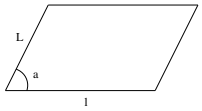
Université de Nice-Sophia Antipolis — POLYTECH
ELEC4 — 2018–2019

Examen de Programmation C++

Durée : 1h
Aucun document autorisé
Mobiles interdits Nom : Prénom :

1 Question 1

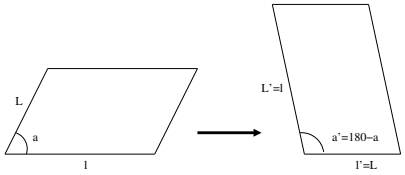
On définit les *parallélogrammes* dont un des cotés est posé sur un axe horizontal par leur longueur horizontale l, leur longueur inclinée L et l'angle inférieur gauche a. La figure ci-dessous montre un exemple.



On veut définir la hiérarchie de classes C++ qui représente les parallélogrammes *généraux*, les rectangles (des parallélogrammes dont l'angle est toujours égal à 90), et les losanges (un parallélogramme ayant deux côtés consécutifs de même longueur).

On veut également effectuer des déformations sur les différents parallélogrammes. On définira les méthodes `allongerX(double coeff)`, `allongerY(double coeff)` et `changerAngle(double coeff)` qui modifie respectivement l, L et a. `allongerX` sera par exemple `l=l*coeff`. **Important :** les déformations ne doivent pas changer la nature du parallélogramme sur lequel elles sont appliquées. Par exemple, un losange devra garder ses cotés égaux, ou rectangle son angle droit quelle que soit la déformation.

On souhaite également définir la méthode `pivoter` qui effectue la transformation donnée ci-dessous :



Enfin, on veut définir la classe `Carré`. Veillez à ce qu'on puisse appeler des méthodes pour déformer ou faire pivoter un carré, néanmoins, tout en le laissant carré.

▶ 1. Écrivez en C++, les classes `Parallelogramme`, `Losange`, `Rectangle` et `Carre` munies de leurs constructeurs et méthodes qui répondent à l'organisation définies précédemment. Veillez à factoriser le code, afin de ne pas répéter inutilement des déclarations, en particulier pour la classe `Carre` (quel type d'héritage ?). Soyez très précis dans le code que vous écrivez.


```
/** Cette classe définit les parallélogrammes dont un des cotés est
 *  posé sur un axe horizontal par leur longueur horizontale l, leur
 *  longueur inclinée L et l'angle inférieur gauche a.
 */
class Parallelogramme {
protected:
  double l, L, a;
public:
  Parallelogramme(double l=0, double L=0, double a=0) {
    assert(l>=0 && L>=0 && a>=0 && a <=180);
    this->l = l;
    this->L = L;
    this->a = a;
  }

  /**
   *  Antécédent : c>0
   *  Rôle : allonge la longueur horizontale d'un coefficient
   *         multiplicateur c
   */
  virtual void allongerX(double c) {
    assert(c>0);
    this->l*=c;
  }

  /**
   * Antécédent : c>0
   * Rôle : allonge la longueur inclinée d'un coefficient
   *        multiplicateur c
   */
  virtual void allongerY(double c) {
    assert(c>0);
    this->L*=c;
  }

  /**
   * Antécédent : 0 <= a <=180
   * Rôle : change l'angle du parallélogramme courant à la valeur a
   */
  virtual void changerAngle(double a) {
    assert(a>=0 && a <=180);
    this->a=a;
  }

  /**
   *  Rôle : pivote le parallélogramme courant vers la droite
   */
  void pivoter() {
    // échanger la longueur et la largeur
    double aux = this->l;
    this->l = this->L;
    this->L = aux;
    // prendre l'angle opposé
    this->a =  this->a-180;
  }
};
```

```
/**
 *  Cette classe définit des Losange par héritage de Parallélogramme
 *  Elle garantit que les cotés restent égaux par redéfinition des méthodes
 *  allongerX et allongerY
 */
class Losange : public virtual Parallelogramme {
public:
  Losange(double c=0, double a=0) : Parallelogramme(c, c, a) {}
  /**
     *  Antécédent : c>0
     *  Rôle : allonge la longueur horizontale ET
     *  inclinée d'un coefficient multiplicateur c, et donc assure que
     *  les longueurs des 2 cotés du Losange courant restent égales
     */

  void allongerX(double c) override {
    Parallelogramme::allongerX(c);
    Parallelogramme::allongerY(c);
  }

  /**
   *  Antécédent : c>0
   *  Rôle : allonge la longueur horizontale ET
   *  inclinée d'un coefficient multiplicateur c, et donc assure que
   *  les longueurs des 2 cotés du Losange courant restent égales
   */
  void allongerY(double c) override {
    this->allongerX(c);
    }
};

/**
 *  Cette classe définit des Rectangle par héritage de Parallélogramme
 *  Elle garantit que l'angle reste à 90 par redéfinition de la méthode
 *  changerAngle.
 */
class Rectangle :public virtual Parallelogramme {
public:
  Rectangle(double l=0, double L=0) : Parallelogramme(l, L,90) {}
  void changerAngle(double c) override { assert(c==90); }
};

/**
 * Cette classe définit les Carré par héritage multiple de
 * Losange et  Rectangle.
 */
class Carre: public Losange, public Rectangle {
public:
  Carre(double c) : Parallelogramme(c,c,90) {}
};
```


2 Question 2

► 2. Expliquez de façon claire et synthétique ce qu'on appelle un patron de conception en POO.

Voir support de cours

.....

► 3. Présentez un des patrons de conception vus en cours, celui que vous voulez. Vous expliquerez son rôle et vous donnerez son diagramme de classes UML.

Voir support de cours

.....