22/01/2016

Projet de Refactoring de CPOA

Baptiste CRESPIN - Nicolas CELLIER

baptiste-crespin@univ-tlse2.fr | nicolas-cellier@univ-tlse2.fr

Sommaire

Table des matières

[I – Refactoring 2](#_Toc441261493)

[1 - Mise en place des bonnes pratiques 2](#_Toc441261494)

[2 - Projet.java 2](#_Toc441261495)

[3 – Application.java 2](#_Toc441261496)

[II – Ajout de fonctionnalités 3](#_Toc441261497)

[1 – Deadlines 3](#_Toc441261498)

[a) Date limite 3](#_Toc441261499)

[b) Today 4](#_Toc441261500)

[2 – Suppression 4](#_Toc441261501)

[3 – Visualisation 5](#_Toc441261502)

[a) View by date 5](#_Toc441261503)

[b) View by deadline 5](#_Toc441261504)

[c) View by project 6](#_Toc441261505)

[4 - Permettre qu’une tâche puisse « appartenir » à plusieurs projets en même temps 6](#_Toc441261506)

[5 - Faire en sorte que l’application maintienne en permanence 2 listes de tâches par projet 7](#_Toc441261507)

[6- Faire en sorte qu’une tâche puisse être elle-même une liste de tâche 7](#_Toc441261508)

[III – Tests 7](#_Toc441261509)

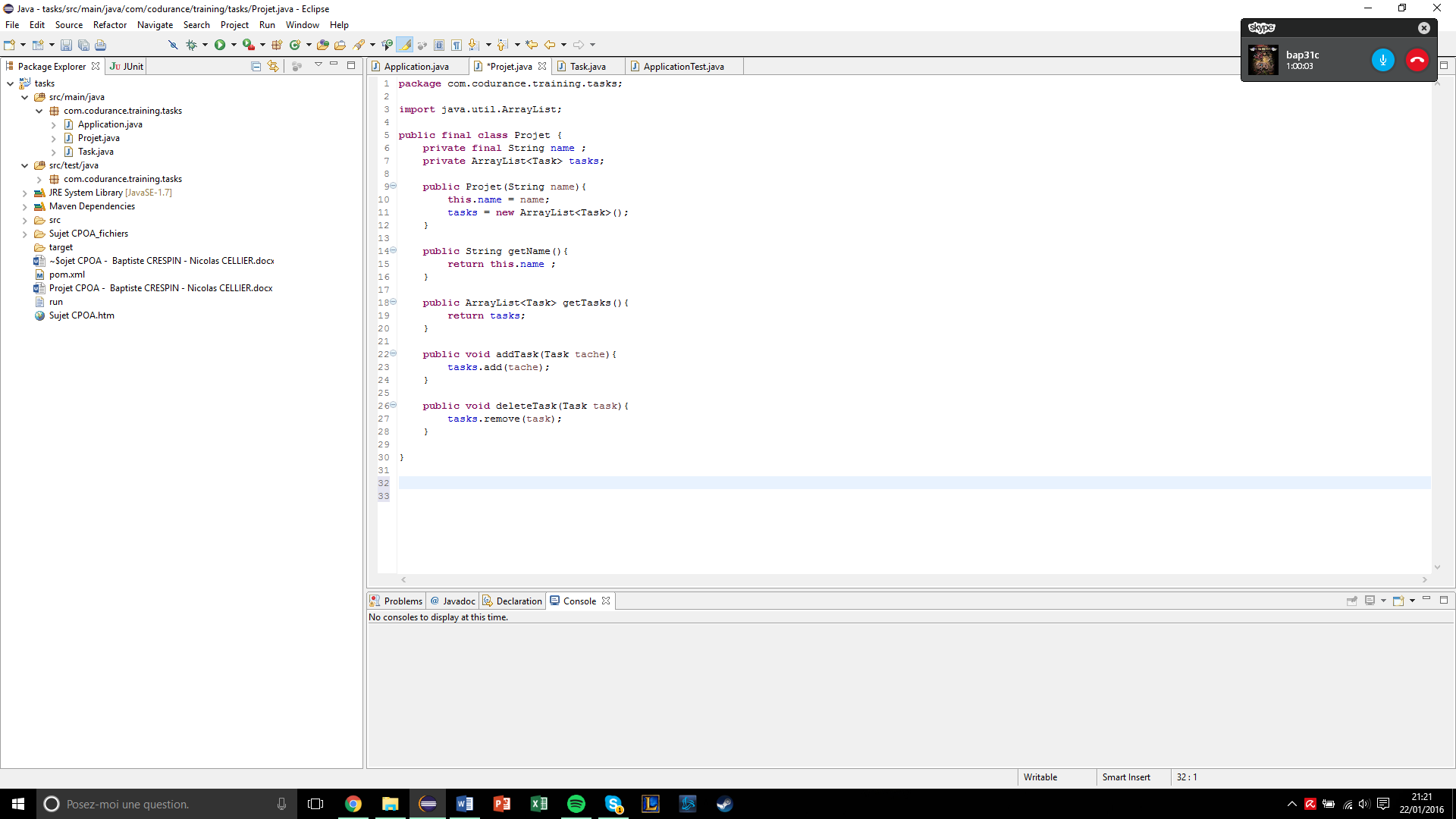
# I – Refactoring

## 1 - Mise en place des bonnes pratiques

Nous avons fait le choix de débuter le projet par améliorer le plus possible le code déjà existant, afin d’avoir de bonnes bases avant de démarrer, et faciliter le travail ultérieur, et l’ajout de fonctionnalités futures.

## 2 - Projet.java

Nous avons commencé par modifier le code existant pour utiliser un objet Projet, plutôt qu’une liste de taches. Pour cela nous avons créé la class Projet, qui a pour attribut un nom, ainsi qu’une liste de projets. Il y a un getter pour chacun de ses attributs, et on peut ajouter ou supprimer des taches du projet.



## 3 – Application.java

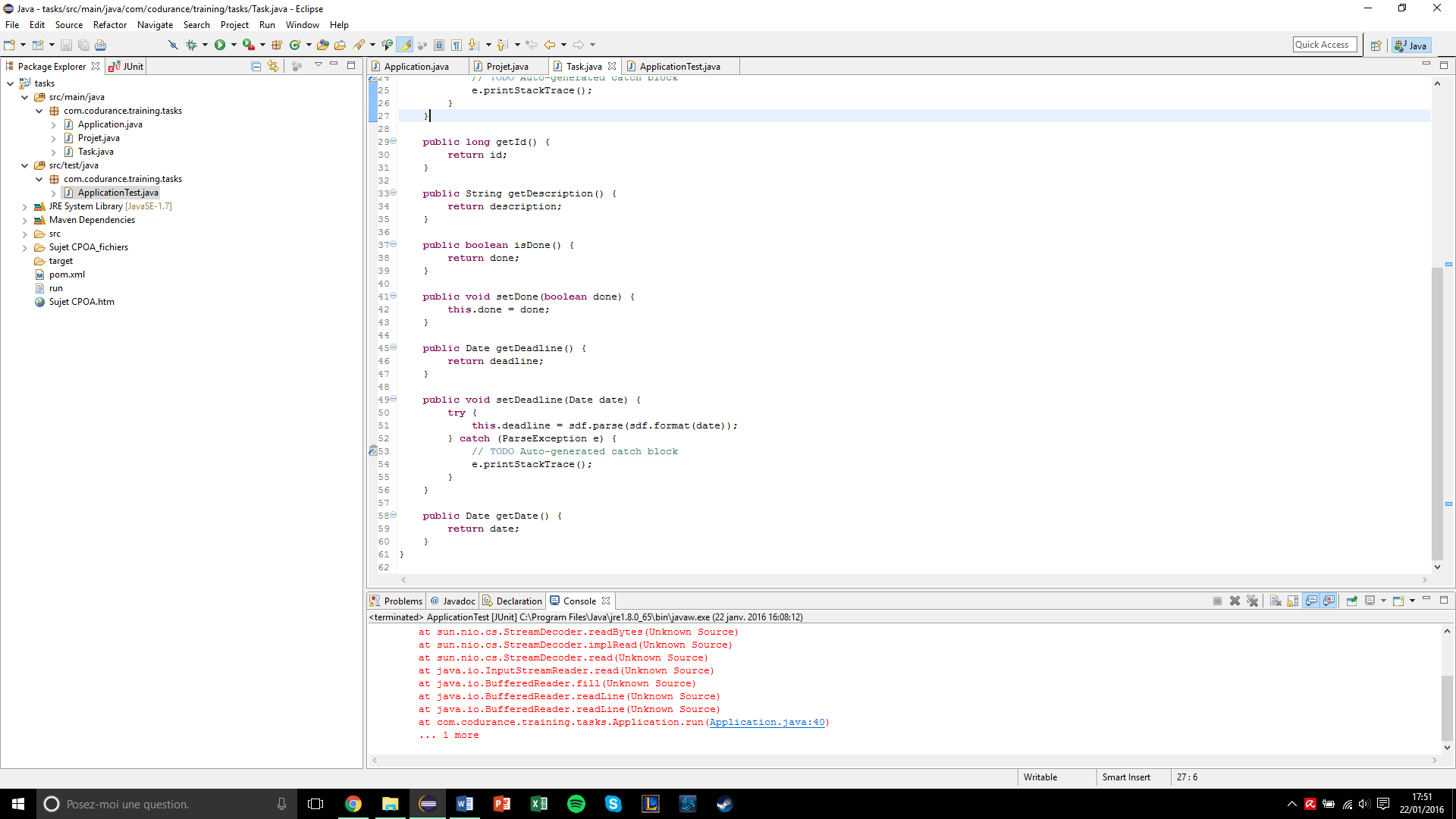
Nous avons ensuite modifié application.java, afin que celle-ci soit en adéquation avec la classe Projet.

# II – Ajout de fonctionnalités

## 1 – Deadlines

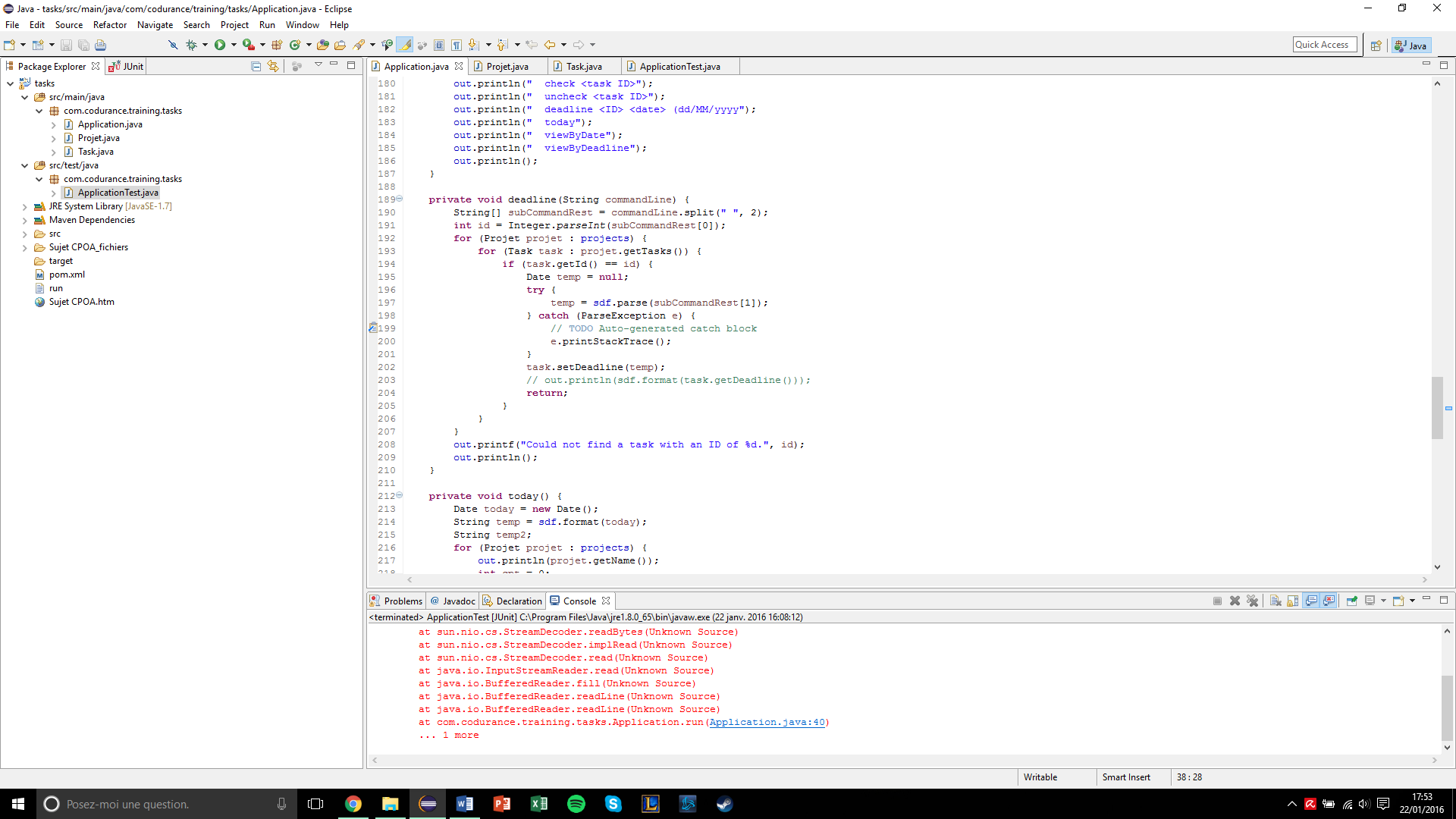
### Date limite

Pour l’ajout de la fonctionnalité deadline, nous avons commencé par ajouter dans Task.java un attribut de type private Date deadline dans la classe Task, ainsi qu’un getter et un setter.



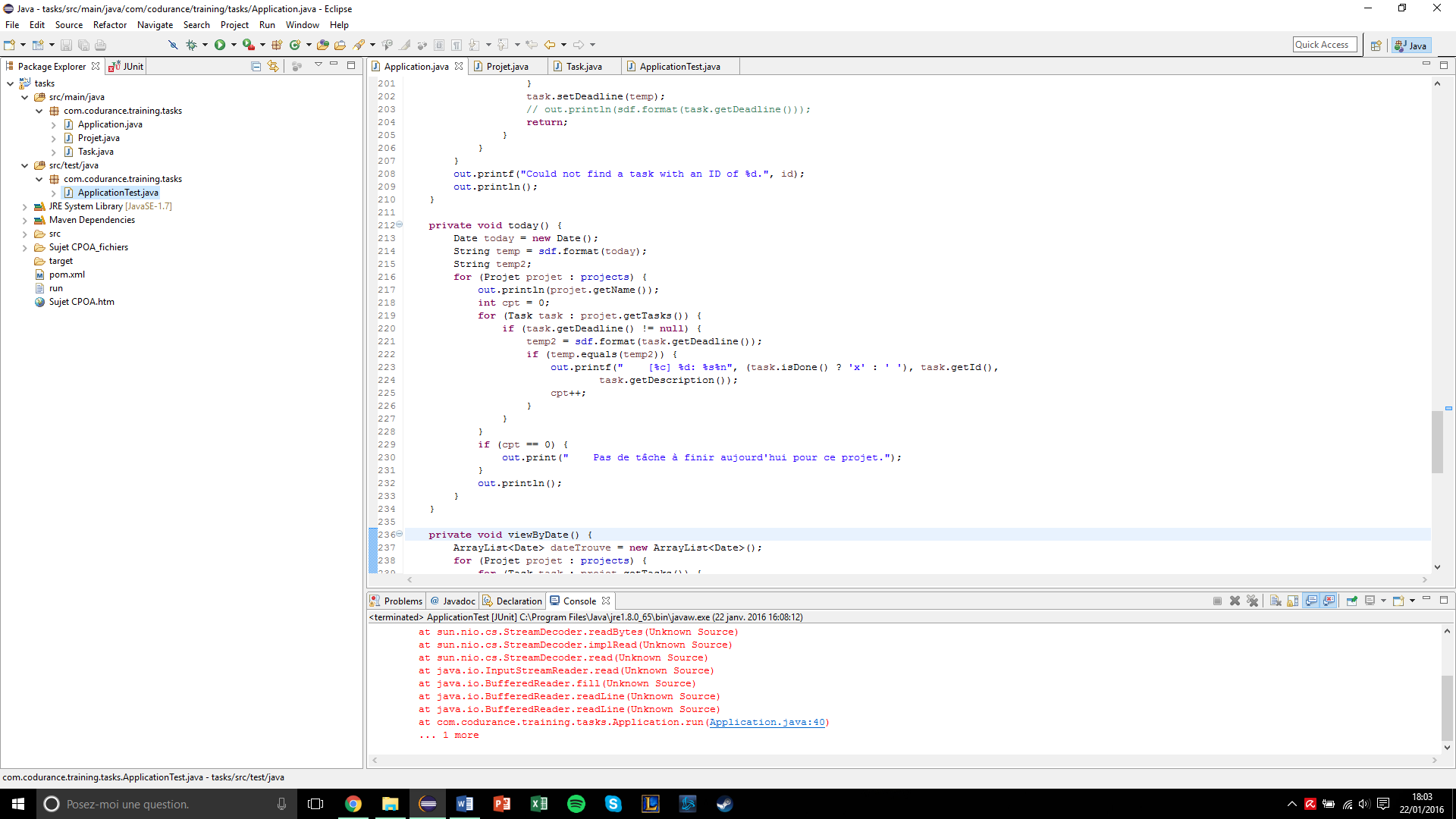
Puis nous avons initialisé celui-ci à null dans le constructeur par défaut.

Ensuite, après avoir ajouté les choix correspondant dans le menu, ainsi que l’aide, nous avons ajouté la méthode deadline, permettant de modifier la deadline d’une tache en donnant en paramètre l’ID de la tache ainsi que la date de la nouvelle deadline. Pour cela, elle parcourt toutes les taches de tous les projets jusqu’à rencontrer une tache à l’ID correspondant au paramètre. Puis une fois cette tâche trouvée, elle modifie sa deadline.



### Today

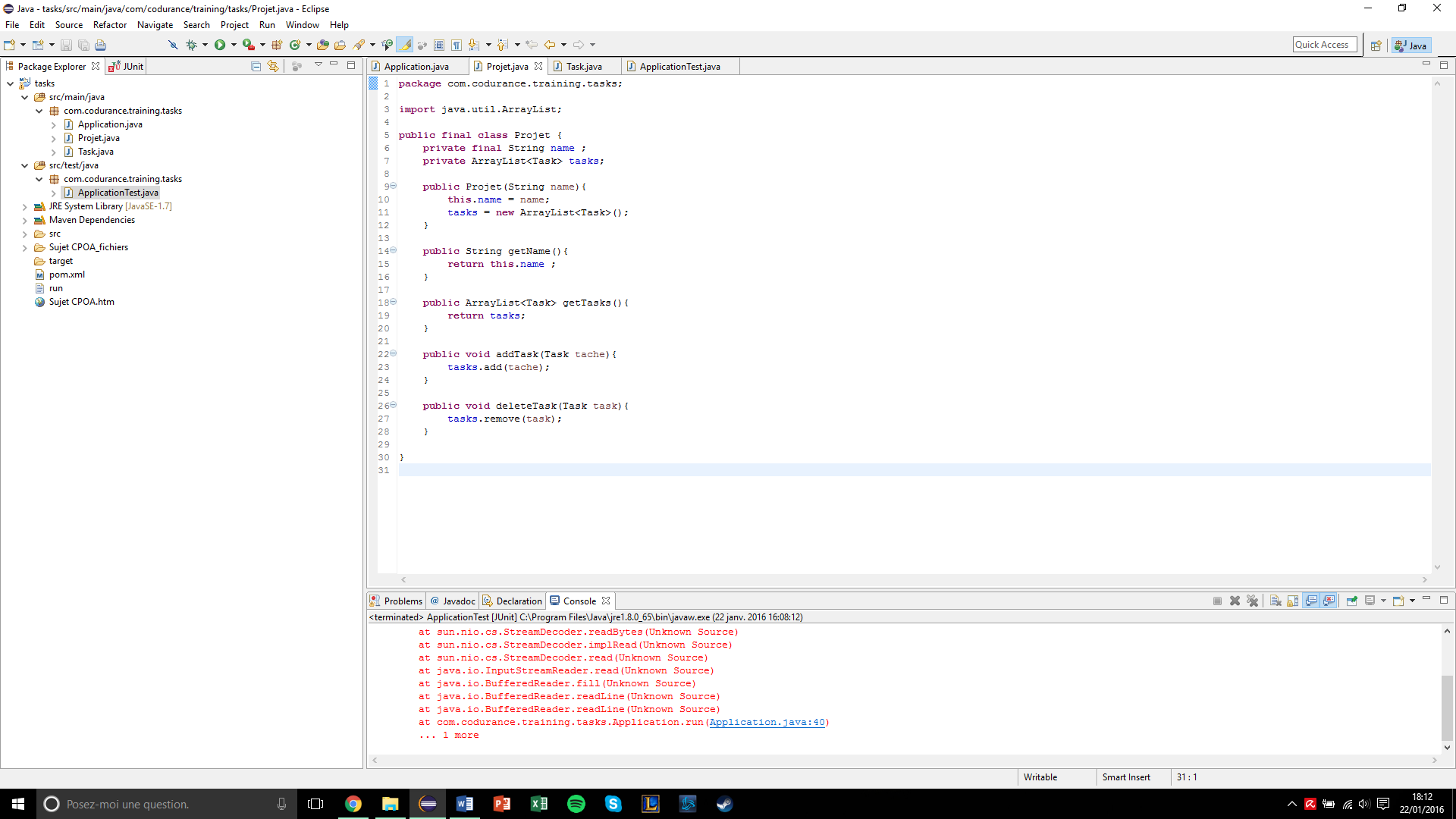
La commande today permet d’afficher toutes les taches dont la deadline est située aujourd’hui. Pour cela, elle va parcourir toutes les taches de tous les projets en affichant la tache au passage si la deadline de celle-ci est égale à celle d’aujourd’hui.



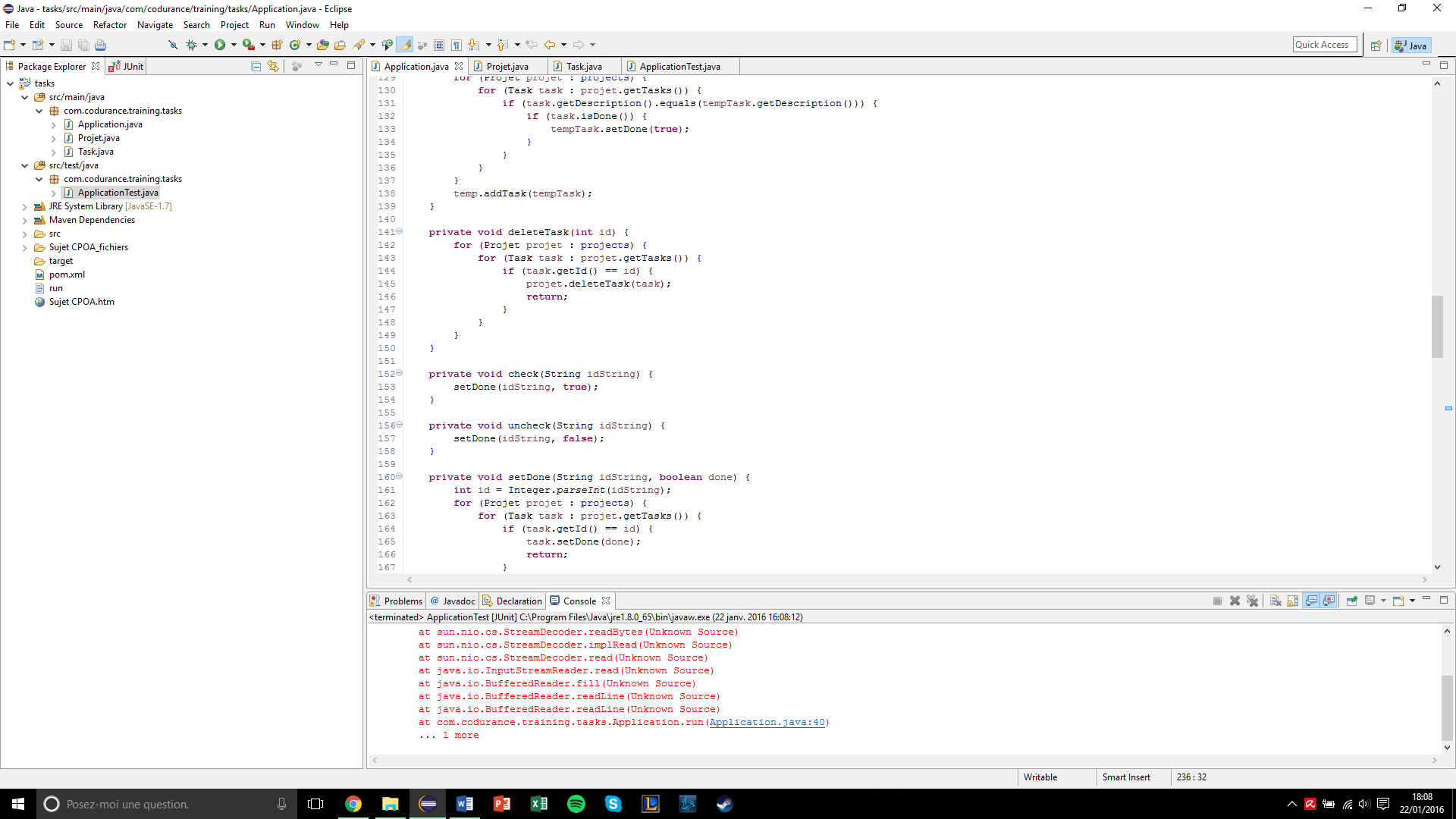
## 2 – Suppression

Pour implémenter la suppression d’une tâche, nous avons, après avoir ajouté les choix correspondant dans le menu, ainsi que l’aide, ajouter la méthode deleteTask, qui parcours toutes les taches de tous les projets jusqu’à trouver une tache ayant un ID correspondant. Si l’ID correspond, la tâche est supprimée du projet à l’aide de la fonction deleteTask de projet :

deleteTask de projet :



deleteTask de l’application :

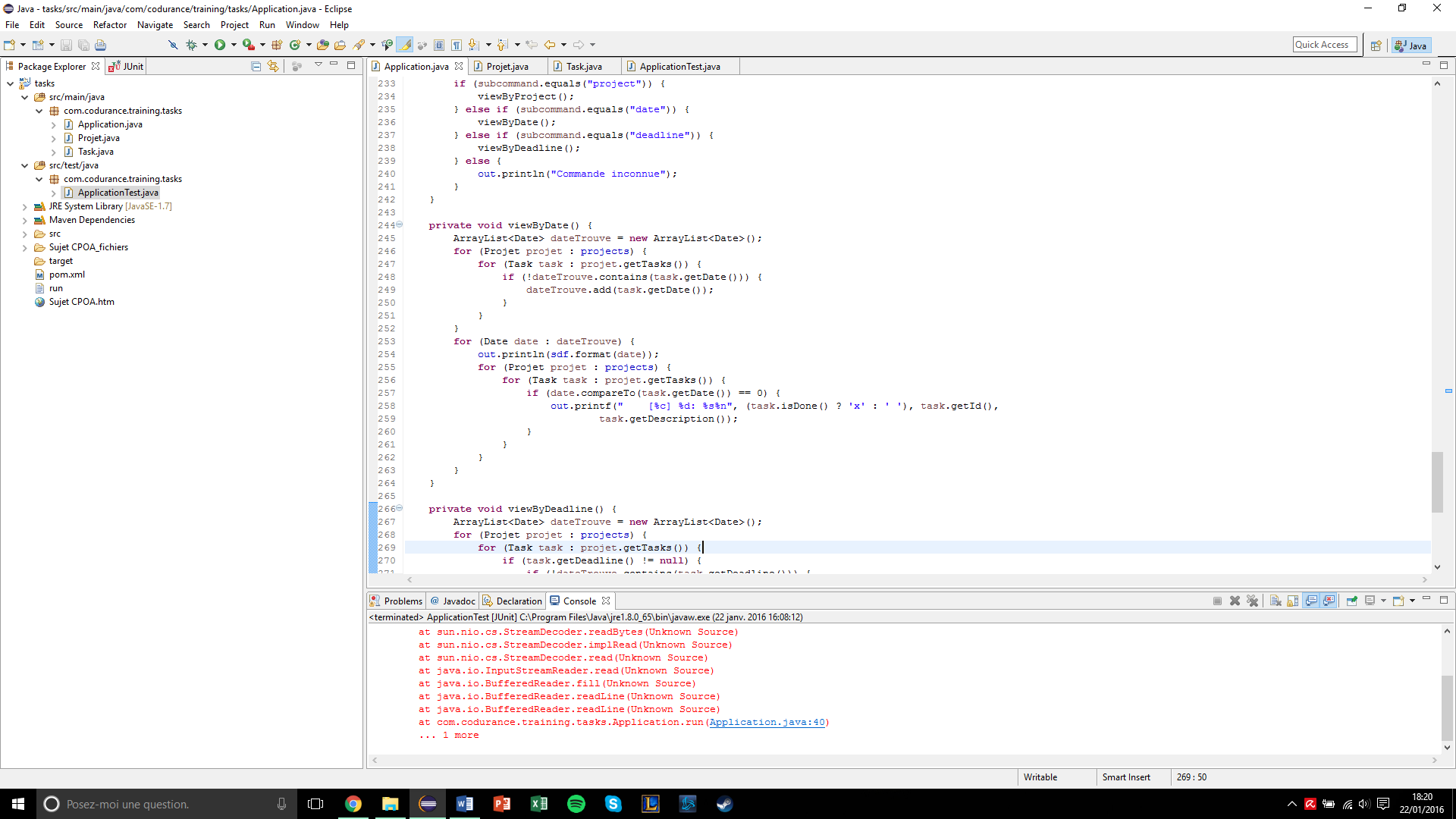


## 3 – Visualisation

### View by date

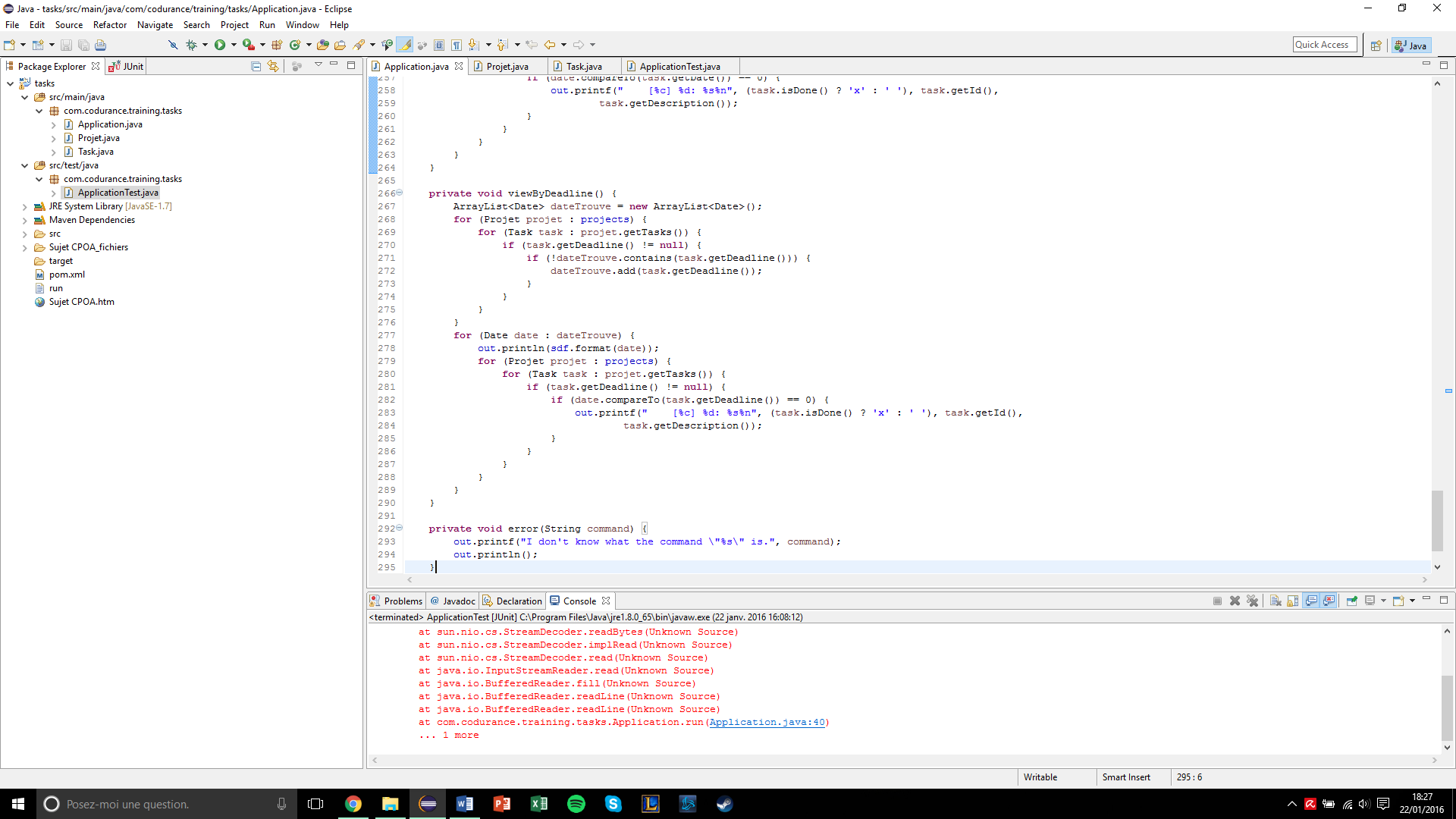
Pour l’ajout de la fonctionnalité view by date, nous avons commencé par ajouter dans Task.java un attribut de type private Date date dans la classe Task, ainsi qu’un getter. Cette date est automatiquement initialisée à la date d’aujourd’hui au moment de la création d’une tâche.

Ensuite, après avoir ajouté les choix correspondant dans le menu, ainsi que l’aide, nous avons ajouté la méthode viewByDate, permettant d’afficher toutes les taches, en fonction de leur date de création. Pour cela, la méthode parcourt toutes les taches de tous les projets en ajoutant leur date dans une liste, dans leur ordre de création. Puis, on parcourt la liste des dates, en affichant chacune des taches correspondantes.



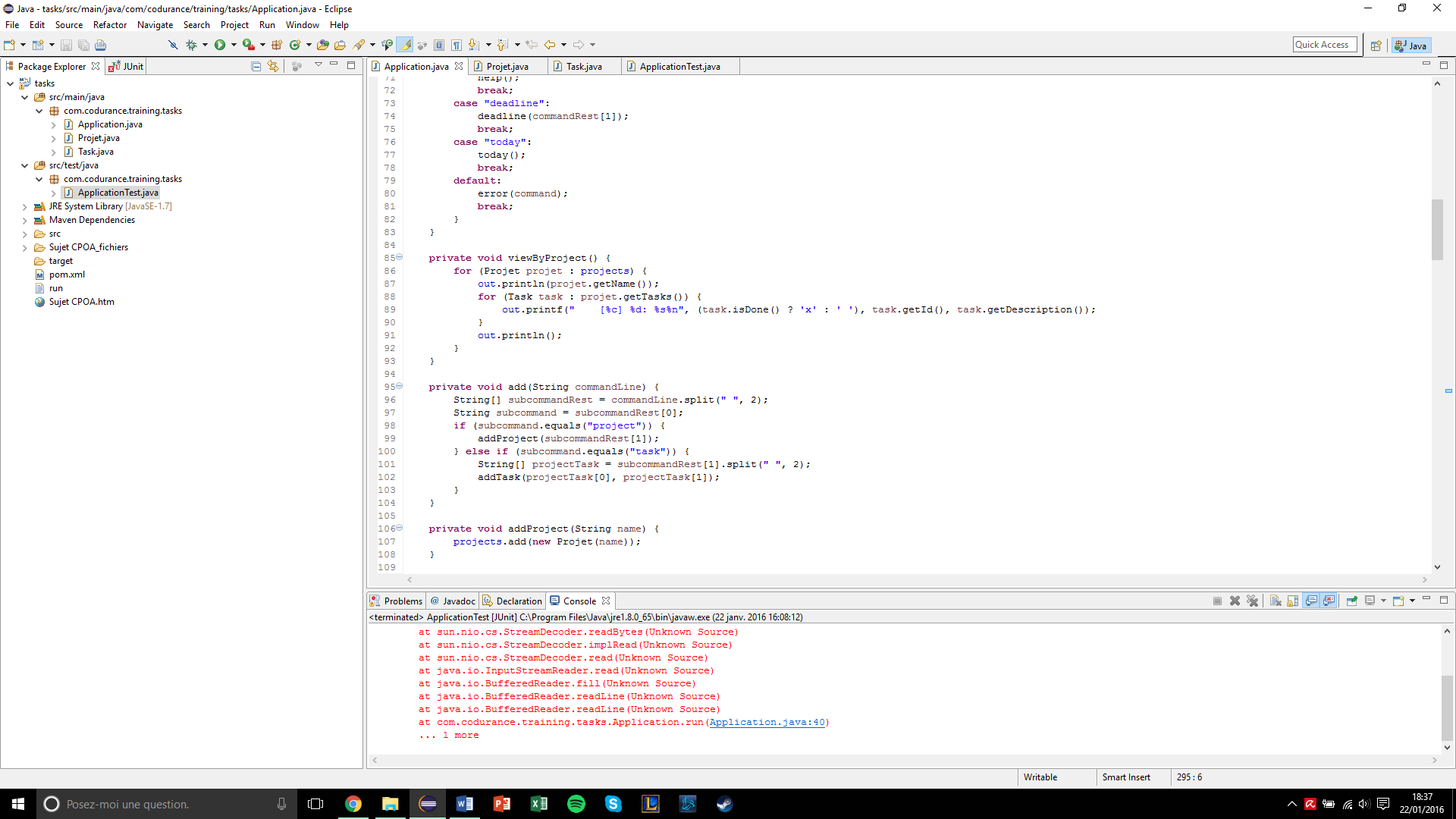
### View by deadline

View by deadline reprends le même principe que view by date, mais en utilisant cette fois ci les deadlines des projets. Cependant, toutes les tâches n’ont pas forcement de deadline, il a donc fallu ajouter quelques conditions afin de vérifier que deadline soit différent de null.



### View by project

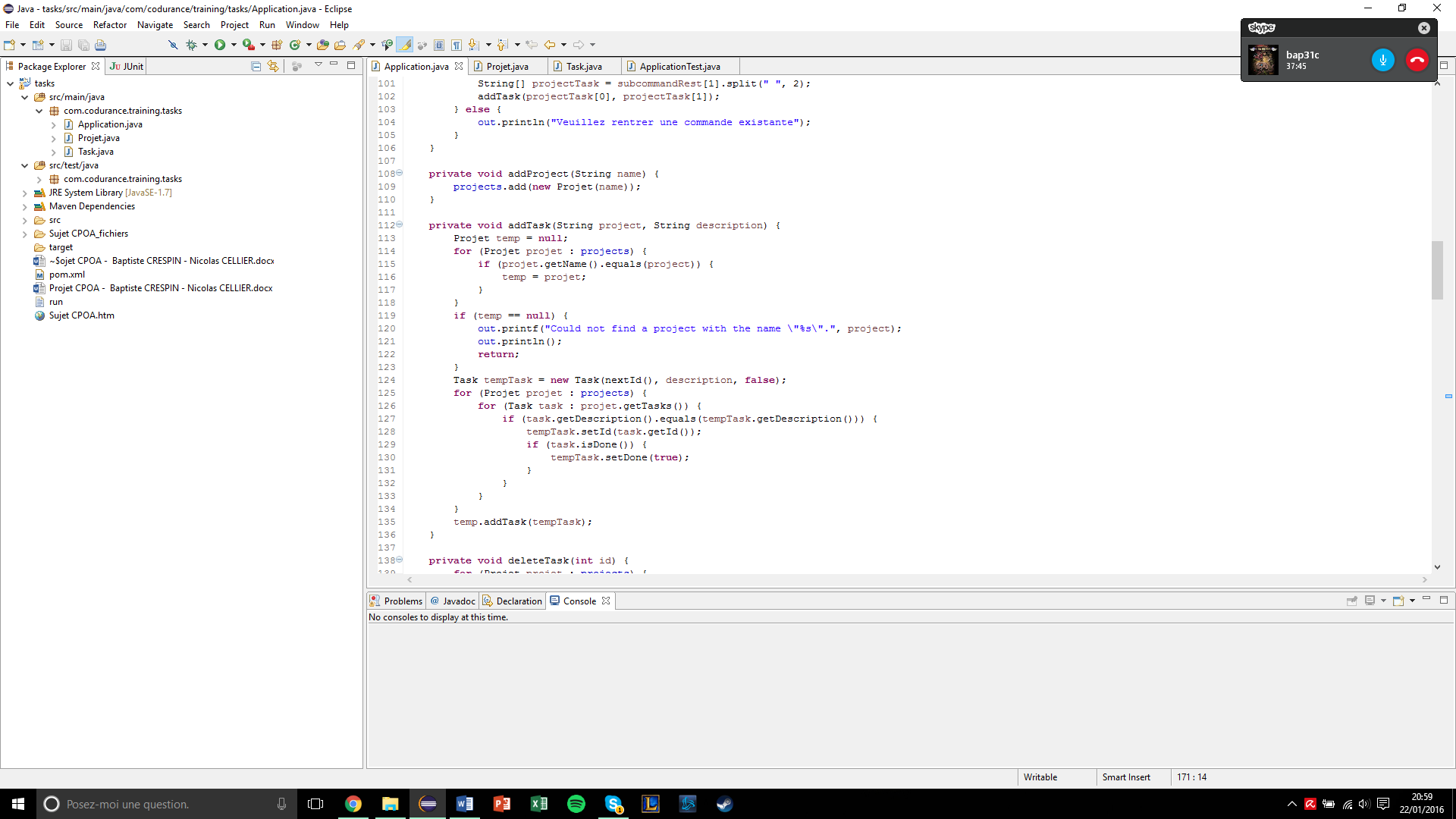
La méthode view by project existait déjà sous le nom show. Il n’y a donc pas eu de modification à apporter hormis dans le menu et l’aide.



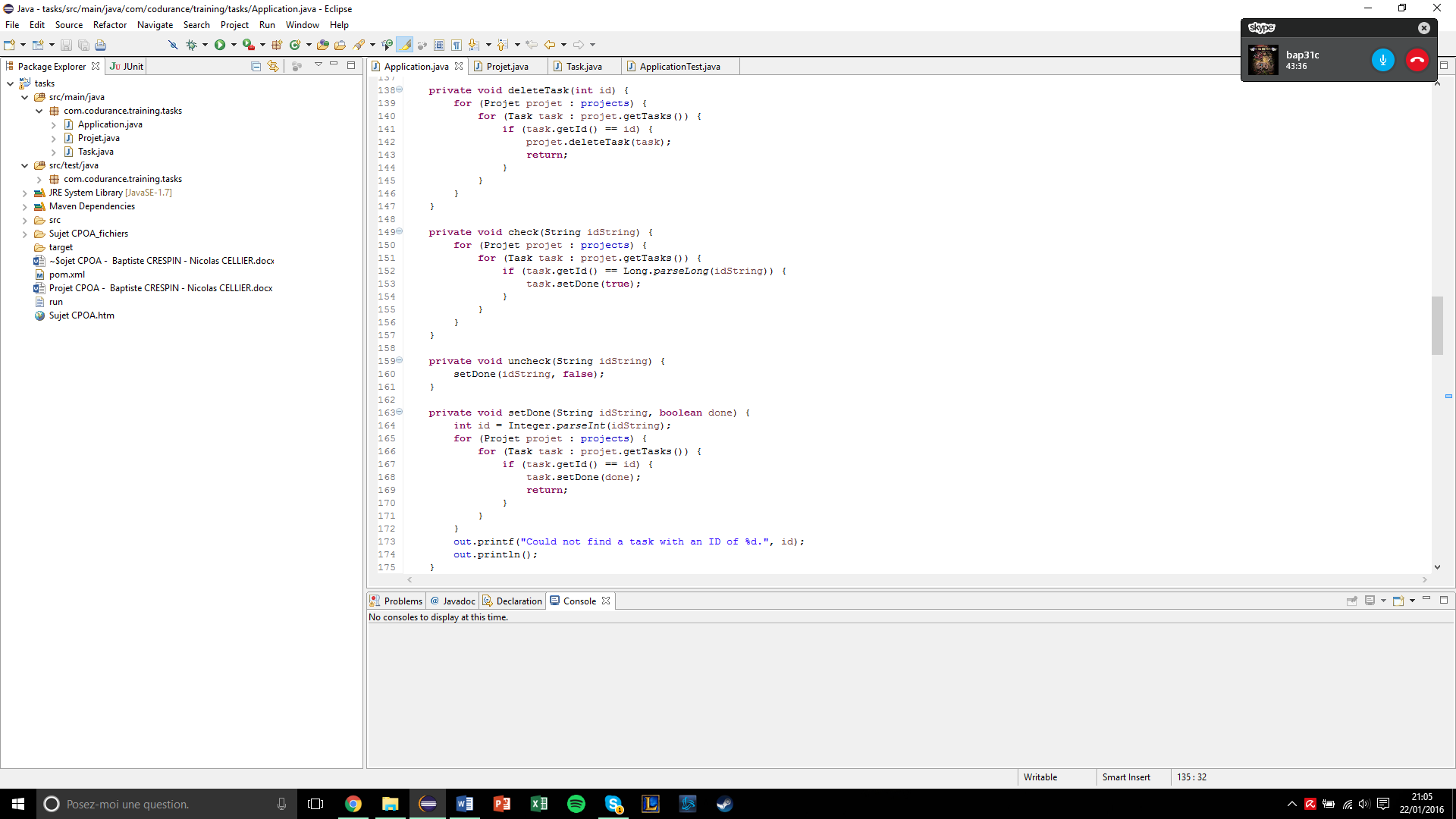
## 4 - Permettre qu’une tâche puisse « appartenir » à plusieurs projets en même temps

Pour implémenter cette fonctionnalité, nous avons modifié la méthode addTask, afin de permettre au programme d’ajouter des taches dans différents projets. Pour cela avons ajouté que la méthode crée une nouvelle tâche, et si celle-là a le même nom qu’une tache déjà existante, elle change son ID afin que les taches correspondent.

Nous avons dû également modifié Task.java, afin de faciliter l’accès à l’ID et le rendre modifiable, nous avons enlevé le final de l’attribut ID, et ajouté un setter.



Afin de faire en sorte que lorsqu’une tache est cochée dans un projet, elle le soit aussi dans tous les autres projets, nous avons dû également modifier la méthode check, afin qu’elle coche toutes les taches de même ID :



## 5 - Faire en sorte que l’application maintienne en permanence 2 listes de tâches par projet

## 6- Faire en sorte qu’une tâche puisse être elle-même une liste de tâche

Nous n’avons malheureusement pas eu le temps d’implémenter cette fonctionnalité.

# III – Tests

Nous avons rencontré un problème lors de l’exécution des tests, ils ne marchent pas.