Licence Informatique - Université Montpellier 2

Programmation par objets 2 - GLIN 603 - 25 juin 2014

2h - Tous documents autorisés

1 Héritage multiple

Soit l'extrait de programme suivant.

```
class Animal{
public:
static int nbAnimal;
string nom;
Animal(){cout<<" Animal ";}
virtual ~Animal(){cout<<" ~Animal \n";}
};

int Animal::nbAnimal=0;

class Ovipare : public virtual Animal{
public:
Ovipare(){cout<<" Ovipare ";}
virtual ~Ovipare(){cout<<" ~Ovipare ";}
};

class Vivipare : public virtual Animal{
public:
Vivipare(){cout<<" Vivipare ";}
virtual ~Vivipare(){cout<<" ~Vivipare ";}
};

class Ornithorynque : public virtual Ovipare, public virtual Vivipare{
public:
Ornithorynque(){cout<<" Ornithorynque ";}
virtual ~Ornithorynque(){cout<<" ~Ornithorynque ";}
};

main(int i){  Ornithorynque o;  }
```

Question 1 *Donnez le schéma mémoire de l'instance* `o`*. Justifiez le schéma proposé.*

On complète à présent par les classes suivantes et on écrit un nouveau programme `main`.

```
class AnimalSangFroid : public virtual Animal{
public:
AnimalSangFroid(){cout<<" AnimalSangFroid ";}
virtual ~AnimalSangFroid(){cout<<" ~AnimalSangFroid ";}
};

class Requin: public virtual AnimalSangFroid{
public:
Requin(){cout<<" Requin ";}
virtual ~Requin(){cout<<" ~Requin ";}
};

class RequinTaupe: public virtual Vivipare, public virtual Requin, public virtual Ovipare{
public:
RequinTaupe(){cout<<" RequinTaupe ";}
virtual ~RequinTaupe(){cout<<" ~RequinTaupe ";}
};
```

```
main(int i){
cout << endl << "-----" << endl;
RequinTaupe r0;
cout << endl << "-----" << endl;
RequinTaupe *r1;
cout << endl << "-----" << endl;
Animal *r2 = new RequinTaupe();
cout << endl << "-----" << endl;
}
```

Question 2 *Faites un schéma de la hiérarchie d'héritage montrant l'ordre de l'héritage et l'ordre local servant à appliquer l'algorithme d'appel des constructeurs / destructeurs.*

Question 3 *Qu'affiche l'exécution du programme ?*

2 Exceptions

Soit les deux classes d'exception suivantes.

```
class RectangleException : public virtual exception
{
public:
RectangleException(){}
virtual ~RectangleException() throw(){}
virtual const char* what() const throw()
{char *str=new char[15];strcpy(str," Exc rectangle ");return str;}
};

class CarreException : public virtual RectangleException
{
public:
CarreException(){}
virtual ~CarreException() throw() {}
virtual const char* what() const throw()
{char *str=new char[25];strcpy(str," Exc carre ");
 strcat(str,this-> RectangleException::what());return str;}
};
```

Soit à présent les classes (très partielles) représentant les rectangles et les carrés.

```
class Rectangle{
private:
     float x, y, largeur, hauteur;
public:
virtual void setLargeur(float l){
  if (l < 0) throw new RectangleException();
  this->largeur = l;
}
virtual void setHauteur(float l){
  if (l < 0) throw new RectangleException();
  this->hauteur = l;
}
};

class Carre : public virtual Rectangle{
public:
virtual void setLargeur(float l){
  if (l < 0) throw new CarreException();
  Rectangle::setLargeur(l); Rectangle::setHauteur(l);
}
};
```

Question 4
Qu'affiche l'exécution du programme suivant ? Justifiez.

```
main(int i)
{     try{
      Rectangle *r = new Carre();
      r->setLargeur(-2);
      }
      catch(CarreException *c){cout << "C " << c->what() << endl;}
      catch(RectangleException *r){cout << "R" << r->what() << endl;}
      catch(exception *e){cout << "E " << e->what() << endl;}
}
```

Question 5
Qu'affiche l'exécution du programme suivant ? Justifiez.

```
main(int i)
{     try{
      Rectangle *r = new Carre();
      r->setLargeur(-2);
      }
      catch(RectangleException *r){cout << "R" << r->what() << endl;}
      catch(exception *e){cout << "E " << e->what() << endl;}
}
```

3 Généricité

Nous proposons de développer à présent quelques éléments d'un logiciel de manipulation de plaques alvéolées pour plantations. Ces plaques sont rectangulaires et composées de n lignes de m alvéoles. Chaque alvéole a une profondeur et un diamètre, dont on peut déduire le volume si on considère que l'alvéole est un cylindre.

FIGURE 1 – Plaque d'alvéoles pour plantations

Question 6 *Ecrivez deux classes représentant respectivement les alvéoles et les alvéoles biodégradables (une donnée supplémentaire des alvéoles biodégradables sera le temps de dégradation en semaines).*

Question 7 *Ecrivez une classe générique* `PlaqueAlveolee` *paramétrée par le type de ses alvéoles. Un type d'alvéole passé en paramètre de généricité doit disposer des méthodes pour connaître le diamètre, la profondeur et le volume. La plaque sera implémentée par un tableau à deux dimensions. Dans cette question, écrivez seulement l'entête de la classe, son attribut et son (ses) constructeur(s).*

Question 8 *Ecrivez dans la classe générique* `PlaqueAlveolee` *une méthode permettant de retourner le volume de terre qui sera nécessaire pour remplir les alvéoles.*

Question 9 *Complétez le main pour créer une* `PlaqueAlveolee` *de 3 lignes et 4 colonnes d'alvéoles biodégradables.*