

Rapport  
PROJET JAVA – SEMESTRE 1

aubry Rodolphe, hermal Vincent

Table des matières

[Présentation du projet 1](#_Toc408430420)

[Diagramme de classes 1](#_Toc408430421)

[Etapes de réalisation du projet 2](#_Toc408430422)

[Documentation du code source : Les points essentiels 2](#_Toc408430423)

[Avancement des voitures 2](#_Toc408430424)

[L’avarie et la panne 3](#_Toc408430425)

[Verification du réservoir 3](#_Toc408430426)

[Arrêt aux stands 4](#_Toc408430427)

[Dépassement et collision 4](#_Toc408430428)

[Conclusion 5](#_Toc408430429)

# Présentation du projet

Le projet nous ayant été donné de faire, en utilisant le langage Java était d’écrire un programme permettant de simuler une saison de Formule 1. Les concurrents se composent de 10 écuries de 2 voitures chacune. De nombreux évènements se déroulent au cours d’une course comme par exemple le réservoir qui se vide au fil de la course, mais aussi des chances de tomber en panne ou d’entrer en collision avec une autre voiture lors d’un dépassement., ce qui rend cette course « vivante » et rapproche cette simulation de la réalité.

## Diagramme de classes

## Etapes de réalisation du projet

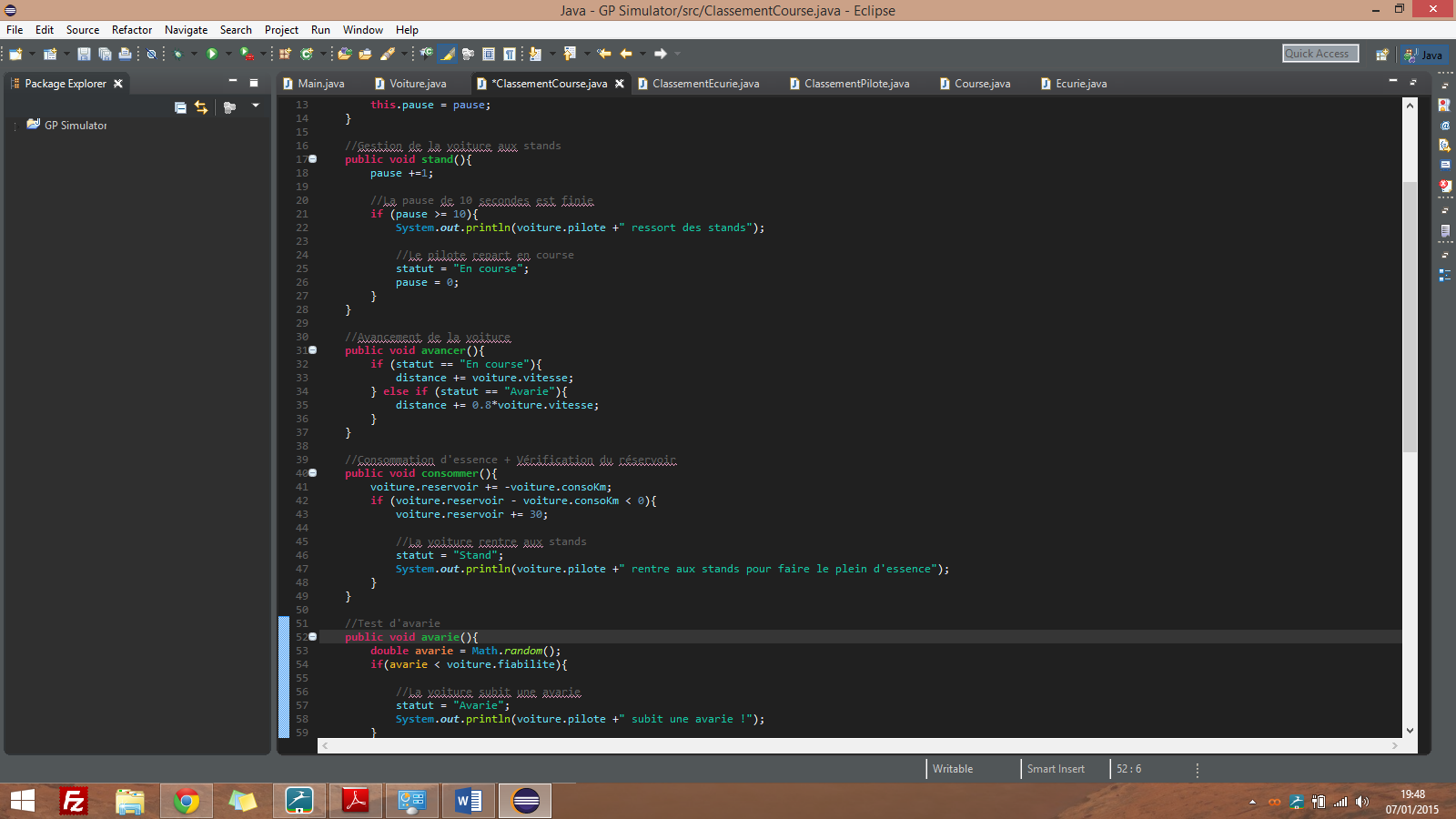
Nous avons divisé notre travail en plusieurs étapes. Pour commencer, nous avons fait un premier diagramme de classe sur lequel nous nous sommes basé lors du codage.

Nous avons dû procéder à quelques changements au niveau du diagramme car certains attributs supplémentaires ont été nécessaires comme par exemple l’attribut statut pour connaître la situation de la voiture au sein d’une course (est-il encore en course ou non par exemple).

Au niveau du codage, nous avons commencé par la réalisation des classes ainsi que leurs constructeurs en Java à l’aide d’Eclipse, l’IDE que nous avons tous les deux utilisé. Nous avons donc créé la base de données nécessaire à la simulation comme les circuits, les pilotes, les écuries, etc. Une fois cette base créée, nous avons d’abord fait avancer les voitures simplement avant d’y ajouter les évènements. Il a ensuite fallu créer la gestion des multiples évènements qui peuvent intervenir au sein d’une course. Une fois ces tâches terminées, il ne nous restait plus qu’à ajouter les points aux différents classements ainsi que simuler cette gestion sur toute une saison.

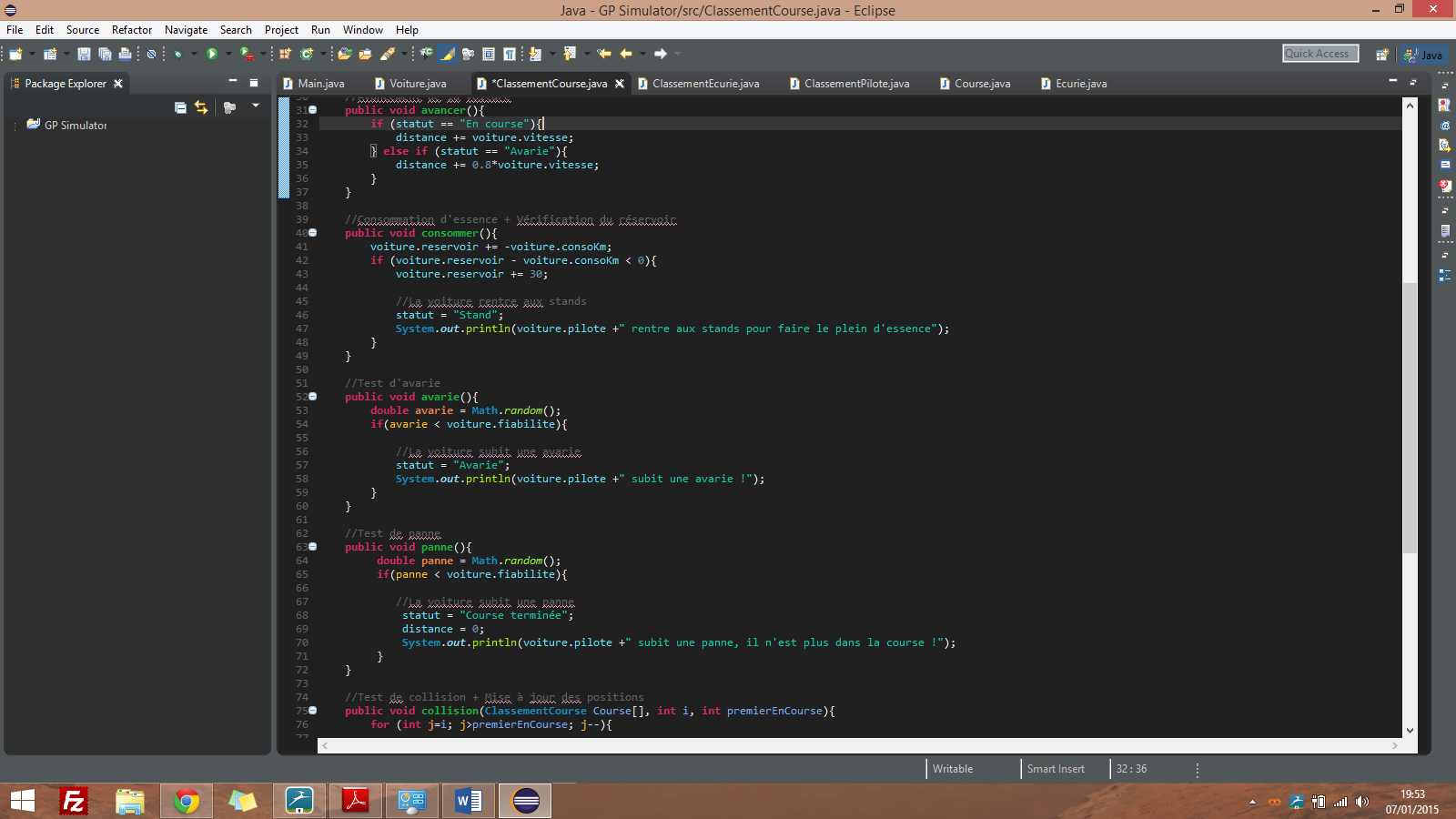
## Documentation du code source : Les points essentiels

### Avancement des voitures



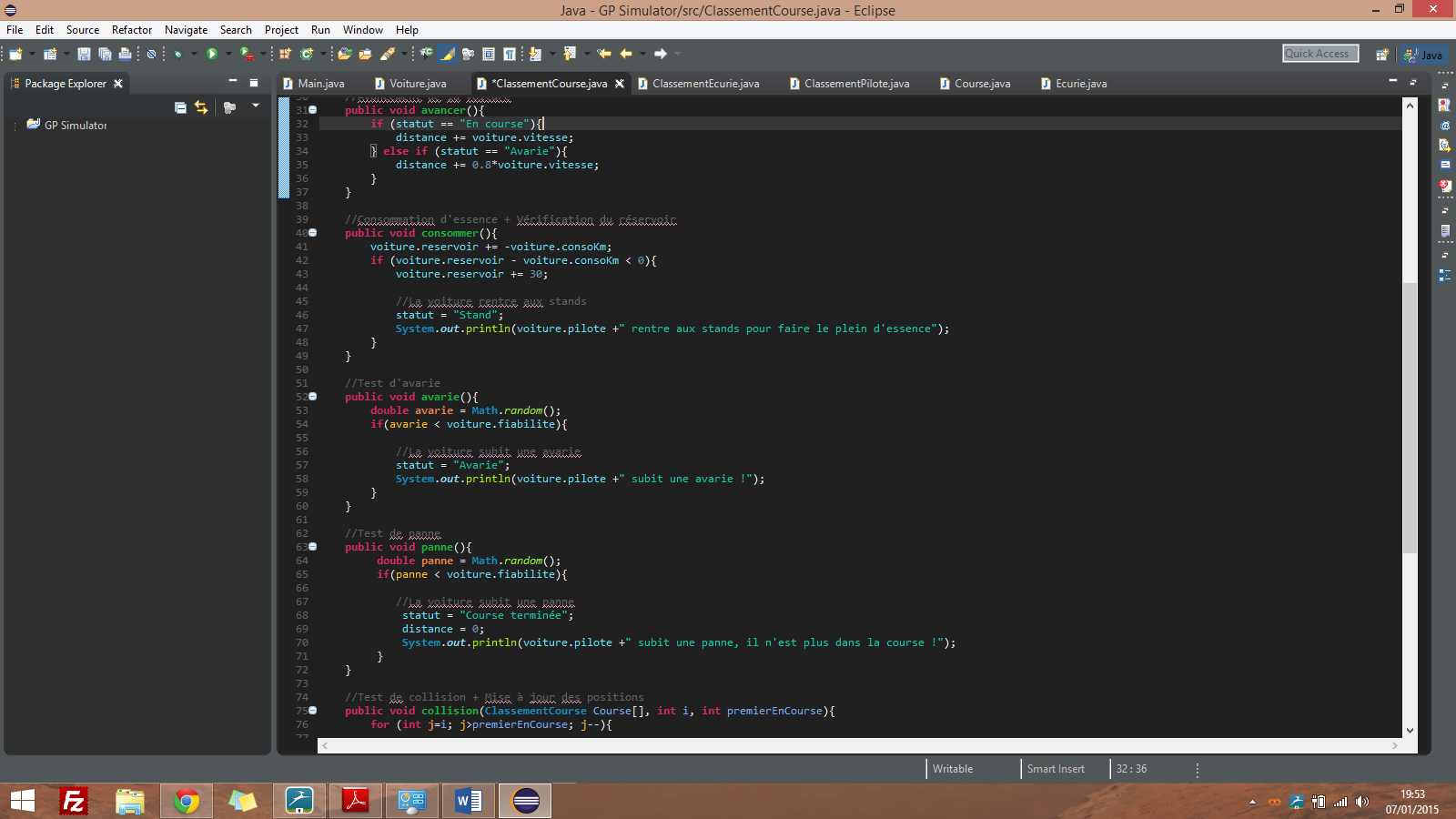
La méthode « avancer() » est forcément à la base de cette simulation puisque sans avancement il n’y a pas de course. Si la voiture n’a pas de problème, elle avance normalement, par contre, s’il est subit une avarie, sa vitesse est réduite jusqu’au tour suivant.

### L’avarie et la panne



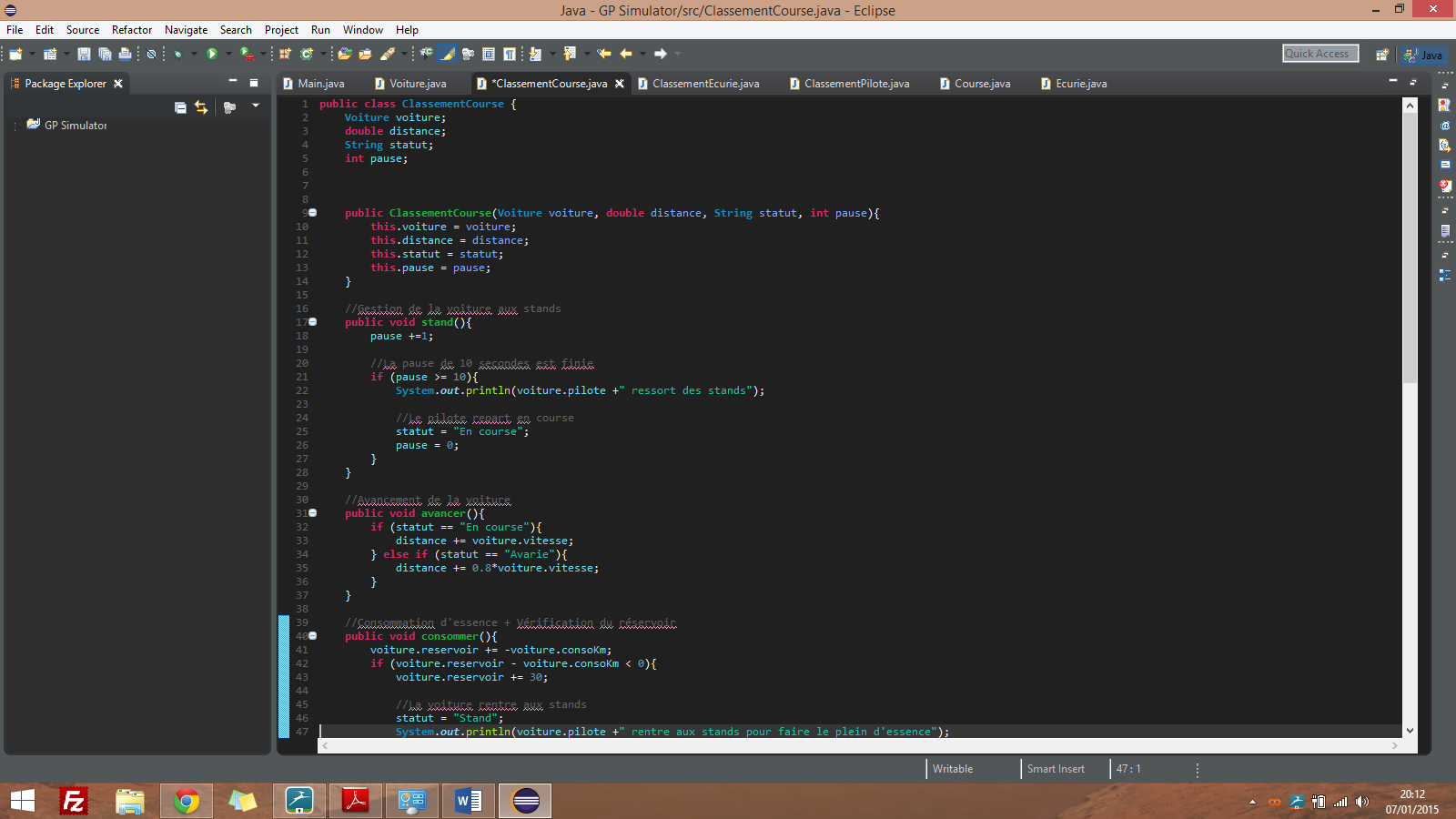
L’avarie et la panne se simulent par un tirage au hasard d’une valeur qui est ensuite comparée à la fiabilité de la voiture. Si cette valeur est inférieure, en fonction du test correspondant, la voiture subit une panne ou une avarie.

### Verification du réservoir



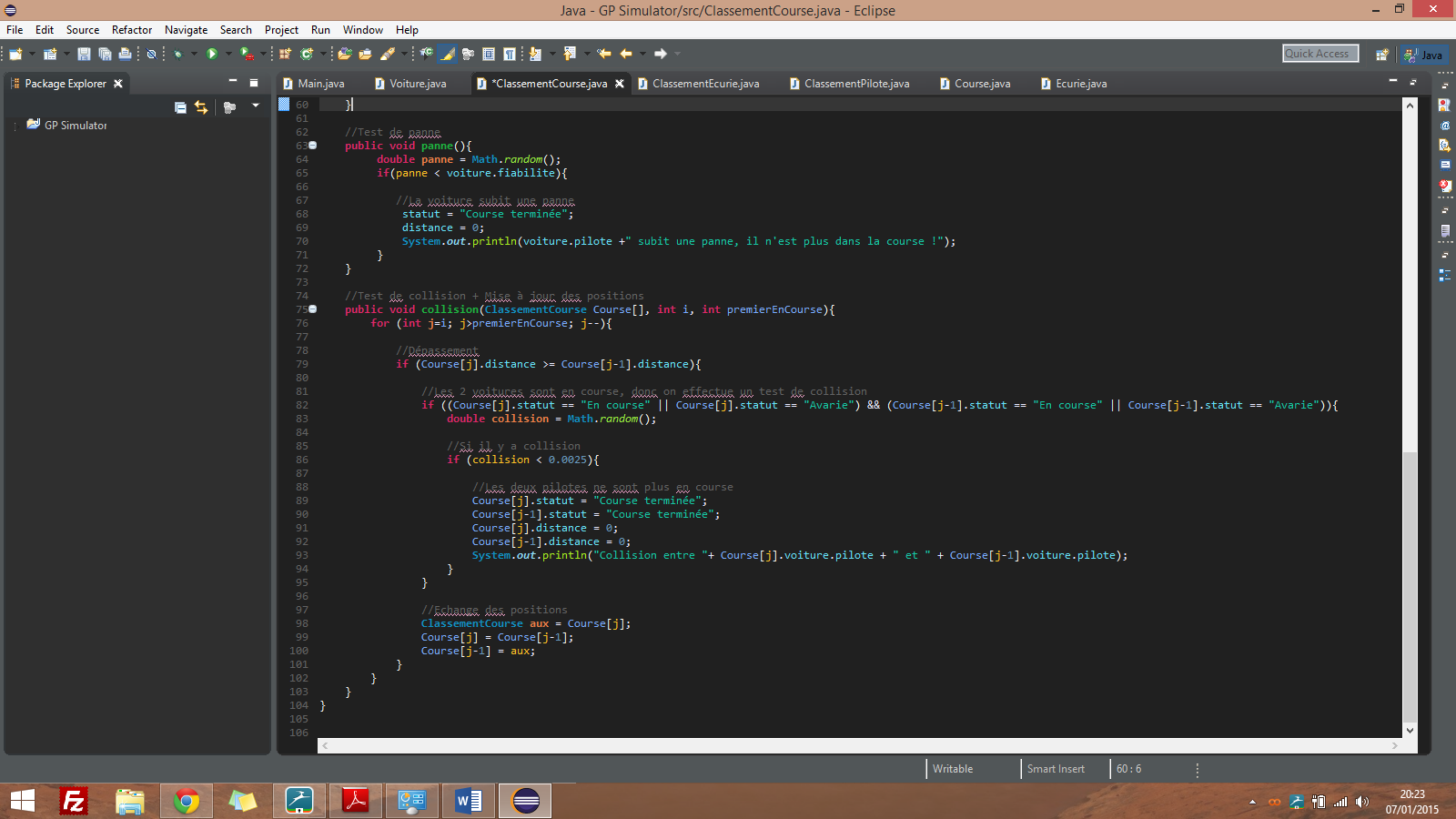
La vérification du réservoir se fait après chaque tour. On vérifie donc si le réservoir est encore suffisant pour effectuer un tour. Si c’est le cas, la voiture peut continuer le tour qu’elle vient d’entamer, sinon le véhicule rentre dans les stands et remplit son réservoir.

### Arrêt aux stands



L’arrêt aux stands est simulée par une pause (un compteur qu’on incrémente) qui correspond à 10 secondes. Dès que cette pause a dépassé ces 10 secondes, la voiture reprend la course.

### Dépassement et collision



Si la voiture a parcouru une distance supérieure ou égale à la voiture devant lui au classement, alors il y a dépassement. Lorsqu’il y a dépassement on effectue un test de collision, collision est une variable à laquelle on assigne un nombre aléatoire compris entre 0 et 1. Si ce nombre aléatoire est inférieur à 0,0025 (ce qui équivaut à 0,25% de chances qu’il y ai une collision) alors on a collision, les statuts des deux voitures sont alors mis à jour et la course est terminée pour eux. On échange ensuite les positions des deux voitures. On n’effectue pas de test de collision lorsqu’une voiture dépasse une voiture aux stands.

# Conclusion

Le projet Java s’est déroulé plutôt correctement , nous avons eu quelques difficultés que nous sommes parvenus à résoudre. Au niveau des données, par rapport au cahier des charges, le projet est cohérent, seule l’unité de la vitesse a été modifiée, car nous utilisons une vitesse est tour/seconde pour pouvoir effectuer les pauses plus simplement. Nous nous sommes d’abord concentré sur la simulation d’une course, avant de l’effectuer sur toute une saison. De ce fait, nous avons utilisé une boucle while pour simuler simplement la course.

Au final, nous avons fait le choix de ne pas utiliser de timer, car pas forcément nécessaire, et compliquant la simulation. Cependant, il est quand même possible de rendre un temps par rapport à notre implémentation.

Nous pourrions améliorer la « simulation » en variant la taille des circuits, en ajoutant des contraintes comme par exemple un système météorologique augmentant plus ou moins les risques d’accident. Nous pourrions ajouter d’autres circuits aussi afin de créer des saisons aléatoires en piochant dans une liste de circuits.