Java Cours 12

Explication TP9

**Diapo 3**

On veut des chenilles et visage sur le même dessin, donc il faut modifier la classe dessin pour qu’elle puisse afficher des objets plus généraux.

Pour se faire il faut une interface IDessinable, qui reprend la méthode dessiner.

**Diapo 5**

Refactoring de notre code pour avoir l’interface IDessinable. Chenille implements IDessinable et on @override la méthode dessiner. Même chose pour VisageRond

**Diapo 8**

On sépare les méthodes dessiner et déplacer dans deux interfaces différentes IAnimable hérite de IDessinable. Ainsi une personne qui dessine des objets fixe, il n’aura pas à implémenter une méthode déplacement.

**Diapo 9**

Dans la fonction animer, il faut vérifier si l’objet que l’on dessine c’est un IAnimable ou juste un IDessinable. Ce fait avec instanceof et on fait un downcasting.

**Diapo 10**

Nouvel objet à faire dessiner, les étoiles.

g.create(), cela créé une copie de l’objet graphics g. g c’est comme un pinceau, on peut changer son épaisseur de trait, son type de trait … donc comme ça on a pas à redéfinir g2 à chaque fois, on travaille juste sur g2. Si on ne fait pas cette copie, on va modifier l’objet graphics pour un objet et quand on créera un autre objet, il aura les même caractéristiques que le nouveau g, on a donc un effet de bord.

G2.translate(x, y) c’est pour placer l’étoile à l’endroit désiré sur la fenêtre avant qu’elle soit dessinée.

Les calculs pour les sommets et le chemin reliant les points, on le fait dans le constructeur, comme ça c’est calculé au moment de la création et pas à chaque fois que l’on redessine.

Remarque : float 2.0f, le petit « f » c’est pour indiquer que c’est un float, si on ne le met pas le compilateur considère comme un double. Float plus petit que double donc peut poser des problèmes dans certain cas.

**Diapo 11**

Nouvelle forme, polygone régulier. La différence avec l’étoile, c’est la construction du chemin qui change et le nombre de sommets. Donc ces parties seront dans les classes respectives et le reste sera dans leur classe abstraite mère.

**Diapo 14**

Insertion de nouveaux objets, on fait une succession d’héritage de classe abstraite pour faire une spécialisation des formes.

**Diapo 16**  
Ce que la classe forme va gérer. Pour dessiner ce sera une méthode abstraite qui sera redéfinie dans les classes filles.

**Diapo 19**

Animation des formes, différents type d’animation ou sans animation.

**Diapo 20**

Cette première solution ne factorise pas le code, car une étoile ou un polygone qui rebondissent, on aura du code similaire dans chacune des sous classes.

2ième inconvénient, si on veut changer le type de déplacement d’un objet en cours de déplacement, cela va ajouter beaucoup de code et avec beaucoup de redondance.

Si on est obligé d’avoir du code redondant, c’est qu’il y a une meilleure façon de modifier notre code

**Diapo 22**

2ème solution, si implémentation d’IAnimable dans Forme, alors il va falloir ajouter bcp de paramètres à la classe forme

**Diapo 24**

Il faut exploiter la délégation. Nouvelle classe, FormeAnimée qui implements IAnimable.

L’interface IAnimateur, le déplacement est délégué à l’objet Animateur.

**Diapo 27**

Attribut dessin c’est pour savoir quand il va taper sur les bords. La méthode de déplacement des visages ou de la tête de la chenille marche pour les formes inscrites dans un cercle. Pour une forme quelconque il faut utiliser un carré ou rectangle englobant. A ce moment-là on va évaluer les angles de cette boite vis-à-vis des bords du dessin.

L’animateur récupère le rectangle englobant et teste si sort du dessin (droite, gauche, haut, bas)

Le code dans AnimateurRebond qui peut servir pour d’autre type de déplacement avec rebond. Ainsi on crée une classe abstraite AnimateurAvecDessin dont AnimateurRebond va hériter. Toutes les classes qui auront un autre type de déplacement avec rebond hériteront de cette classe.

**Diapo 32**

Solution 1 : un inconvénient, à chaque fois que l’on crée un objet, on aura le rectangle englobant qui sera recalculé (on crée l’objet le rectangle englobant à chaque fois)

Solution 2 : rectEnglobant comme attribut. On ne peut pas mettre rectEnglobant = calculRectEnglobant() dans le constructeur. Le chainage des constructeurs fait que l’on risque d’avoir un rectangle englobant de largeur et hauteur nulles car r n’aura pas encore été initialisé. L’initialisation se fait lors du premier appel de getRectEnglobant.

**Diapo 39-41**

On passe forme en interface et donc on peut avoir un tableau avec des IForme et des IAnimable sans problème. API c’est la partie interface. C’est les méthodes de base qui permettent de faire facilement l’implémentation (partie bleu)