);
```

b) Demander à l'utilisateur de saisir un nombre entier entre 0 et 99.

c) Tant que l'utilisateur n'a pas trouvé le nombre secret :

- Si le nombre saisi est plus grand que le nombre secret → afficher "Trop grand ! Essayez encore."
- Si le nombre saisi est plus petit que le nombre secret → afficher "Trop petit ! Essayez encore."
- Redemander à l'utilisateur un nouveau nombre.

d) Quand l'utilisateur trouve le bon nombre, afficher :

- Vous avez trouvé le nombre secret en X coups ! **Remarque :** X est le nombre d'essais avant de trouver la bonne réponse.

4. Dans la méthode **main** de la classe **Main**, située dans le package **exercices.exercice1** :
   - Modifiez le menu pour ajouter l'option "**Trouver le nombre secret**".
   - Appelez la méthode `Jeu.trouverNombreSecret()` lorsque l'utilisateur choisit cette option.

➔ **Attention, la classe Jeu se trouve dans un autre package que la classe Main.**

1. Vérifiez que la classe, **Jeu**, est déclarée avec le mot-clé **public**.
2. Pour utiliser **Jeu** dans **Main** :
   - Soit, vous utilisez son nom complet qui comporte le nom du package ou elle est située.
     ⇨ `exercices.exercice2.Jeu`
   - Soit, vous utilisez l'instruction **import** en haut de votre fichier **Main.java**
     ⇨ `import exercices.exercice2.Jeu ;`

## Exercice 3 : Analyser une pharse

1. Dans le projet **TP1,** créez un nouveau package que vous nommerez **exercices.exercice3**
2. Dans ce package, créez la classe **AnalysePhrase**
3. Ajoutez dans cette classe :

***a)*** Une méthode **`public static int nombreDeMots(String phrase)`** qui compte et retourne le nombre de mots dans une phrase. ***Une phrase est une séquence de mots séparés par des espaces.***

**Idée :**

- Appliquez la méthode **split("\\s+ ")** sur la phrase pour obtenir un tableau de mots, puis utilisez **length** pour obtenir le nombre d'éléments de ce tableau.

| Syntaxe | Description |
|---|---|
| **String [] mots = phrase.split("\\s+ ") ;** | La méthode **split** divise une chaîne en plusieurs sous-chaînes, en fonction d'un séparateur spécifié. Ici, le séparateur est l'espace.<br>La méthode retourne un tableau stockant les sous-chaînes de caractères après division. |

b) Une méthode **public static String remplacerVoyelle(String phrase, char symbol)** qui remplace chaque voyelle par le symbole passé en paramètre.

**Idée :** Appliquez la méthode **replace,** vue dans le cours. Vous pouvez utiliser aussi cette méthode qui vérifie si un caractère est une voyelle.

```
public static boolean estVoyelle(char c) {
    return "aeiouAEIOU".indexOf(c) != -1;
}
```

c) Une méthode **public static String inverserPhrase(String phrase)** qui retourne la phrase avec l'ordre des mots inversé.

**Idée :**

- Utilisez **split** pour obtenir le tableau contenant les mots de la phrase.
- Inversez l'ordre des mots en les stockant dans un autre tableau "reverse_tab"
- Utilisez la méthode **String.join** pour construire la phrase inverse.

| Synatxe | Description |
|---|---|
| String.join(" ", reversed_tab) | Assemble les mots du tableau reversed_tab en une seule chaîne, en insérant un espace entre un mot et un autre. Elle retourne la chaîne assemblée. |

4. Dans la méthode **main** de la classe **Main**, située dans le package **exercices.exercice1** :
   a) Modifiez le menu pour ajouter l'option "**Analyser Phrase**".

b) Dans le bloc correspondant à cette option :

- Demandez à l'utilisateur de saisir une phrase.
- Affichez le nombre de caractères de cette phrase.
- Affichez la première et la dernière lettre.
- Affichez le nombre de mots de la phrase ; appelez la méthode **nombreDeMots.**
- Affichez la phrase inversée ; appelez la méthode **inverserPhrase.**
- Affichez la phrase après avoir remplacé les voyelles, par un symbole choisi. Appelez la méthode **remplacerVoyelles.**