université PARIS DIDEROT PARIS 7

EXAMEN | Master 1 Informatique – LOA | 13/1/2015 8^{30}–11^{30}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé.
(La copie ou les brouillons du voisin ne sont pas des supports autorisés).
Éteignez impérativement vos mobiles.

1 Problème

Soit les déclarations et définitions (incomplètes) C++ suivantes :

```
1  class Point {
2  private:
3    int x, y;
4  public:
5    Point(int x,int y) : x(x), y(y) {};
6    int getX() { return x; }
7    int getY() { return y; }
8    Point auNord() { return Point(x,y-1); }
9    Point auSud() { return Point(x,y+1); }
10   Point aLEst() { return Point(x+1,y); }
11   Point aLOuest() { return Point(x-1,y); }
12 };
13
14 enum Direction { NORD, SUD, OUEST, EST };
15
16 class Ver {
17 private:
18   int length;
19   char head, body, queue;
20   typedef Point *PtrPoint;
21   PtrPoint *corps;
22 public:
23   Ver(int l,char t,char c,char q,int x,int y);
24   void va(Direction dir);
25   ~Ver();
26 };
27
28 int main() {
29   EspaceDeVie espace(8,5);
30   Ver beurk(5,'X','*','.',6,4);
31   espace.print(beurk);
32   beurk.va(EST);
33   espace.print(beurk);
34   for (int i=0; i<3; i++) {
35     beurk.va(NORD);
36     espace.print(beurk);
37   }
38   for (int i=0; i<3; i++) {
39     beurk.va(OUEST);
40     espace.print(beurk);
41   }
42 }
```

Après complétion du code, on souhaite obtenir le résultat suivant :

```
---------- time 0




  .***X
---------- time 1




   .***X
---------- time 2



       X
    .***
---------- time 3


       X
       *
     .**
```

```
---------- time 4

       X
       *
       *
      .*
---------- time 5

      X*
       *
       *
       .
---------- time 6

     X**
       *
       .

---------- time 7

    X***
       .


----------
```

1. Quelles sont les méthodes définies qui peuvent être qualifiées de `const` ?
2. Implémentez les méthodes de la classe `Ver`.
3. Déclarez la classe `EspaceDeVie`.
4. Implémentez les méthodes de la classe `EspaceDeVie`.
5. On souhaiterait pouvoir faire en sorte que l'espace puisse contenir plusieurs vers, le code doit donc être modifié, fournissez un modèle UML de la nouvelle architecture

2 Exercice (inspiré par Jacques Farré)

```
1  class A {
2  private:
3    int x;
4  public:
5    A (int t = 0)  { x = t;   cout << "A(" << t << ')' << endl; }
6    A (const A& a) { x = a.x; cout << "A(" << a << ')' << endl; }
7    const A& operator= (const A& a) {
8      cout << *this;
9      x = a.x;
10     cout << "␣<==␣" << a << endl;
11     return *this;
12   }
13   friend inline ostream& operator<<(ostream& os, const A& a) { return os << "A{x=" << a.x << '}'; }
14 };
15
16 class B : public A {
17 private:
18   A a;
19   int y;
20 public:
21   B (int t=1, int u=2) : a(t), y(u) { cout << "B(" << t << ',' << u << ')' << endl; }
22   B (const A& t, int u=0) : a(t), y(u) { cout << "B(" << t << ',' << u << ')' << endl; }
23   friend inline ostream& operator<<(ostream& os, const B& b) {
24     return os << "B{a=" << b.a << ",y=" << b.y << '}';
25   }
26 };
27
28 int main() {
29   cout << "A----------------" << endl;
30   A a1;
31   A a2=1;
32   A a3=a2;
33
34   cout << "B----------------" << endl;
35   B b1;
36   B b2(2, 3);
37   B b3(a2, 4);
38   b2 = b3;
39 }
```

1. Quels sont les affichages produits à l'exécution du code précédent ? Justifiez
2. Modifiez le code de sorte que les opérateurs de sorties ne soient plus qualifiés de `friend`.

3 Exercice

Étant donné le code suivant :

```
class Jeu {
private:
  list<Joueur *> joueurs;
protected :
  virtual void initialiserLeJeu()=0;
  virtual bool partieTerminee()=0;
  virtual void proclamerLeVainqueur(Joueur *)=0;
public :
   Joueur *joueurSuivant() {
    static list<Joueur *>::iterator i = joueurs.begin();
    if (i==joueurs.end()) i=joueurs.begin();
    return *i;
  }
  void onJoueUnePartie() {
    Joueur *courant;
    this->nombreDeJoueurs = nombreDeJoueurs;
    initialiserLeJeu ();
    while (!partieTerminee() ) {
      courant = joueurSuivant();
      courant->joue();
    }
    proclamerLeVainqueur(courant) ;
  }
};
```

1. pourquoi les méthodes `onJoueUnePartie` et `joueurSuivant` ne sont-elles pas qualifiées par `virtual` ?
2. pourquoi les méthodes `initialiserLeJeu`, `partieTerminee` et `proclamerLeVainqueur` sont-elles qualifiées par `virtual` ?
3. à quoi sert `=0` dans la déclaration de ces mêmes méthodes (celles de la question précédente) ?
4. que peut-on dire de la classe `Jeu` ?
5. une telle classe correspond au design pattern appelé **template method pattern**. Pourquoi ? Qu'est-ce que cela signifie ?

4 Exercice

ïmplémentez une classe `Triplet` permettant d'exécuter le code suivant :

```
int main() {
  Triplet<> t1(2,3,4);
  cout << t1 << endl; // produit l'affichage [2,3,4]
  Triplet<int,float,string> t2(23,99.55f,"dobeedoo");
  cout << t2 << endl; // produit l'affichage [23,99.55,dobeedoo]
  cout << t2[2] << endl; // produit l'affichage dobeedoo
  t2[1] = 44.33f;
  cout << t2 << endl; // produit l'affichage [23,44.33,dobeedoo]
  return 0;
}
```