TD Concepts objet et Java

2016-17

Exercice 1

Vous allez développer une gestion simple d'autotamponneuses. Celles-ci sont caractérisées par :
— le fait qu'elles soient placées ou non sur une piste d'autotamponneuses, leur position étant alors représentée par une paire de valeurs réelles (coordonnées dans un plan à deux dimensions)
— le fait qu'elles soient occupées ou non, et si oui par quelle personne (il faut qu'elles soient placées pour pouvoir être occupées)
— le fait qu'elle soient allumées ou non (il faut qu'elles soient occupées pour pouvoir être allumées)
— le fait qu'elle soient clignotantes ou non (il faut qu'elles soient allumées pour pouvoir être clignotantes)

1. Créez une nouvelle classe `Autotamponneuse` dans un *package* `com.feteforraine`.

2. Ajoutez les données nécessaires à la classe en choisissant les niveaux de *visibilité* appropriés.

3. Ajoutez deux constructeurs à la classe : l'un sans paramètre, l'autre avec une coordonnée x et une coordonnée y. Faites en sorte de réutiliser ce qui peut l'être dans vos constructeurs.

4. Ajoutez des *méthodes d'instance* dans la classe `Autotamponneuse` pour **consulter** l'état d'une autotamponneuse particulière : `estOccupee`, `getNomOccupant`, `estAllumee`, `estClignotante`.

5. Ajoutez une méthode `main` dans la classe `Autotamponneuse` afin de pouvoir définir un programme principal de *test*. Testez-y sur des exemples le code précedemment défini, et utilisez-la pour tester le code des questions suivantes.

6. Ajoutez le code nécessaire pour qu'une autotamponneuse ait désormais un *identifiant* entier unique. Cette valeur ne devra pas être choisie par l'utilisateur, et devra commencer à la valeur 1 pour la première autotamponneuse.

7. Redéfinissez dans la classe `Autotamponneuse` la méthode particulière suivante de la classe `java.lang.Object` :
   `String toString()`
   qui retourne une représentation sous forme de chaîne de caractères de l'état d'un objet. Des exemples de chaînes pourraient être :
   ```
   [1] (5.0,5.0) libre éteinte non clignotante
   [2] (7.0,2.0) occupée (Charles Darwin) éteinte non clignotante
   [2] (5.1,5.0) occupée (Charles Darwin) allumée clignotante
   ```

8. Ajoutez à présent des *méthodes d'instance* dans la classe `Autotamponneuse` pour **modifier** l'état d'une autotamponneuse particulière, en respectant les règles énoncées ci-dessus : `place`, `ajouteOccupant`, `enleveOccupant`, `allume`, `eteint`, `demarreClignotement`, `arreteClignotement`. Afin d'informer le programme appelant sur la modification effective ou non de l'objet correspondant, faites retourner à vos méthodes une valeur booléenne. On décide que ces méthodes ne pourront être accessibles qu'aux autres classes *du même package* (`com.feteforraine`).

9. On souhaite détecter des collisions entre autotamponneuses. Pour simplifier, on dira que deux autotamponneuses sont en collision si la distance entre leur position (comprise ici comme un point dans le plan) est inférieure à une valeur constante définie au niveau de la classe. Ajoutez cette valeur de la manière appropriée, puis ajoutez une méthode d'instance pour le calcul de distance entre deux autotamponneuses :
   ```
   double calculeDistance(Autotamponneuse autreAuto)
   ```
   ainsi qu'une méthode d'instance indiquant si une collision a lieu entre deux autotamponneuses :
   ```
   boolean collision(Autotamponneuse autreAuto)
   ```

10. Afin d'autoriser d'autres notations, proposez des équivalents sous forme de *méthodes de classes* pour les deux méthodes d'instance précédentes :
    ```
    static double calculeDistance(Autotamponneuse auto1, Autotamponneuse auto2)
    ```
    et :
    ```
    static boolean collision(Autotamponneuse auto1, Autotamponneuse auto2)
    ```
    Attention dans ce cas à bien prendre en compte le fait que les paramètres peuvent tous les deux avoir une valeur `null` à l'exécution.

11. Redéfinissez dans la classe `Autotamponneuse` la méthode de la classe `java.lang.Object` :
    ```
    boolean equals(Object autreObjet)
    ```
    qui indique si deux objets sont égaux (même état complet ici). Attention, la méthode étant déclarée au niveau de la classe la classe `java.lang.Object`, il faudra donc vérifier puis transtyper le type du paramètre de la méthode.

12. Ajoutez une nouvelle classe `PisteAutotamponneuses` dans le package `com.feteforraine`. Une telle classe comporte une collection d'autotamponneuses représentée par un tableau. Ajoutez-y un constructeur prenant pour paramètre la taille de cette collection.

13. Ajoutez une nouvelle méthode de classe `main` dans la classe `PisteAutotamponneuses` pour définir un programme de test qui crée le nombre d'autotamponneuses requis et les place de façon aléatoire sur la piste (cf. `Math.random()`). Attention, faites en sorte qu'aucune autotamponneuse nouvellement placée ne soit en collision avec une autre autotamponneuse.

14. Redéfinissez la méthode `String toString()` de la classe `PisteAutotamponneuses` afin qu'elle affiche l'état de la collection complète (l'ordre des autotamponneuses sera par ordre d'ajout).

15. Ajoutez une méthode `dereglementAleatoire` à la classe `PisteAutotamponneuses` qui déplace de façon aléatoire et continue chaque autotamponneuse encore pilotée tour à tour, et élimine les autotamponneuses entrées en collision. Cette méthode affichera l'historique des autotamponneuses éliminées, et le vainqueur.