INFO-F-202
Langages de programmation 2
Université libre de Bruxelles
Laboratoire 8

Justin Dallant John Iacono
Yannick Molinghen Alexis Reynouard

1 But

Héritage.

2 Sommaire

Faites la même chose que dans Lab 6 mais au lieu d'utiliser des modèles, utilisez l'héritage.

3 Details

Dans le laboratoire 6, nous avons utilisé des modèles. Dans la solution, nous avions

```
template <typename Sketchable>
class Spin
....

template <class Sketchable>
class Bounce {
....

template <typename Sketchable,typename Animation>
class ClickableCell {
....
```

Nous avons utilisé Sketchable pour représenter un Circle ou un Rectangle, et nous avons utilisé Animation pour représenter un Bounce ou une Spin.

En conséquence, la classe Canvas finale ressemblait à :

```
class Canvas {
  vector< ClickableCell< Rectangle, Spin<Rectangle>> > spinners;
  vector< ClickableCell< Rectangle, Bounce<Rectangle> > >
     bouncingRectangles;
  vector< ClickableCell< Circle, Bounce<Circle> > > bouncingCircles;
 public:
  Canvas();
```

```
  void draw();
  void mouseClick(Point mouseLoc);
  void keyPressed(int keyCode);
};


Canvas::Canvas() {
  for (int x = 50; x<500; x+=100)
    spinners.push_back({{{x, 400},50,100}});
  for (int x = 50; x<500; x+=100)
    bouncingRectangles.push_back({{{x, 250},75,75}});
  for (int x = 50; x<500; x+=100)
    bouncingCircles.push_back({{{x, 150},30}});
}

void Canvas::draw() {
  for (auto &c: spinners) {
    c.draw();
  }
  for (auto &c: bouncingRectangles) {
    c.draw();
  }
  for (auto &c: bouncingCircles) {
    c.draw();
  }
}

void Canvas::mouseClick(Point mouseLoc) {
  for (auto &c: spinners)
    c.mouseClick(mouseLoc);
  for (auto &c: bouncingRectangles)
    c.mouseClick(mouseLoc);
  for (auto &c: bouncingCircles)
    c.mouseClick(mouseLoc);
}
```

Ce qui est ennuyeux à ce sujet, c'est que nous devions avoir trois vecteurs différents de ClickableCell, chacun modélisé d'une manière différente. Il n'y avait aucun moyen d'avoir un seul vecteur car tous les éléments d'un vecteur doivent avoir le même type.

Votre objectif aujourd'hui est d'utiliser l'héritage pour que tous les objets puissent être placés dans un seul vecteur comme ci-dessous.

On remarque aussi que nous n'avons plus besoin de savoir quel est le type exacte stocké dans une variable. Avec l'héritage, un pointeur d'un certain type pointe vers un objet de ce type ou un objet qui en hérite.

```
vector< ClickableCell > sketchables;
```

Cela permet de simplifier draw et keyPressed en une seule boucle for :

```
void Canvas::draw() {
  for (auto &c: sketchables) {
    c.draw();
  }
```

```
}

void Canvas::mouseClick(Point mouseLoc) {
  for (auto &c: sketchables)
    c.mouseClick(mouseLoc);
}
```

Comment faire fonctionner cela ? Premièrement :

Créez une nouvelle classe Sketchable et faites en sorte que Rectangle et Circle héritent de Sketchable.

Créez une nouvelle classe Animation et faites en sorte que Spin and Bounce hérite d'Animation.

Que doivent contenir Sketchable et Animation ? Ils devraient avoir des méthodes nécessaires aux fonctions qui les utilisent. Par exemple, lorsque nous avons utilisé un Rectangle ou un Circle, via les modèles, nous avons utilisé le fait que Rectangle et Circle avaient les méthodes draw, contains et getCenter.

On pourrait ainsi définir Sketchable comme suit :

```
class Sketchable {
public:
  virtual void draw()=0;
  virtual bool contains(Point p) const=0;
  virtual Point getCenter() const=0;
  virtual ~Sketchable(){};
};
```

Rappelez-vous que =0 indique qu'il n'y a pas d'implémentations pour ces méthodes dans Sketchable lui-même, et que Sketchable est une classe abstraite et que vous ne pouvez pas créer une instance de Sketchable.

Vous devrez appliquer la même logique pour créer une classe Animation.

Donc, en résumé, vous devez :

- Démarrer du code qui vous est fourni
- Créer des classes Animation et Sketchable
- Supprimer les modèles et utiliser les nouvelles classes Animaiton et Sketchable
- Réfléchir. Ce n'est pas seulement un exercice de recherche et de remplacement. Vous devrez repenser le fonctionnement de certaines choses.

Conseils :

- Auparavant, ClickableCell avait une variable ordinaire sketchable :

  ```
  template <typename Sketchable,typename Animation>
  class ClickableCell {
    Sketchable sketchable;
  ```

  Cependant, comme vous l'avez appris dans la vidéo, les variables ordinaires ne peuvent stocker que quelque chose de ce type, pas un type descendant. Vous devez donc maintenant utiliser un pointeur ici.
- Auparavant, ClickableCell avait un pointeur vers une animation. Lorsqu'on cliquait sur une cellule, cela appelait new Animation. Cela a bien fonctionné car Animation était une

variable de modèle représentant un Spin ou un Rounce. Mais maintenant, Animation est une classe abstraite, vous ne pouvez donc pas écrire new Animation. L'animation doit maintenant être transmise via le constructeur, et elle ne doit pas être créée et détruite par ClickableCell. Cela nécessitera de permettre à l'animation de s'exécuter plus d'une fois.

4 Pour aller plus loin

Une fois que vous avez les bases de travail, une chose que vous pouvez réaliser est que votre classe Animation a une méthod draw. Cela en fait-il également un Sketchable (et par extension, Spin and Bounce) ? Faites en sorte que la classe Animation hérite de Sketchable.

Cela a l'avantage que vous pouvez maintenant animer n'importe quel Sketchable, y compris une autre Animation.

Par exemple, en utilisant le code suivant, j'ai une quatrième rangée de formes qui tournent et rebondissent. C'est donc possible car Spin et Bounce animent les Sketchables, et donc maintenant que Spin et Bounce sont eux-mêmes des Sketchables, les deux animations peuvent être combinées.

```
Canvas::Canvas() {
  for (int x = 50; x<500; x+=100)
  {
    sketchables.emplace_back(
        make_shared<Bounce>(
            make_shared<Circle>(Point{x,400},30)));
    sketchables.emplace_back(
        make_shared<Bounce>(
            make_shared<Rectangle>(Point{x,300},30,30)));
    sketchables.emplace_back(
        make_shared<Spin>(
            make_shared<Rectangle>(Point{x,200},30,30)));
    sketchables.emplace_back(
        make_shared<Bounce>(
            make_shared<Spin>(
                make_shared<Rectangle>(Point{x,100},30,30))));
  }
}
```