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

23 mai 2019

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**.
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, 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 110) :
   1. questions de cours : 14 points ;
   2. programmation : 23 points ;
   3. conception : 32 points ;
   4. déroulement de programme : 22 points et
   5. chercher les erreurs : 19 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Questions de cours [14 points]

① **[4 points]** Qu'affiche le programme suivant ? *Justifier* brièvement (points principaux). **(Répondre sur cette page.)**

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

class A {
public:
    A(string const& s) : a(s) {}
    virtual void operator+=(string const& s) {
        a += s;
    }
    void show() {
        cout << a << endl;
    }
protected:
    string a;
};

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

    void operator+=(string const& s) {
        a = s + a;
        A::operator+=(s);
    }
};

void f(A& x, string const& s) {
    x += s;
}

int main() {
    A a("AA");
    B b("BB");
    f(a, " XX ");
    f(b, " ZZ ");
    a.show();
    b.show();
    return 0;
}
```

② **[4 points]** Qu'affiche le programme suivant ? *Justifier* brièvement (points principaux). **(Répondre sur cette page.)**

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

class A {
public:
    A() {
        cout << "A, ";
    }
    A(A const& a) {
        cout << "AC, ";
    }
    ~A() {
        cout << "AF, ";
    }
};

class B : public A {
public:
    B(int u = 0) : i(u) {
        cout << "B" << i << ", ";
    }
    B(B const& b) : i(b.i) {
        cout << "BC" << i << ", ";
    }
    ~B() {
        cout << "BF, ";
    }
private:
    int i;
};

void f(B b) {}

int main() {
    B b;
    B bb(2);
    f(bb);
    return 0;
}
```

suite au dos ☞

③ **[6 points] (Conception)** Pour une classe `A` ayant un attribut `x`, décrire trois approches différentes permettant d'afficher la valeur de `x` dans de la surcharge de l'opérateur `<<` pour la classe `A`. Critiquer ensuite chacune des trois approches proposées (avantages/inconvénients, lesquelles sont meilleures, ...).

Question 2 Programmation [23 points]

① **[9 points]** Compléter les classes (c.-à-d. ajouter des lignes sans modifier celles existantes) afin que le code ci-dessous *compile* (possiblement avec des *warnings*). On s'intéresse ici uniquement au fait que le code compile (même avec des *warnings*) ; les aspects dynamiques (exécution) ou de conception du programme ne seront critiqués qu'au point suivant (voir sous-question ②).

```
#include <iostream>
#include <algorithm>
#include <iterator>
#include <vector>
using namespace std;

class A {
public:
    A(int x) : a(x) {}
    virtual int f() const = 0;
    int get() const { return a; }

private:
    int a;
};

ostream& operator<<(ostream& flot,
                    A const* a) {
    return flot << a->get();
}

class B : public A {
public:
    B(int x = 0) : A(x) {}
    virtual int f() const override
    { return 5; }
};

class Collection {
private:
    vector<A*> contenu;
public:
    void affiche() const {
      copy(contenu.begin(), contenu.end(),
           ostream_iterator<A*>(cout,
                                ", "));
  }
};

void test1 (Collection& c) {
  B b;
  c.ajoute(&b);
}

void test2 (Collection& c) {
  c.ajoute(new B(12));
}

void test3 (Collection& c) {
  B b(13);
  c.ajoute(b);
}

int main () {
  Collection c;
  test1(c);
  test2(c);
  test3(c);
  c.affiche(); cout << endl;
  return 0;
}
```

Réponse :

suite au dos ☞

Suite de la réponse ① :

② **[14 points]** Critiquez (avantages/inconvénients) les trois fonctions `test1()`, `test2()` et `test3()` dans la perspective de vos compléments précédents. Proposez éventuellement d'autres solutions ici pour corriger ces critiques. En particulier, proposez des implémentations possibles de vos compléments indiqués en ① (ou discutez-les ici, si vous avez déjà donné des implémentations en ①) .

Réponse :

Suite de la réponse ② :

Question 3 Conception OO et programmation (32 points)

On veut créer un programme pour gérer des personnages (informatiques). Pour chaque personnage, on souhaite pouvoir représenter son nom et le fait qu'il représente un humain (p.ex. : Tintin) ou non (p.ex. : Donald Duck).

(**Note :** quelque soit le type de personnages, on pourra avoir plusieurs instances dans le programme, par exemple dans plusieurs sessions. On pourra par exemple avoir plusieurs Schtroumpfettes dans le même programme, lesquelles pourraient avoir des noms différents, p.ex. « schtroumpfette_148 », « schtroumpfette_212 », . . .).

Parmi ces personnages, on souhaite distinguer les personnages de bande dessinée (BD) et les personnages pour enfants. Un personnage de BD est caractérisé par le nom de son dessinateur. Concernant les personnages pour enfants, il est nécessaire de connaître la classe d'âge à laquelle est destiné le personnage (ex : 5–10 ans).

Plus précisément, on s'intéresse aux Schtroumpfs, qui sont des personnages de BD pour des enfants, dessinées par Peyo et visant un public entre 4 et 12 ans. Chaque Schtroumpf possède un habit de couleur rouge ou blanche, et est capable de donner un conseil qui lui est propre (un conseil est une chaîne de caractères). De plus, chaque Schtroumpfs est capable de faire une action `faire_action()` qui lui est propre, p.ex. :

- `faire_action()` des Schtroumpfettes les font chanter ;
- `faire_action()` des Schtroumpfs cuisiniers les font... ...cuisiner.

Les Schtroumpfs étant très sociaux, on souhaite également représenter pour chaque Schtroumpf la liste des (autres) Schtroumpfs avec lesquels il interagit (appelons les ses « amis »). On supposera que si l'on clone un Schtroumpf, les deux clones ont, en tout cas au départ, la même liste d'amis.

Finalement, on désire modéliser des sessions, qui à ce stade représentent simplement une collection de personnages.

① **[24 points] Conception**

Sans donner le détail du code lui-même (on ne demande ici qu'une *conception*), écrivez **en C++** les classes, les 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).

Indiquez également les constructeurs et les destructeurs *lorsque nécessaire* (sans leur définition).

② **[3 points] Conception**

Quel est votre point de vue par rapport à la copie de Schtroumpfs ? Justifiez.

③ **[5 points] Programmation**

Implémentez (c.-à-d. définissez) un constructeur pour les Schtroumpfs, autre que le constructeur par défaut.

Réponses Question 3 :

Suite réponses Question 3 :

Suite réponses Question 3 :

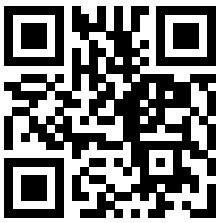
Question 4 – Déroulement de programme [22 points]

Même si le programmeur n'a pas beaucoup suivi les conseils du cours, 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, ni de justifier chaque affichage individuellement, mais bien de montrer que vous avez compris ce qui se passe ! Il n'est pas nécessaire de remplir plus que la place laissée ci-dessous. Vous pouvez, si nécessaire, annoter le code ci-contre).

Réponse (11 points) :

1)
2)
3)
4)
5)
6)
7)
8)
9)
A)
B)

Justifications (11 points) :



```
1   #include<iostream>
2   using namespace std;
3
4   class A {
5   public:
6       A (int x) : a(x) {}
7       int get() const { return a; }
8       virtual void action(int& i) { a += i; i--; }
9
10  private:
11      int a;
12  };
13
14  class B : public A {
15    public:
16    B (int x) : A(x) {}
17    int get() const { return a; }
18    void action(int& i) { a -= i; i++; }
19
20    protected:
21    static int a;
22  };
23
24  class C : public B {
25    public:
26    C (int x) : B(x) {}
27  };
28
29  int B::a = 5;
30
31  int main () {
32    int value(1);
33    A a(15);  B b(21);  C c(10);  A* d(&c);
34    cout << "1) " << a.get() << endl;
35    cout << "2) " << b.get() << endl;
36    cout << "3) " << c.get() << endl;
37    a.action(value);
38    cout << "4) " << a.get() << endl;
39    b.action(value);
40    cout << "5) " << b.get() << endl;
41    c.action(value);
42    cout << "6) " << c.get() << endl;
43    d->action(value);
44    cout << "7) " << a.get()  << endl;
45    cout << "8) " << b.get()  << endl;
46    cout << "9) " << c.get()  << endl;
47    cout << "A) " << d->get() << endl;
48    cout << "B) " << value << endl;
49    return 0;
50  }
```

Question 5 Chercher les erreurs [19 points]

Le programme fourni ci-contre contient 10 erreurs de différente nature (syntaxe, conception, exécution, ...).

① **[18 points]** Marquer les 10 erreurs directement sur le code ci-contre, et les *expliquer brièvement* à droite du code ou ci-dessous.

Si une même erreur a plusieurs conséquences, ne la compter qu'une seule fois (et ne la marquer qu'à un seul endroit, au choix).

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

② **[1 point]** Le compilateur indique une erreur à la ligne 39 :

```
exo.cc:36: error: new types may not be defined in a return type
exo.cc:36: error: two or more data types in declaration of ‘main’
```

De quoi s'agit-il ?

```
1   #include <iostream>
2   using namespace std;
3
4   class etre_vivant : public humain {
5   public:
6     etre_vivant(int age) : age(age) {}
7     virtual void respire() const = 0;
8
9   private:
10    int age;
11  };
12
13  class mammifere : public etre_vivant {
14  public:
15    mammifere(int age, double taille)
16      : etre_vivant(age), taille(taille) {}
17    void allaite() { cout << "glouglou" << endl; }
18
19  private:
20    double taille;
21  };
22
23  class humain : public mammifere {
24  public:
25    humain(int age, double taille)
26      : etre_vivant(age, taille) {}
27    void pense() { cout << "donc je suis !" << endl; }
28
29    double imc(double poids) {
30      double imc;
31      imc = poids / ((taille/100)*(taille/100));
32      return imc;
33    }
34  }
35
36  int main() {
37    mammifere m = new mammifere (5, 35.0);
38    m.respire();
39    humain h(23, 182);
40    mammifere* m2 = &h;
41    m2->pense();
42    etre_vivant h2(12);
43    cout << "imc = " << humain::imc(65) << endl;
44    return 0;
45  }
```

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