TD: POO de base en C++, Relations entre classes : Composition, agrégation & Héritage

Questions de révision générale :

1) Que veut dire une portée d'un attribut ? donner des exemples de portée et expliquer leurs significations avec des exemples.
2) Que veut dire une relation de composition et une relation d'agrégation ?
3) Comment ces deux relations sont implémentées dans le langage C++ ?
4) Que veut dire héritage ? expliquer sur un exemple ?
5) Quels sont les types d'héritage ? donner des exemples.

Exercice 1 : (POO de base)

Dans le petit code C++ suivant, combien d'objets résideront-ils de façon inaccessible en mémoire, jusqu'à la fin du programme ?

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


class O1 {};
class O2 {
   private: O1 *unO1;
   public: O2() {
         unO1 = new O1();
   }
};
class O3 {
private:
 O2 *unO2;
public:
 O3(){
 unO2 = new O2();
 }
};

int main(int argc, char* argv[]) {
 O3 *unO3 = new O3();
 delete unO3;
 return 0;
}
```

Exercice 2 : (POO de base)

Quel nombre affichera ce programme C++ à l'issue de son exécution ?

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


class O1 {
private:
 static int morts;
public:
 static int getMorts(){
 return morts;
 }
public:
 ~O1(){
 morts++;
 }
};
int O1::morts = 0;
void essai(O1 unO1) {
 O1 unAutreO1;
}
int main(int argc, char* argv[]) {
 O1 unO1;
 for (int i=0; i<5; i++) {
 essai(unO1);
 }
 cout << O1::getMorts() << endl;
 return 0;
}
```

Exercice 3 : (POO de base)

Soit le code suivant:

```
#include <iostream>
using namespace std;
class A{
public:
      A();
      ~A();
};
A::A(){
cout<<"je suis crée"<<endl;
}
A::~A(){
cout<<"je suis détruit"<<endl;
}
int main() {
      A a;
      cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
      return 0;
}
```

1) déduire l'affichage suite à l'exécution de ce programme. Justifier votre réponse.
2) peut-on appeler un constructeur comme on appelle les autres méthodes?
3) que se passe-t-il si on change le code du main pour qu'il soit comme suit:

```
int main() {
   A a;
   a.~A();
   cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
   return 0;
}
```

4) Que se passe dans le programme si on change A() comme suit:

```
A::A(){
   A b;
   cout<<"je suis crée"<<endl;
}
```

5) déduire et justifier l'exécution du code suivant:

```
#include <iostream>
using namespace std;
class A{
      private:
      int x;
public:
      A();
      ~A();
      void M();
};
A::A(){

      cout<<"je suis crée"<<endl;
}
A::~A(){
      cout<<"je suis détruit"<<endl;
}
void A::M(){
      A b;
      b.x=2;
      cout<<"x="<<b.x<<endl;
}
int main() {
      A a;
      a.M();
      cout << "!!!Hello World!!!" <<
endl; // prints !!!Hello World!!!
      return 0;
   }
```

Exercice 4 : (Composition et Agrégation: Exemples en C++) : Rattrapage POO 2018-2019

Soit le code suivant:

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

class A{
    int id;//identifiant de l'objet
    int x;
    int y;
public: A(int id); ~A();
};

A::A(int id){
    this->id=id;
    cout<<"je suis créé de classe A, objet id= "<<id<<endl;
}

A::~A(){
    cout<<"je suis mort de classe A, objet id= "<<this->id<<endl;
}
```

```
   class B{
   int id;//identifiant de l'objet
       A a1, a2;
public: B(int id); ~B();
};

B::B(int id){
      this->id=id;
      cout<<"je suis créé de classe B, objet id= "<<id<<endl;
}

B::~B(){
      cout<<"je suis mort de classe B, objet id= "<<this->id<<endl;
}

int main(){
      B b;
      cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
      return 0;
}
```

- Expliquer l'objectif de ce programme est que penser de son exécution?
- Il y'a une erreur lors de l'exécution de ce programme, c'est laquelle? expliquer et justifier?
- corriger l'erreur.
- Donner les affichage du l'exécution de ce code. Justifier cet affichage.
- Proposer les modifications nécessaires pour rendre la classe B une agrégation plutôt qu'une composition de A.
- Déduire l'affichage. Justifier.
- Proposer une manière pour créer les deux objets a1, a2 et de transmettre leurs références à B. Déduire l'exécution.

Exercice 5 : (relation entre classes)

Considérez les deux classes suivantes : Interrupteur et Lampe, telles que, quand l'interrupteur est allumé, la lampe s'allume aussitôt.

1) Identifier le type de relation entre classes ;
2) Réalisez dans un pseudo-code objet le petit programme permettant cette interaction.

Exercice 6 : (relation entre classes)

Considérez les deux classes suivantes : Souris et Fenêtre, telles que, quand la souris clique sur un point précis de la fenêtre, celle-ci se ferme.

1) Identifier le type de relation entre classes ;
2) Réalisez dans un pseudo-code objet le petit programme permettant cette relation.

Exercice 7 : (Héritage: Exemples en C++)

Soit le code suivant:

```
#include <iostream>
using namespace std;
class A{
public:
	A();
	~A();
};
A::A(){
cout<<"je suis crée"<<endl;
}
A::~A(){
cout<<"je suis détruit"<<endl;
}
int main() {
	A a;
	cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
	return 0;
}
```

1) Déduire l'affichage suite à l'exécution de ce programme. Justifier votre réponse.
2) Peut on appeler un constructeur comme on appelle les autres méthodes?
3) Que se passe-t-il si on change le code du main pour qu'il soit comme suit:

```
int main() {
   A a;
   a.~A();
   cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
   return 0;
}
```

4) Que se passe dans le programme si on change A() comme suit:

```
A::A(){
   A b;
   cout<<"je suis crée"<<endl;
}
```

5) Déduire et justifier l'exécution du code suivant:

```
#include <iostream>
using namespace std;
class A{
	private:
	int x;
public:
	A();
	~A();
	void M();
};
A::A(){

	cout<<"je suis crée"<<endl;
}
A::~A(){
	cout<<"je suis détruit"<<endl;
}
void A::M(){
	A b;
	b.x=2;
	cout<<"x="<<b.x<<endl;
}
int main() {
	A a;
	a.M();
	cout << "!!!Hello World!!!" << endl;
// prints !!!Hello World!!!
	return 0;
   }
```

Exercice 8 : (Portée et héritage : Exemples en C++)

1. soit le code suivant:

```
class A{
 int x;
};
int main(){
 A a;
 a.x=0;
 return 0;
}
```

- Déduire son exécution. Y'a-t-il des erreurs? Comment peut on corriger ces erreurs?

2. Si on reprend le code précédent (sans aucune modification) et on ajoute le code suivant:

```
class B{
     A a;
public: void M();
};
void B::M(){
     a.x=0;
}
```

- Quelle est la relation entre les deux classe A et B?
- Déduire son exécution. Y'a-t-il des erreurs? Comment peut on corriger ces erreurs?

3. Si on reprend le code 1 (sans aucune modification) et on ajoute le code suivant:

```
class B:A{
     int y;
public: void M();
};
void B::M(){
     x=0;
}
```

- Quelle est la relation entre les deux classe A et B?
- Déduire son exécution. Y'a-t-il des erreurs? Comment peut on corriger ces erreurs?