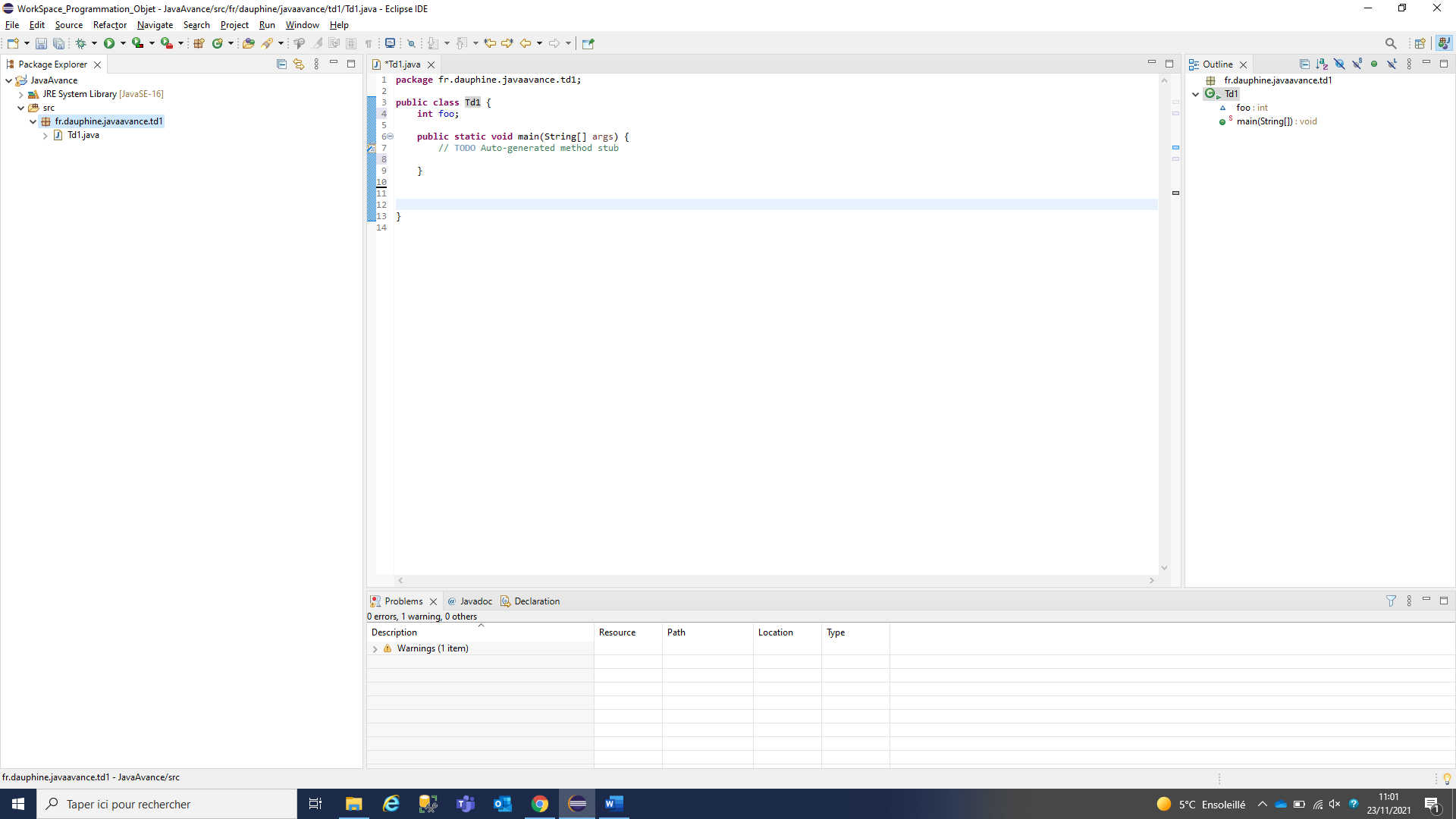
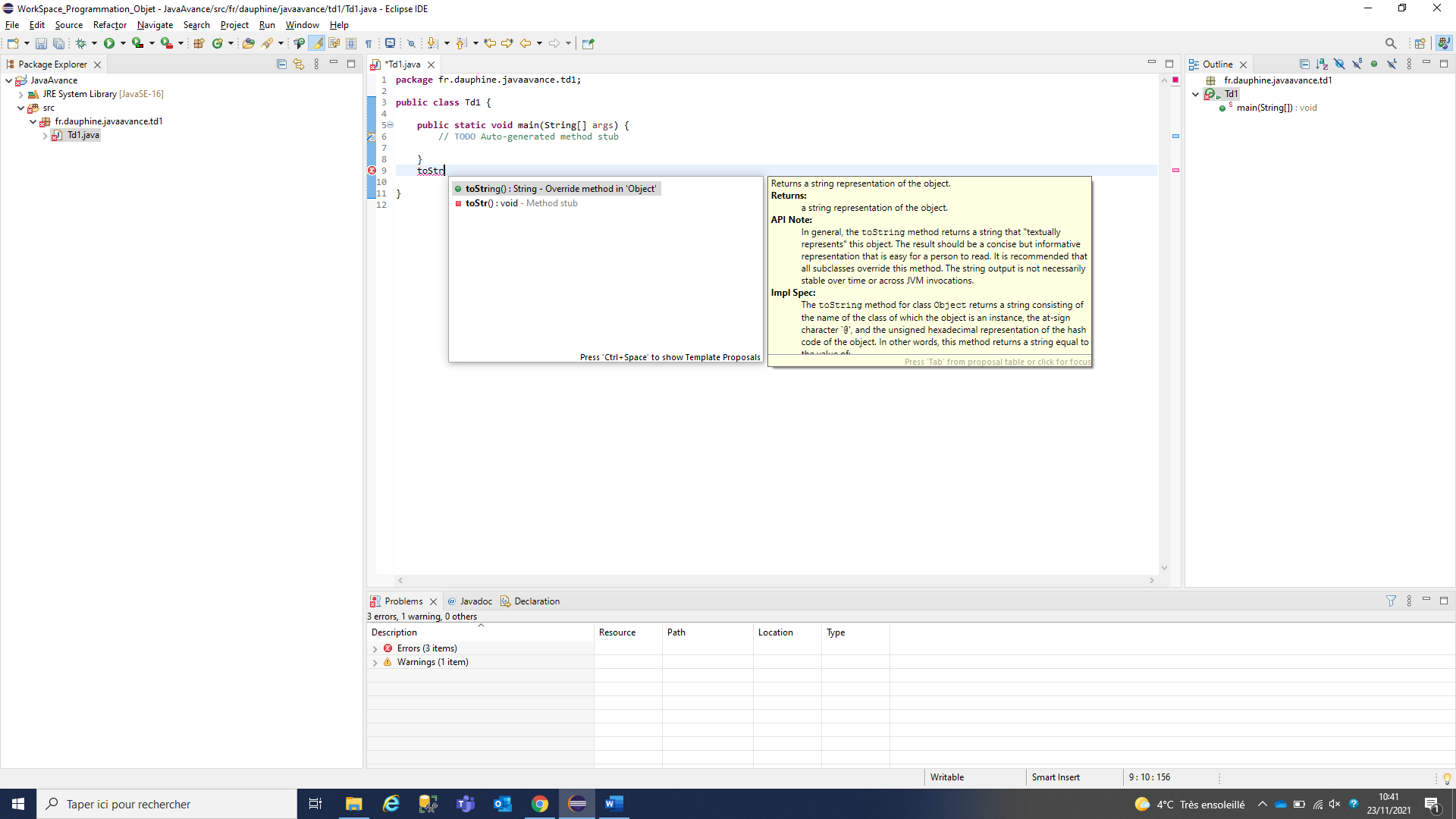
# Exercice 1 :





1. Cela nous propose de généré le code de la fonction suivante : System.***out***.println();
2. Cela nous propose de Override une méthode toString ou de créer une méthode toStr de type void. (Voir image ci-dessous)

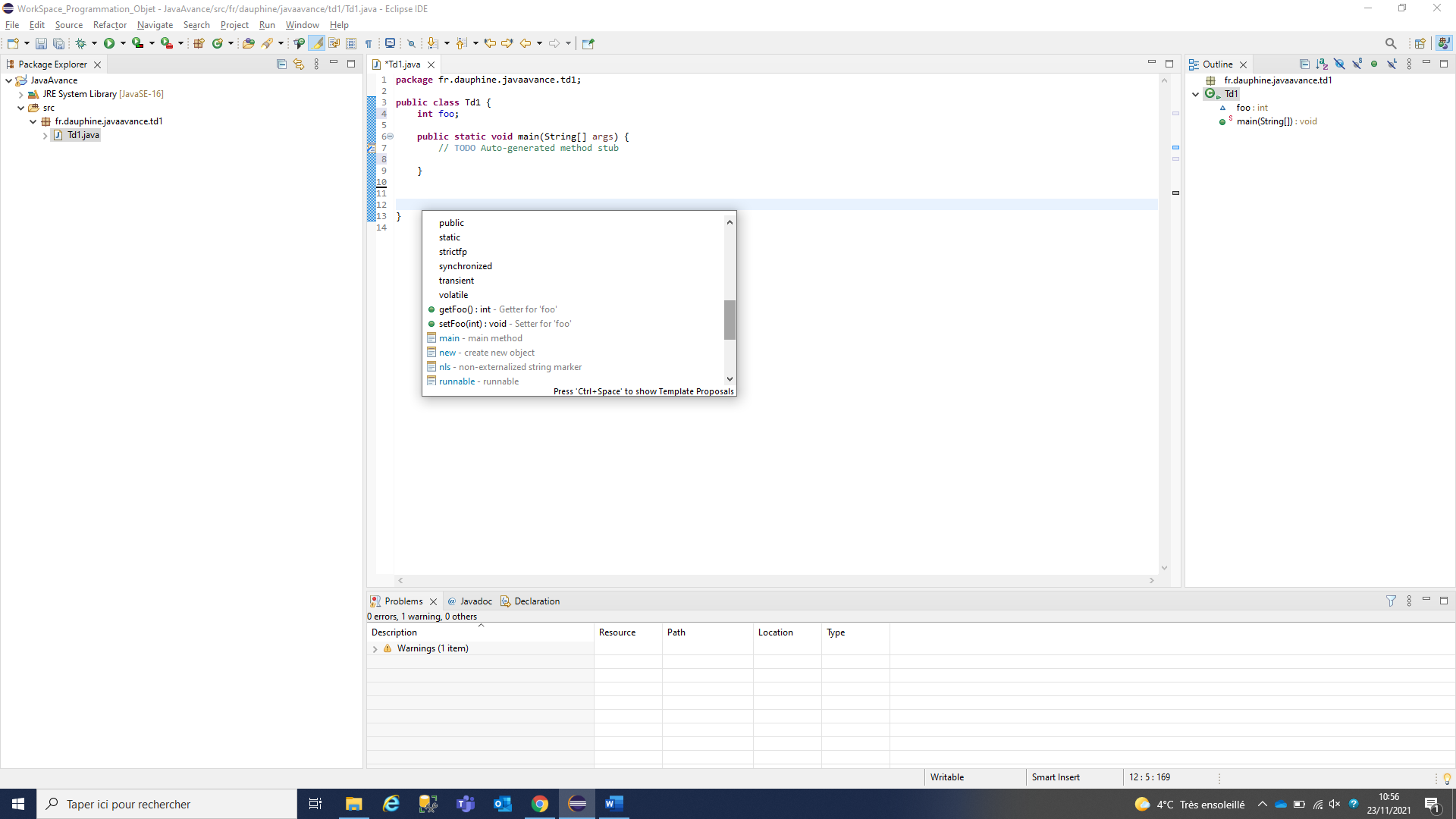


1. Cela nous propose de créer la méthode main.

**public** **static** **void** main(String[] args) {

}

1. Si on tape ctrl + espace, Eclipse nous propose de créer un getter et un setter pour la variable int foo :



Et pour lorsque l’on écrit set puis on fait ctrl + espace, Eclipse nous propose de créer la méthode setFoo :

**public** **void** setFoo(**int** foo) {

**this**.foo = foo;

}

1. Cela nous propose d’utiliser l’outil d’Eclipse « Refactor Rename » qui permet de changer le nom de la classe, et cela va modifier le nom de la classe dans toutes les méthodes où il est appelé.

# Exercice 2 :

1. Ça fonctionne car mêmes si les attributs sont privés, on peut y accéder car ils sont dans la même classe. De plus il existe un constructeur par défaut qui va initialiser les variables x et y à zéro, d’où l’affichage en sorte : 0 0
2. Ça ne va pas fonctionner car on ne peut pas accéder à des attributs privés d’une autre classe. Pour résoudre ce problème, on peut créer des Getters dans la classe Point, ce qui va permettre d’accéder aux attributs.
3. Pour encapsuler les attributs. Cela va permettre de protéger les données de l’extérieur et de contrôler l’accès aux composants critiques de l’objet.
4. C’est une méthode permettant d’accéder à la valeur d’un attribut private tout en le protégeant des modifications extérieures. Oui on doit le faire car les attributs x et y sont privés.
5. Le problème est que ça ne compile plus car il n’y a plus de constructeur par défaut (il est écrasé par le nouveau constructeur), en effet dorénavant le constructeur prend 2 paramètres en entré.
6. Cela peut créer une ambigüité entre les paramètres du constructeur et les attributs de la classe, il faut donc utiliser « this » pour bien différencier les champs.
7. On peut créer une variable static qui sera incrémenter à chaque appel au constructeur.