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

17 août 2020

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 deux heures pour faire cet examen (9h15 – 11h15).
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, de pages blanches supplémentaires 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, 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 107) :
   1. question courte : 13 points ;
   2. conception : 22 points ;
   3. programmation : 28 points ;
   4. déroulement de programme (1) : 26 points ; et
   5. déroulement de programme (2) : 18 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Gestion de données [13 points]

Considérez la classe suivante :

```
class DataList
{
private:
    vector<Data*> content;
};
```

① **[5 points]** Implémentez une méthode `add()` qui reçoit une référence constante sur un objet `Data`, et en ajoute une copie au contenu de `DataList`. Chaque `DataList` est responsable des `Data` qu'elle contient et ne les partage en aucune façon avec aucune autre partie du code.

Vous pouvez faire toutes les suppositions *raisonnables* dont vous avez besoin sur la classe `Data`, mais veuillez les expliquer brièvement.

② **[8 points]** Implémentez ensuite l'opérateur d'affectation pour la classe `DataList` (qui doit être copiable).

Réponses :

(si nécessaire, suite des réponses à la Question 1)

Question 2 – Conception [22 points]

On souhaite modéliser informatiquement une boîte d'envois postaux, dans laquelle on peut placer différents courriers. Les courriers stockés peuvent être de deux types :

- des lettres, dont le tarif d'affranchissement dépend du caractère d'urgence de chaque lettre : en tarif normal, on affranchit les lettres au coût de 3 francs l'unité et en courrier rapide à 5 francs l'unité ;
- des colis, dont l'affranchissement est de 10 francs par unité plus 2 francs par kilo (poids).

Les lettres et colis ont par ailleurs d'autres spécificités propres non explicitées ici, lesquelles les différencient. Ces spécificités propres sont simplement ignorées dans cet examen.

On souhaite créer une méthode `affranchir()` qui affranchit automatiquement l'ensemble des courriers contenus dans la boîte en fonction de leur type précis.

① **[18 points] Conception**

Sans donner tout le détail du code complet (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 **[4 points] Programmation**

Implémentez (c.-à-d. définissez) la méthode `affranchir()`. Commentez votre solution en insistant sur les principaux aspects et explicitez son fonctionnement.

Suite des réponses à la Question 2 :

Question 3 – Programmation [28 points]

Dans cette question, on vous demande de complèter les parties manquantes d'un programme C++ dont le but est de modéliser différents types de comptes bancaires : comptes de base, compte épargne et compte à prestations facturées (dits « comptes payants ») :

- les comptes de base possèdent un identificateur unique, ainsi qu'un certain montant (solde du compte) ; on peut de plus ajouter ou retirer de l'argent de ces comptes et afficher leurs informations ;
- les comptes épargne sont des comptes de base qui en plus rémunèrent leur argent suivant un certain taux ;
- les comptes payants sont des comptes de base pour lesquels chaque opération (dépôt ou retrait) coûte de l'argent.

Dans le code fourni ci-dessous, certaines portions sont manquantes et on vous demande de les écrire, à l'endroit des commentaires « `// COMPLÉTER ICI ...` » :

- les méthodes `deposer()` et `retirer()` de `Compte` ; les sommes déposées ou retirées doivent être strictement positives (sinon il ne se passe simplement rien) ;
  de plus la méthode `retirer()` doit indiquer en retour si le compte est à découvert ou non ;
- la première ligne de la définition de chacune des classes `CompteEpargne` et `ComptePayant` ;
- le constructeur de chacune des classes `CompteEpargne` et `ComptePayant` ;
  on vous demande d'obliger, lors de la construction, de fournir les valeurs des attributs spécifiques supplémentaires (taux, taxe) ;
- la méthode `affiche()` de chacune des classes `CompteEpargne` et `ComptePayant` ;
  ces méthodes doivent afficher les informations communes à tous les comptes exactement comme la classe `Compte`, puis afficher ensuite toutes les informations supplémentaires des classes concernées ;
- les méthodes `deposer()` et `retirer()` de la classe `ComptePayant`.

Répondez directement sur la donnée.

```
class Compte { // Un compte en banque
private:
  unsigned int const id; // numéro de compte
  double solde_;         // état du compte (solde)

public:
  Compte(unsigned int numero) : id(numero), solde_(0.0) {}

  virtual ~Compte() = default;

  double solde()  const { return solde_; }
  double numero() const { return id;     }

  // Méthode pour tester si le compte est à découvert ou non
  bool decouvert() const { return (solde() < 0.0); }

  // ... suite du code sur la page de droite ...
```

```
  // Méthode pour déposer de l'argent sur le compte
  // COMPLÉTER ICI (méthode deposer())




  // Méthode pour retirer de l'argent du compte
  // COMPLÉTER ICI (méthode retirer())




  // affichage
  virtual void affiche() const {
    cout << "Etat du compte ";
    affiche_nom();
    cout << " #" << numero() << " : " << solde() << endl;
  }

private:
  virtual void affiche_nom() const { cout << "de base"; }
};

// -------------------------------------------------------------
// COMPLÉTER ICI (définition CompteEpargne, 1ère ligne)


private:
  double interet; // taux d'intérêt

public:
  // COMPLÉTER ICI (constructeur)




  virtual ~CompteEpargne() = default;

  double taux() const { return interet; }
  void taux(double t) { interet = t; }

  // ... suite du code au dos ...
```

suite au dos ☞

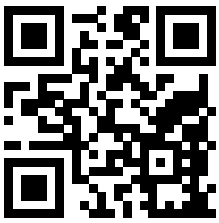
```
  // calcul des intérêts
  double calcul_interets() const { return taux() * solde(); }

  // ajout des intérêts
  void credite_interets() { deposer(calcul_interets()); }

  // affichage
  // COMPLÉTER ICI (méthode affiche())









private:
  virtual void affiche_nom() const override {
    cout << "épargne";
  }
};

// ------------------------------------------------------------
// COMPLÉTER ICI (définition ComptePayant, 1ère ligne)


private:
  double taxe_; // taxe sur les opérations

public:
  // COMPLÉTER ICI (constructeur)









  virtual ~ComptePayant() = default;

  double taxe() const { return taxe_; }
  void taxe(double t) { taxe_ = t; }

  // méthode de dépot d'argent
  // COMPLÉTER ICI (méthode deposer())









  // ... suite du code sur la page de droite ...
```

```
  // Méthode pour retirer de l'argent du compte
  // COMPLÉTER ICI (méthode retirer())









  // affichage
  // COMPLÉTER ICI (méthode affiche())









private:
  virtual void affiche_nom() const override {
    cout << "payant";
  }
};
```

Question 4 – Déroulement de programme [26 points]

On considère le début de programme suivant, qui, même s'il ne suit pas tous les conseils du cours, compile et s'exécute sans erreurs :

```
1   #include <iostream>
2   using namespace std;
3
4   class A {
5   private:
6     int x;
7   protected:
8     int y, z;
9   public:
10    int t;
11
12    A() : x(0), z(0), t(0)
13    { cout << "A()" << endl; }
14    A(bool inutile) : z(5) { y = 7; }
15    virtual ~A() {
16      cout << "bye A" << endl;
17    }
18    int calcul() {
19      int z(42);
20      return 101;
21    }
22  };
23
24  class B : public A {
25    int w;
26  public:
27    B() { cout << "B()" << endl; }
28    B(int u) : A(false), w(3) {
29      cout << "B(" << u << ')' << endl;
30      z = z + u;
31    }
32    ~B() {
33      cout << "bye B" << endl;
34    }
35  };
36
37  class C : public B {
38  protected:
39    int s;
40  public:
41    C() : B(4), s(7) {
42      cout << "C()" << endl;
43    }
44    ~C() {
45      cout << "bye C" << endl;
46    }
47  };
48
49  class D : public A {
50    int r;
51  public:
52    D() : r(2) {
53      cout << "D:D()" << endl;
54    }
55    D(bool p) : A(p) {
56      cout << "D(" << p << ')' << endl;
57    }
58    virtual ~D() {
59      cout << "bye D" << endl;
60    }
61    int calcul() { return r; }
62  };
63
64  class E : public D {
65  public:
66    int q;
67    E(int r) {
68      cout << "E(" << r << ')' << endl;
69      y = calcul();
70      if   (r == 2) { q =  8; }
71      else          { q = 10; }
72    }
73    virtual ~E() {
74      cout << "bye E" << endl;
75    }
76  };
77
78  int main() {
79    A a;
80    B b;
81    C c;
82    D d(true);
83    E e(11);
84
85    // *** ICI
86    return 0;
87  }
```



① **[15 points]** Il vous est demandé d'indiquer, dans le tableau suivant, la valeur des attributs `x`, `y`, `z`, `t`, `w`, `s`, `r` et `q` pour chacun des objets `a`, `b`, `c`, `d` et `e` au moment de l'éxécution de la ligne correspondant au commentaire « `// *** ICI` » (ligne 85).

Si un des attributs n'existe pas pour l'objet en question, indiquez-le par un tiret « — » à la place de la valeur. Si une valeur n'est pas connue (non initialisée), indiquez le par un point d'interrogation « ? ». Si, par contre, *vous*, vous ne connaîssez pas la réponse, laissez en blanc ! **Toute mauvaise valeur sera pénalisée** : −0.25 point par réponse incorrecte contre 0.5 point par réponse non-tiret correcte, 0.15 si c'est un tiret.

		attribut							
		x	y	z	t	w	s	r	q
variable	a								
	b								
	c								
	d								
	e								

② **[11 points]** Dans cette sous-question, on s'intéresse aux droits d'accès pour chacune des classes.

Imaginez pour cela que chaque classe possède une méthode `f()` de la forme

```
void X::f(Y const& autre) {
  m = 33;
  cout << autre.m;
}
```

où *X* représente la classe en question (`A`, `B`, `C`, `D` ou `E`), *Y* une autre classe (possiblement la même : `A`, `B`, `C`, `D` ou `E`), et *m* un attribut possible (`x`, `y`, `z`, `t`, `w`, `s`, `r` ou `q`).

Pour les différents cas possibles considérés dans le tableau ci-contre, indiquez par `oui` ou par `non` dans chacune des cases du tableau, si la ligne de code correspondante est acceptée sans erreur à la compilation.

Par exemple, pour *X*=`A`, *Y*=`A` et *m*=`t`, la méthode `f()` considérée est :

```
void A::f(A const& autre) {
  t = 33;
  cout << autre.t;
}
```

dont les deux lignes compilent et nous avons donc répondu « `oui` » dans les deux cases correspondantes.

X	*Y*	*m*	`m = 33;`	`cout << autre.m;`
A	A	t	oui	oui
		x		
		z		
	B	z		
B	B	x		
		w		
C	A	t		
		x		
		z		
	B	z		
		w		
E	B	z		

En cas de doute, laissez un blanc, car **toute mauvaise valeur sera pénalisée** : −0.5 point par réponse incorrecte contre 0.5 point par réponse correcte.

Question 5 – Déroulement de programme [18 points]

Le programme ci-dessous compile et s'exécute sans erreurs. Indiquez l'affichage exact qu'il produit 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 !)
Vous pouvez répondre à droite du code et, si nécessaire, annoter le code lui-même.

```
1   #include <iostream>
2   #include <vector>
3   using namespace std;
4
5   class Legume {
6   public:
7     Legume() : qualite(7), poids(1.5) {
8       cout << "Encore un légume" << endl;
9     }
10    ~Legume() {
11      cout << "Et un légume de moins" << endl;
12    }
13    virtual void affiche () const {
14      cout << "Qualité       : " << qualite << endl;
15      cout << "Poids (masse) : " << poids   << endl;
16    }
17  protected:
18    int qualite;
19  private:
20    double poids;
21  };
22
23  class Poireau : public Legume {
24  public:
25    Poireau(double h) : hauteur(h) {
26     cout << "C'est un poireau" << endl;
27    }
28    virtual ~Poireau() {
29      cout << "Un poireau à la poubelle" << endl;
30    }
31    virtual void affiche() const {
32      cout << "Hauteur : " << hauteur << endl;
33    }
34  private:
35    double hauteur;
36  };
37
38    // ... suite du code sur la page de droite ...
```

```
38  class Courgette : public Legume {
39  public:
40    Courgette(double L) : longueur(L) {
41     cout <<"C'est une courgette" << endl;
42    }
43    virtual ~Courgette(){
44      cout << "La courgette se ratatine" << endl;
45    }
46    void affiche () const {
47      cout << "Longueur : " << longueur << endl;
48    }
49  private:
50    double longueur;
51  };
52
53  class Navet : public Legume {
54  public:
55    Navet(double r) : rayon(r) {
56     cout << "C'est un navet" << endl;
57    }
58    ~Navet() {
59      cout << "Le navet s'en va" << endl;
60    }
61    virtual void affiche () const {
62      cout << "Rayon : " << rayon << endl;
63    }
64  private:
65    double rayon;
66  };
67
68  int main() {
69    vector<Legume*> panier({
70        new Navet(4.5),
71        new Courgette(12.3),
72        new Poireau(25.6)
73    });
74
75    for (auto x : panier) {
76      x->affiche();
77      cout << "----" << endl;
78    }
79    for (auto x : panier) {
80      delete x;
81    }
82    return 0;
83  }
```

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

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

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