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

6 juillet 2021

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 de trois heures pour faire cet examen (16h15 – 19h15).
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 des assistants.
6. L'examen comporte cinq exercices indépendants (sur 20 pages), qui peuvent être traités dans n'importe quel ordre, mais qui ne rapportent pas la même chose :
   — questions « courtes » : 29 points ;
   — aide à la programmation : 35 points ;
   — conception : 28 points ;
   — trouver les erreurs : 24 points ; et
   — déroulement de programmes : 14 points.

   Le total est de 130 points ; tous les exercices comptent pour la note finale.

Question 1 – Plus ou moins courtes questions de cours [29 points]

1.1 Échauffement [1.5 points]

Que se passe-t-il si on écrit le code suivant dans la déclaration d'une classe :

```
virtual double f(int) const = 0;
```

a) Le code n'est pas correct.
b) Il n'est pas possible de redéfinir la méthode `f()` dans les sous-classes.
c) On s'attend à ce que cette classe ait au moins une sous-classe.
d) Il n'est pas possible de créer des objets de cette classe.

Entourez la ou les réponse(s) correcte(s). Pénalité en cas de sélection d'une mauvaise réponse.

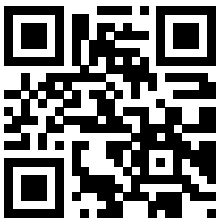
1.2 Pas très solide [3 points]

Le programmeur d'une classe `Solide` a défini l'attribut `masse` comme ci-contre. Un utilisateur de cette classe peut donc accidentellement affecter une masse négative ; ce qui n'a pas de sens.
Expliquez comment améliorer cette classe pour éviter ce problème. Quelle(s) modification(s) suggérez-vous (explication sous forme de texte, pas de code) ?

```
class Solide {
  // ...(autres choses)...
public:
  double masse;
};
```

Réponse :

(ne pas écrire dans cette zone)



1.3 B ? A ? Bah... [9 points]

Considérant les déclarations suivantes :

```
class A {
public:
          int f1() const { return 1    ; }
  virtual int f2() const { return 2    ; }
  virtual int f3() const { return f1() ; }
          int f4() const { return f2() ; }
};

class B : public A {
public:
  int f1() const          { return 101  ; }
  int f2() const override { return 202  ; }
  int f3() const override { return f2() ; } //!\ f2
  int f4() const          { return f1() ; }
};

B babe;
A aha;
A* ptr1(&babe);
B* ptr2(&babe);
A* ptr3(&aha);
```

quelle est la valeur des expressions suivantes ? Indiquez « *erreur* » si l'expression ne compile pas ou est erronée.
Dans tous les cas, justifiez brièvement votre réponse.

a) `ptr1->f1()` :

b) `ptr3->f1()` :

c) `ptr1->f2()` :

d) `ptr1->f3()` :

e) `ptr2->f3()` :

f) `ptr3->f3()` :

suite au dos ☞

1.4 Constructions [4 points]

Considérant les *extraits* de code à droite :

```
class A {
public:
  A() = default;
  A(double);
};

double f1(A);
double f2(A const&);
// ...
A un_a;
double x;
// ...
x = f1(un_a);
// ...
x = f2(un_a);
```

① Quel constructeur est utilisé lors de l'appel `f1(un_a)` ?

a) `A::A()`

b) `A::A(double)`

c) `A::A(A const&)`

d) Aucun.

e) La question n'a pas de sens, un tel appel ne peut pas compiler.

Justifiez *brièvement* votre réponse :

② Quel constructeur est utilisé lors de l'appel `f2(un_a)` ?

a) `A::A()`

b) `A::A(double)`

c) `A::A(A const&)`

d) Aucun.

e) La question n'a pas de sens, un tel appel ne peut pas compiler.

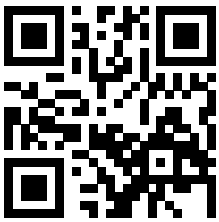
Justifiez *brièvement* votre réponse :

1.5 Vécu [6.5 points]

① **[2.5 points]** Le programme en page ci-contre compile-t-il et s'exécute-t-il correctement ? Si oui, dire ce qu'il affiche, sinon justifiez pourquoi.
Réponse :

② **[4 points]** Si l'on suppose que l'on peut l'exécuter, combien de `double` ce programme crée/utilise-t-il en tout ? Justifiez votre réponse. (Vous pouvez continuer à répondre sur la page suivante.)
Réponse :

(ne pas écrire dans cette zone)



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

class ChampPotentiels {
public:
  ChampPotentiels(size_t taille)
    : contenu(taille)
  {
    for(size_t i(0); i < taille; ++i)
    { contenu[i] = 7.0 * sin(12.0*i + 0.75); }
  }

  size_t size() const { return contenu.size(); }

  virtual void affiche() const
  { cout << size() << " potentiels" << endl; }

private:
  vector<double> contenu;
};

class Ciel : public ChampPotentiels {
public:
  Ciel(ChampPotentiels x)
   : ChampPotentiels(x), autre_chose(x.size())
  {
    for(size_t i(0); i < autre_chose.size(); ++i)
    { autre_chose[i] = 2.236 * sqrt(pow(i, 2.0) + pow(2*i, 2.0)); }
  }

  virtual void affiche() const override
  { cout << autre_chose.size() << " autres choses" << endl; }

private:
  vector<double> autre_chose;
};

int main()
{
  ChampPotentiels a(12);
  Ciel c(a);
  c.affiche();
  return 0;
}
```

suite au dos ☞

(ne pas écrire dans cette zone)

1.6 Qu'en dites vous ? [5 points]

① **[2.5 points]** Le programme ci-contre compile-t-il et s'exécute-t-il correctement ?
Si oui, dire ce qu'il affiche.
Dans tous les cas, justifiez votre réponse.
Réponse :

② **[2.5 points]** Peut-on faire l'appel : `v.g(1.1)` ?
Si oui, dire ce qu'il affiche.
Dans tous les cas, justifiez votre réponse.
Réponse :

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

class A {
public:
  A(double x) : x_(x) {}

  void f(double a) const
  { cout << a + x_ << endl; }

  void g(double a) const
  { cout << x_ * a << endl; }

  virtual void g(const string& s) const
  { cout << s << 50.0 * x_ << endl; }

protected:
  double x_;
};

class B : public A {
public:
  using A::A;

  void g(const string& s) const override
  { cout << s << 2.0 * x_ << endl; }
};

int main() {
  A u(3.0);
  B v(7.0);
  u.g("val1 = ");
  v.g("val2 = ");
  v.f(1.1);
  return 0;
}
```

(ne pas écrire dans cette zone)

Question 2 – Et si vous deveniez assistant(e) ? [35 points]

2.1 You know what ? [8 points]

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

class Pet {
public:
  Pet(const string& name) : name_(name)
  {}
  virtual ~Pet() = default;
  virtual void eat()  = 0;
  virtual void talk() = 0;
private:
  const string name_;
};

class Home {
public:
  Home() : companion(nullptr) {}
  void adopt(Pet* p) { companion = p; }
private:
  Pet* companion;
  Home& operator=(Home const&) = delete;
  Home(Home const&)            = delete;
};

class Dog : public Pet {
public:
  void eat()
  { cout << "Crunch!"    << endl; }
  void talk()
  { cout << "Woof woof!" << endl; }
};

int main()
{
  Home sweet_home;    Dog d("Droopy");
  // Quoi faire ?...


  return 0;
}
```

Dans un exercice qui fournissait les classes `Pet` et `Home` (inchangées), un étudiant est parvenu au code ci-contre, mais il se trouve bloqué dans l'écriture du `main()` car il ne sait pas comment écrire l'adoption du `Dog d` par `sweet_home`. Il vous demande de l'aide en faisant la réflexion suivante :
« Je voudrais mettre `d` dans `sweet_home`, mais il n'y a de la place que pour un `Pet`... Ça ne joue pas ! »

① Qu'avez-vous à lui répondre ?

② Écrivez (dans le code C++ ci-contre) une solution pour que `sweet_home` adopte `d`.

③ Comment appelle-t-on une classe comme la classe `Pet` ?
Pourquoi l'appelle-t-on ainsi ?

④ Le code écrit ci-contre par l'étudiant (hors classes `Pet` et `Home`) ne compile pas. Sauriez-vous dire pourquoi (une seule raison) ?

Réponses :

suite au dos ☞

2.2 Ça ne compile pas ! [14 points]

Un autre étudiant vous interpelle. Il a écrit le code suivant :

```
1  class Vector2D {
2  public:
3    Vector2D(double a = 0.0, double b = 0.0) : x(a), y(b) {}
4
5    bool operator==(const Vector2D& autre) const
6    { return ((x == autre.x) and (y == autre.y)); }
7
8    double norme2() const
9    { return double(x*x + y*y); }
10
11   double norme() const
12   { return double(sqrt(x*x + y*y)); }
13
14   // vecteur unitaire colineaire et de meme sens
15   Vector2D& operator~() const
16   { return Vector2D(x / norme(), y / norme()); }
17
18 private:
19   double x, y;
20 };
```

qui provoque cette erreur de compilation :

```
vecteur_classe.cc: In member function 'Vector2D& Vector2D::operator~() const':
vecteur_classe.cc: error: cannot bind non-const lvalue reference of type
                          'Vector2D&' to an rvalue of type 'Vector2D'
   16 |   { return Vector2D(x / norme(), y / norme()); }
      |            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

① **[1.5 points]** Corrigez (sur le code) l'erreur pour que le code compile.

② **[5.5 points]** Quel(s) autre(s) conseil(s) pourriez-vous donner à cet étudiant ? Essayez d'être exhaustive/exhaustif.

(ne pas écrire dans cette zone)

(place pour encore plus de conseils si nécessaire.)

Plus loin dans son code vous voyez la chose suivante :

```
1  vector<double> vitesse = calcule_vitesse(machin);
2
3  if (sqrt(vitesse[0]*vitesse[0]+vitesse[1]*vitesse[1]+vitesse[2]*vitesse[2])
4      >= vitesse_vent) {
5    faire_quelque_chose(vitesse, machin);
6  } else {
7    faire_autre_chose(vitesse, machin);
8  }
```

③ **[7 points]** Quel(s) autres conseil(s) voudriez-vous lui donner ?
Réécrivez (ensuite, *après* les conseils) le code C++ que vous proposeriez *pour cette partie* (en supposant écrit tout le reste que vous souhaiteriez, mais sans nécessairement l'expliciter ici).

(ne pas écrire dans cette zone)

suite au dos ☞

2.3 C'est bien ? [13 points]

Une partie d'un projet de groupe (fictif ! !) peut se résumer par le code suivant :

```
1  #include <iostream>
2  using namespace std;
3
4  class Dessinable {
5  public:
6    virtual void dessine() const = 0;
7  };
8
9  class Montagne : public Dessinable {
10 public:
11   void dessine() const {
12     cout << "Une montagne" << endl;
13   }
14 };
15
16 class System : public Dessinable {
17 public:
18   System(Montagne* pm) : m(pm) {}
19   ~System() { delete m; }
20   void dessine() const {
21     cout << "Un systeme" << endl;
22   }
23   Montagne* get_montagne() const { return m; }
24
25 private:
26   Montagne* m;
27 };
28
29 int main() {
30   Montagne m;
31   System s(&m);
32   s.dessine();
33   s.get_montagne()->dessine();
34   return 0;
35 }
```

① Ce code compile-t-il et s'exécute-t-il correctement ? Si oui, qu'affiche-t-il ?
Justifiez votre réponse.

② Quel(s) conseil(s) voudriez-vous donner à ce groupe ?
Réécrivez (ensuite, après les conseils) le code C++ que vous leur proposez (uniquement les parties proposées, ne récrivez pas le code inchangé).

(ne pas écrire dans cette zone)

(suite des conseils.)

(ne pas écrire dans cette zone)

Question 3 – Conception [28 points]

On souhaite modéliser informatiquement la vente de billets à des files de clients de spectacles. Pour simplifier, on s'intéresse *uniquement* aux aspects décrits ci-dessous et l'on ignore tout le reste (on ne se préoccupe en particulier ni de l'argent possédé par les personnes, ni du prix des places, ni du nombre de places disponibles).

Dans la file d'attente, il y a des personnes. Chaque personne a le titre du spectacle qu'elle voudrait voir, et le nombre de places qu'elle veut acheter.

En plus des personnes « standard », il y a deux sortes de personnes spécifiques :
- des salariés, qui ont en plus le nom de leur entreprise ;
- des retraités, qui ont en plus leur âge.

Pour simplifier, il n'y a pas de personne retraitée qui soit salariée.

Chaque personne peut « se présenter », d'une façon qui dépend de sa nature (standard, salarié ou retraité) :
- les personnes standard disent simplement :
  « *Bonjour, je voudrais* `[nombre]` *billet(s) pour* `[titre]`. »
- les employés disent :
  « *Bonjour, je voudrais* `[nombre]` *billet(s) pour* `[titre]`.
  Je travaille chez `[entreprise]` ; *faites-vous des réductions ?* »
- et les retraités disent :
  « *Bonjour, je voudrais* `[nombre]` *billet(s) pour* `[titre]`.
  J'ai `[age]` *ans ; ai-je droit à un tarif spécial ?* »

Du coté de la vente/salle de spectacle, il faut pouvoir :
- ajouter une personne à la (fin de la) file d'attente ;
- vendre les billets qu'elle veut à une personne ;
- gérer la file en s'occupant de la première personne dans la file (détails en sous-question ②).

① **[18.5 points] Conception**

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).

Lorsque c'est nécessaire, indiquez également les constructeurs et les destructeurs (sans leur définition).

② **[9.5 points] Programmation**

a) Implémentez (c.-à-d. définissez) la méthode ou la fonction qui s'occupe de la première personne dans la file. Cette méthode ou fonction doit (s'il y a quelqu'un) :

a) la faire « se présenter » (comme expliqué plus haut) ;

b) lui vendre les billets qu'elle désire ;

c) la retirer de la file.

b) Écrivez ensuite la ou les fonction(s) ou méthode(s) qui permettent aux différentes personnes de « se présenter ». Vous pouvez si nécessaire abréger le *texte* effectivement écrit, mais sans négliger/retirer les paramètres/attributs nécessaires.

Réponses :

(suite des réponses à la Question 3)

(ne pas écrire dans cette zone)

Question 4 – Trouver les erreurs [6x4=24 points]

Les six *extraits* de programmes suivants contiennent chacun une erreur. Pour chaque programme, indiquez (sur le programme ou à coté) où se situe l'erreur, expliquez pourquoi c'en est une et comment la corriger.

Attention ! On ôtera 1 point pour toute indication d'une erreur qui n'en est pas une.

```
1  #include <iostream>
2  using namespace std;
3
4  class A {
5  public:
6    A(double x = 1.0) : x_(x) {}
7    double operator*(const A& autre) const {
8      return x_ * autre.x_;
9     }
10 private:
11   double x_;
12 };
13
14 int main() {
15   A a1(4.0);
16   A a2();
17   cout << a1 * a2 << endl;
18   return 0;
19 }
```

Réponse :

```
1  class A {
2  public:
3    A(double x = 0.0) : x_(x) {}
4  protected:
5    double x_;
6  };
7
8  class B : public A {
9  public:
10   B(double x) : A(x) {}
11   void f(A const& autre) {
12     cout << "partie A ";
13     if (abs(autre.x_ -x_) < 1e-10)
14       cout << "identique";
15     else
16       cout << "differente";
17     cout << endl;
18    }
19 };
```

Réponse :

(ne pas écrire dans cette zone)

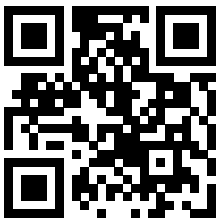
```
1  #include <iostream>
2  using namespace std;
3
4  class A {
5  public:
6    A(double x = 0.0) : x_(x) {}
7    virtual double f(double y) const = 0;
8    virtual double g(double y) const;
9    virtual ~A() = default;
10 protected:
11   double x_;
12 };
13
14 class B : public A {
15 public:
16   B(double x) : A(x) {}
17   double f(double y) const { return g(x_ + y); }
18 };
19
20 int main() {
21   B b(1.0);
22   cout << b.f(2.0) << endl;
23   return 0;
24 }
```

Réponse :

```
1  class A {
2  public:
3    A(double x = 0.0) : x_(x) {}
4    virtual void f() const = 0;
5    virtual void g() { x_ *= 2.0; }
6    virtual ~A() = default;
7  protected:
8    double x_;
9  };
10
11 class B : public A {
12 public:
13   void g() { x_ *= 3.0; }
14 };
15
16 int main() {
17   B u;
18   u.g();
19   return 0;
20 }
```

Réponse :

(ne pas écrire dans cette zone)



```
1  class A {
2  public:
3    A(double x) : x_(x) {}
4  private:
5    double x_;
6  };
7
8  class B : public A {
9  public:
10   B(double y) : y_(y) {}
11 private:
12   double y_;
13 };
```

Réponse :

```
1  #include <iostream>
2  using namespace std;
3
4  class A {
5  public:
6    A(double x): x_(x) {}
7    virtual double f(double y);
8    virtual double g(A const& autre,
9                     double y) const;
10   virtual ~A() = default;
11 protected:
12   double x_;
13 };
14
15 double A::f(double y)
16 { return x_ + y; }
17
18 double A::g(A const& autre, double y) const
19 { return f(autre.f(y)); }
20
21 int main() {
22   A a(1.0);
23   A b(2.0);
24
25   cout << b.g(a, 3.0) << endl;
26
27   return 0;
28 }
```

Réponse :

Indication : pour ce dernier programme, le compilateur indique ceci :

```
prog.cc: In member function 'virtual double A::g(const A&, double) const':
prog.cc:19: error: passing 'const A' as 'this' argument discards qualifiers
   19 |   return f(autre.f(y));
      |                    ^
```

(ne pas écrire dans cette zone)

Question 5 – Différences [14 pts]

Voici, sur ces deux pages, trois programmes qui se ressemblent, mais ont certaines différences indiquées en **gras**.
Pour chacun :

1. dites si vous pensez qu'il compile ou non (**indice :** au moins un des trois compile) ;
2. si vous pensez que oui, dites alors ce qu'il affiche ;
3. dans tous les cas, justifiez votre réponse.

Réponses :

progA.cc :

progB.cc :

progC.cc :

progA.cc

```
1  #include <iostream>
2  using namespace std;
3
4  class Truc {
5  public:
6    void a() const {
7      cout << "hibou" << endl;
8    }
9
10   void b() const {
11     cout << "caillou" << endl;
12   }
13
14   virtual void c() const {
15     cout << "chou" << endl;
16   }
17
18   virtual void d() const {
19     cout << "genou" << endl;
20   }
21 };
22
23 class Blob : public Truc {
24 public:
25   void a() const {
26     cout << "Bal" << endl;
27   }
28
29   void b() const {
30     cout << "Chacal" << endl;
31   }
32
33   void c() const override {
34     cout << "Festival" << endl;
35   }
36
37   void d(double x = 0.0) {
38     cout << "Blob::d(" << x
39          << ")" << endl;      }
40 };
41
42 int main()
43 {
44   Blob x;
45   Truc y(x);
46   y.a();
47   y.b();
48   y.c();
49   y.d();
50   return 0;
51 }
```

(ne pas écrire dans cette zone)

progB.cc

```
1  #include <iostream>
2  using namespace std;
3
4  class Truc {
5  public:
6    void a() const {
7      cout << "hibou" << endl;
8    }
9  protected:
10   void b() const {
11     cout << "caillou" << endl;
12   }
13
14   virtual void c() const {
15     cout << "chou" << endl;
16   }
17 private:
18   virtual void d() const {
19     cout << "genou" << endl;
20   }
21 };
22
23 class Blob : public Truc {
24 public:
25   void a() const {
26     cout << "Bal" << endl;
27   }
28
29   void b() const {
30     cout << "Chacal" << endl;
31   }
32
33   void c() const override {
34     cout << "Festival" << endl;
35   }
36
37   void d(double x = 0.0) {
38     cout << "Blob::d(" << x
39          << ")" << endl;      }
40 };
41
42 int main()
43 {
44   Blob x;
45   Truc y(x);
46   y.a();
47   y.b();
48   y.c();
49   y.d();
50   return 0;
51 }
```

progC.cc

```
1  #include <iostream>
2  using namespace std;
3
4  class Truc {
5  public:
6    void a() const {
7      cout << "hibou" << endl;
8    }
9
10   void b() const {
11     cout << "caillou" << endl;
12   }
13
14   virtual void c() const {
15     cout << "chou" << endl;
16   }
17
18   virtual void d() const {
19     cout << "genou" << endl;
20   }
21 };
22
23 class Blob : public Truc {
24 public:
25   void a() const {
26     cout << "Bal" << endl;
27   }
28
29   void b() const {
30     cout << "Chacal" << endl;
31   }
32
33   void c() const override {
34     cout << "Festival" << endl;
35   }
36
37   void d(double x = 0.0) {
38     cout << "Blob::d(" << x
39          << ")" << endl;      }
40 };
41
42 int main()
43 {
44   Blob x;
45   Truc* y(&x);
46   y->a();
47   y->b();
48   y->c();
49   y->d();
50   return 0;
51 }
```

(ne pas écrire dans cette zone)

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

(ne pas écrire dans cette zone)