ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours de Programmation Orientée Objet, sections MA et PH
Chappelier J.-C.

NOM : Hanon Ymous
(000000)
Place : 0

#0000

Programmation Orientée Objet (SMA/SPH) : Examen final

30 mai 2024

INSTRUCTIONS (à lire attentivement)

IMPORTANT ! Veuillez suivre les instructions suivantes à la lettre sous peine de voir votre examen annulé dans le cas contraire.

1. Vous disposez d'une heure quarante-cinq minutes pour faire cet examen (9h15 – 11h00).
2. Vous devez **écrire à l'encre noire ou bleu foncée**, pas de crayon ni d'autre couleur.

   N'utilisez **pas non plus de stylo effaçable** (perte de l'information à la chaleur).
3. Vous avez droit à toute documentation papier.

   En revanche, vous ne pouvez pas utiliser d'ordinateur personnel, ni de téléphone portable, ni aucun autre matériel électronique.
4. Répondez aux questions directement sur la donnée ; ne joignez aucune feuille supplémentaire ; **seul ce document sera corrigé**.

   Vous disposez, si nécessaire, d'une page blanche supplémentaire en fin de sujet.
5. Lisez attentivement et *complètement* les questions de façon à ne faire que ce qui vous est demandé. Si l'énoncé ne vous paraît pas clair, ou si vous avez un doute, demandez des précisions à l'un(e) des assistant(e)s.
6. L'examen comporte quatre exercices indépendants, qui peuvent être traités dans n'importe quel ordre, mais qui ne rapportent pas la même chose (les points sont indiqués, le total est de 120) :
   1. questions de cours : 14 points ;
   2. correction de code : 34 points ; et
   3. conception : 40 points ;
   4. déroulement de programme : 32 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Les bases [14 points]

1.1 Définition [2 points]

Expliquez clairement et brièvement ce qu'est une classe *virtuelle* et donnez un exemple illustratif simple mais pertinent en code C++ :

1.2 Destruction [1.5 point]

Quel destructeur une classe *abstraite* devrait elle avoir ?

(a) aucun (b) un destructeur de copie (c) virtuel (d) abstrait (e) celui par défaut

Justifiez *brièvement* votre réponse :

Ne pas écrire dans cette zone.

1.3 Appels à la pelle [2.5 points]

Dans la définition d'une méthode `m()` d'une sous-classe `Y`, quelle est la syntaxe pour faire appel à une méthode de la super-classe `X` ?

Explicitez les différents cas concernant la méthode de la classe `X` qui est appelée (p.ex. si c'est `m()` ou une autre méthode, ou suivant les droits d'accès).

Ne pas écrire dans cette zone.

suite au dos ☞

1.4 Taxonomie (ou taxidermie ?) [8 points]

On a défini les classes et prototypé les fonctions suivantes :

```
class Mammifere {
  // ... ici toute la définition de la classe, dont
  virtual void se_deplace() = 0;
};

class Renard : public Mammifere {
  // ... ici toute la définition de la classe, dont celle de
  virtual void se_deplace() override;
};

void f(Renard& c);
void g(Renard* c);
void h(Mammifere& m);
void k(Mammifere* m);
```

Dites si les lignes suivantes compilent et fonctionnent correctement (quand elles compilent). Justifiez *brièvement* votre réponse.

	compile ? (oui/non)	fonctionne ? (oui/non)	justification :
`f(new Renard);`			
`f(new Mammifere);`			
`f(*new Renard);`			
`f(*new Mammifere);`			
`g(new Renard);`			
`g(new Mammifere);`			
`g(*new Renard);`			
`g(*new Mammifere);`			

Ne pas écrire dans cette zone.

	compile ? (oui/non)	fonctionne ? (oui/non)	justification :
`h(new Renard);`			
`h(new Mammifere);`			
`h(*new Renard);`			
`h(*new Mammifere);`			
`k(new Renard);`			
`k(new Mammifere);`			
`k(*new Renard);`			
`k(*new Mammifere);`			

Rappel du code :

```
class Mammifere {
  // ... ici toute la définition de la classe, dont
  virtual void se_deplace() = 0;
};

class Renard : public Mammifere {
  // ... ici toute la définition de la classe, dont celle de
  virtual void se_deplace() override;
};

void f(Renard& c);
void g(Renard* c);
void h(Mammifere& m);
void k(Mammifere* m);
```

suite au dos ☞

Ne pas écrire dans cette zone.

Question 2 – Ça cloche [34 points]

Le programme fourni ci-contre contient plusieurs erreurs, dont certaines sont détectées par le compilateur (messages ci-dessous), d'autres pas et dont une pourrait causer un arrêt prématuré lors de l'exécution du programme.

Indiquez toutes les erreurs, en donnant à chaque fois le numéro de ligne correspondant (ou en l'indiquant directement sur le code ci-contre) et une **explication** de l'erreur. Si cela fait sens, proposez également une correction.
Attention ! On ôtera des points pour toute indication d'une erreur qui n'en est pas une.

Voici les messages d'erreurs du compilateur :

```
debug.cc:17:9: error: conflicting return type specified for
    'virtual int B::f2(std::ostream&) const'
debug.cc:8:18: error:   overriding 'virtual void A::f2(std::ostream&) const'
debug.cc: In function 'int main()':
debug.cc:34:5: error: cannot declare variable 'a' to be of abstract type 'A'
debug.cc:4:7: note:   because the following virtual functions are pure within 'A':
debug.cc:7:18: note: virtual void A::f1() const
debug.cc: In constructor 'B::B()':
debug.cc:13:7: error: no matching function for call to 'A::A()'
debug.cc:13:7: note: candidates are:
debug.cc:6:5: note: A::A(int)
debug.cc:6:5: note:   candidate expects 1 argument, 0 provided
debug.cc:4:7: note: A::A(const A&)
debug.cc:4:7: note:   candidate expects 1 argument, 0 provided
debug.cc:37:8: error: could not convert '* c' from 'A' to 'B'
debug.cc:38:16: error: invalid operands of types 'void' and
    '<unresolved overloaded function type>' to binary 'operator<<'
debug.cc:42:6: error: request for member 'f3' in 'b2', which is of non-class type 'B()'
debug.cc:43:5: error: request for member 'f2' in 'c',
    which is of pointer type 'A*' (maybe you meant to use '->' ?)
debug.cc:14:10: error: 'void B::f3()' is private
debug.cc:44:8: error: within this context
```

Réponses/Justifications :

Ne pas écrire dans cette zone.

```
1   #include <iostream>
2   using namespace std;
3
4   class A {
5     public:
6       A(int x) : x(x) {}
7       virtual void f1() const = 0;
8       virtual void f2(ostream& out) const { out << "A::f2" << endl; }
9     private:
10      int x;
11  };
12
13  class B : public A {
14      void f3() { cout << "B::f3" << endl; }
15    public:
16      void f1() const { cout << "B::f1" <<endl; }
17      int f2(ostream& out) const { out << "B::f2" << endl; return 0; }
18  };
19
20  void operator<<(ostream& out, const B & b) { b.f2(out); }
21
22  class C : public A {
23    public:
24      C() : A(0), y(0) {}
25      void f2(ostream& out) { out << "C::f2" << endl; }
26    private:
27      void f1() const { cout <<"C::f1" <<endl; }
28      int y;
29  };
30
31  void f2(B b) { b.f2(cout); }
32
33  int main() {
34    A a(1);
35    B b;
36    A* c = new C();
37    f2(*c);
38    cout << b << endl;
39    A* a1;
40    a1->f2(cout);
41    B b2();
42    b2.f3();
43    c.f2(cout);
44    b.f3();
45    return 0;
46  }
```

Ne pas écrire dans cette zone.

(place supplémentaire pour répondre au dos ☞)

Ne pas écrire dans cette zone.

Question 3 – Conception OO et programmation [40 points]

Cette question porte sur la conception d'un système de gestion pour un hôtel qui doit permettre de prendre des réservations et de consulter les réservations existantes, en respectant les consignes suivantes.

Données

Les données utilisées par le système correspondent à la description suivante :

- le système de gestion doit contenir une liste des chambres de l'hôtel, ainsi que des réservations existantes ;
- une chambre est caractérisée par un numéro, les lits qu'elle contient, son prix par personne et par jour et le nombre d'occupants (le nombre de personnes qui réservent la chambre) ;

  il existe des lits simples (à une place), doubles (à deux places), et superposés (à nombre d'étages variable, ayant une place par étage) ;
- une réservation est caractérisée par un numéro de réservation, un ensemble de chambres à occuper, une date d'arrivée et une date de départ [1].

Fonctionnalités

Grâce à ces données, le système doit fournir les fonctionnalités suivantes :

- spécifier la liste des chambres lors de l'initialisation de l'hôtel ;
- ajouter et supprimer des réservations ;

  on doit pouvoir créer une nouvelle réservation sans nécessairement spécifier toutes ses caractéristiques, mais chaque réservation aura au moins un numéro unique permettant de l'identifier ;
- étant donné une réservation, mettre à jour toutes ses caractéristiques (sauf le numéro de réservation qui doit rester constant) ;
- calculer le prix total du séjour correspondant à la réservation ;

  le prix total est la somme des prix de toutes les chambres (prix par personne de la chambre fois son nombre d'occupants), le tout multiplié par la durée du séjour (nombre de nuits) ;
- étant donnée une chambre, obtenir une description textuelle de toutes ses caractéristiques ainsi que son nombre maximal d'occupants ;

  étant donné un numéro de chambre, on doit également pouvoir afficher ces informations sur le terminal ; par exemple :

```
Chambre numero 1, 36 CHF par jour (par personne)
  Lit simple
  Lit double
Nb max. personnes : 3
Nb occupants : 2
```

- obtenir une description textuelle de toutes les caractéristiques et du prix total de la réservation ;

  étant donné un numéro de réservation, on doit aussi pouvoir afficher ces informations sur le terminal ; par exemple :

1. Peu importe le format des dates ici, choisissez le vôtre, et, si nécessaire, supposez qu'il existe une fonction `nuits()` qui prend deux dates et retourne le nombre de nuits correspondant.

Ne pas écrire dans cette zone.

```
Reservation numero 1, pour 2 personnes
  Chambre(s) : numero 4,
  Dates : du 3 juillet au 12 juillet (9 nuits)
  Prix total : 648 CHF
```

— finalement, il doit être possible d'afficher sur le terminal la liste de toutes les chambres, avec leurs caractéristiques, et la liste de toutes les réservations avec leurs caractéristiques et prix.

Consignes supplémentaires

La conception de votre programme doit minimiser la duplication de code et, dans chaque partie du programme, abstraire les détails inutiles à cette partie, de manière à minimiser les dépendances entre parties.

Par exemple, calculer le nombre de places total d'une chambre ne demande pas de connaître la nature exacte des lits, mais simplement le nombre de places qu'ils contiennent.

Finalement, il faudrait pouvoir construire la description textuelle de chaque entité composée (chambre, réservation) sans devoir inspecter les détails d'implémentation de leurs composants.

Questions

1. **(29 points) Sans** donner tout le détail du code complet (on ne demande ici qu'une *conception*), écrivez **en C++** les classes, les éventuelles relations d'héritage, les attributs et les méthodes des classes, les droits d'accès et les éventuelles fonctions (externes) que vous utiliseriez pour implémenter un tel programme.

   Précisez les types des attributs et les prototypes des méthodes/fonctions, mais ne donnez pas leur définition (on répète : il s'agit ici de la partie *conception*, pas de l'implémentation ; c.-à-d. les prototypes, pas les définitions des méthodes).

   N'indiquez par contre *pas* les constructeurs, destructeurs, « méthodes set » (modificateurs) et « méthodes get » autres que ceux explicitement évoqués dans la liste de fonctionnalités précédente.
2. **(3 points)** Détaillez la manière dont la mémoire est gérée dans votre programme en spécifiant quelle partie ou classe a la responsabilité d'allouer la mémoire et de la libérer, et dans quelles méthodes ou fonctions vous l'implémenteriez (sans donner le détail du code lui-même. On demande ici toujours une conception).
3. **(2 points)** Détaillez, s'il y a lieu, les précautions à prendre lors de la définition du constructeur de copie et de l'opérateur d'affectation de vos différentes classes.
4. **(6 points)** Implémentez [2] la méthode/fonction du calcul du prix total d'une réservation.

2. C'est-à-dire donnez sa définition en C++.

Ne pas écrire dans cette zone.

Ne pas écrire dans cette zone.

(place supplémentaire pour répondre au dos ☞)

Ne pas écrire dans cette zone.

Ne pas écrire dans cette zone.

suite au dos ☞

Question 4 – Exécution de programme [32 points]

Le programme ci-contre compile et s'exécute sans erreurs. Qu'affiche-t-il ?
Répondez ci-dessous, puis justifiez votre réponse en expliquant les points *importants* (on ne vous demande pas ici de paraphraser le code, mais bien de montrer que vous avez compris ce qui se passe !).

Réponse :

1 :	8 :	14 :
2 :	9 :	15 :
3 :	10 :	16 :
4 :	11 :	17 :
5 :	12 :	18 :
6 :	13 :	19 :
7 :		

Justifications :

Ne pas écrire dans cette zone.

Ne pas écrire dans cette zone.

```
#include <iostream>
using namespace std;

class X {
  public:
    X()                 { cout << "X() " ;       }
    virtual ~X()        { cout << "~X() ";       }
    X(double x) : a(x) { cout << "X(double) ";  }
    void set(double x) { a = x; }
    double get() const { return a; }
    void print() const { cout << get() << " " ; }
  private:
    double a;
};

class Z : virtual public X {
  public:
    Z()                     { cout << "Z() " ; }
    virtual ~Z()            { cout << "~Z() "; }
    void f() const          { cout << "Z::f "; }
    virtual void g() const { cout << "Z::g "; }
};

class Y : virtual public X {
  public:
    Y()          { cout << "Y() " ; }
    virtual ~Y() { cout << "~Y() "; }
};

class D : public Y, public Z {
  public:
    D()             { cout << "D() " ; }
    virtual ~D()    { cout << "~D() "; }
    void f() const { cout << "D::f "; }
    void g() const { cout << "D::g "; }
};

void f1(X  x) { x.set( x.get()  + 1.1); }
void g1(X& x) { x.set( x.get()  + 2.2); }
void h1(X* x) { x->set(x->get() + 3.3); }

X f2(X  x) { x.set(x.get() + 4.4); return x; }
X g2(X& x) { x.set(x.get() + 5.5); return x; }
X h2(X* x) {
  X x1 = *x;
  x->set( x->get() + 6.6 );
  return x1;
}
```

```
void pinc(int& u) {
  cout << endl << u << ": ";
  ++u;
}

int main() {
  int i(1);
                    pinc(i); //  1
  Y c;
                    pinc(i); //  2
  Z* b = new D;
                    pinc(i); //  3
  b->g();
                    pinc(i); //  4
  b->f();

                    pinc(i); //  5
  X a = X(0);
                    pinc(i); //  6
  f1(a);
                    pinc(i); //  7
  a.print();
                    pinc(i); //  8
  g1(a);
                    pinc(i); //  9
  a.print();
                    pinc(i); // 10
  h1(&a);
                    pinc(i); // 11
  a.print();
                    pinc(i); // 12
  a = f2(a);
                    pinc(i); // 13
  a.print();
                    pinc(i); // 14
  a = g2(a);
                    pinc(i); // 15
  a.print();
                    pinc(i); // 16
  a = h2(&a);
                    pinc(i); // 17
  a.print();

                    pinc(i); // 18
  delete b;
                    pinc(i); // 19
  return 0;
}
```

Place supplémentaire pour répondre à n'importe quelle question si nécessaire. Mais VEUILLEZ INDIQUER LE NUMÉRO DE LA QUESTION TRAITÉE.

Ne pas écrire dans cette zone.