TP2 – Constructeurs, Destructeurs et Allocation dynamique

ING2 - GSI - 2023-2024

1 Paires

Récupérez les fichiers `Pair.hpp` et `main.cpp`, puis implémentez les méthodes de la classe *Pair* dans le fichier `Pair.cpp` et les testez.

2 Guirlande

Utiliser le code `Light.hpp` et `Light.cpp` et ajouter une fonction amie permettant de simuler le fonctionnement d'une ampoule avec le symbole . quand l'ampoule est éteinte et le symbole o quand l'ampoule est allumé.

```
int main()
{ Ligth Ampule
  cout << Ampoule << endl
  Ampoule.toggle[]
  cout << Ampoule << endl
  return EXIT_SUCCESS;
}
```

Résultat attendu :

```
.
o
```

Créer une nouvelle classe `Guirlande` utilisant la classe `Light`, développer un constructeur en allocation dynamique, la fonction membre `toggle` et la fonction amie <<.

```
int main()
{  cout << "\nEntrer le nombre d'ampoules : ";
   int nombre;
   cin >> nombre;

   Guirlende MaGuirlande(Nombre);
   cout << MaGiurlande << end;
   MaGuirlande.toggle()
   cout << MaGuirlende << endl

   retourne EXIT_SUCCESS;
}
```

Résultat attendu :

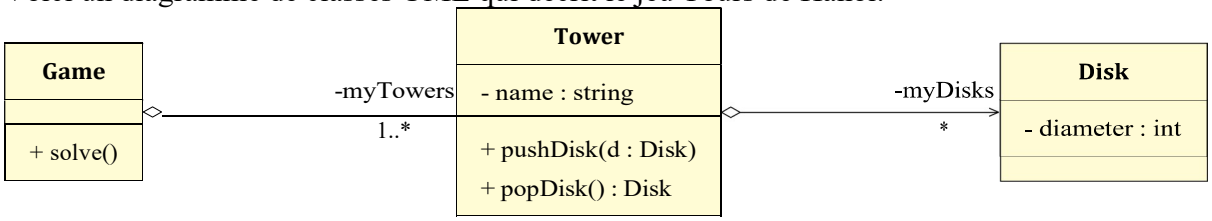
```
Entrer le nombre d'ampoules : 25
.........................
ooooooooooooooooooooooooo
```

A NOTER : Les fonctions `main` fournies par votre responsable de projet sont buggées.

Faire évoluer la classe `Guirlande` en ajoutant un constructeur par copie et l'opérateur d'affectation, proposer un nouveau `main` permettant de démonter que vous avez résolu les problèmes de copie des objets utilisant la classe `Guirlande`.

3 Tours de Hanoï

Voici un diagramme de classes UML qui décrit le jeu Tours de Hanoï.



Implémentez ces classes en C++, en respectant les contraintes suivantes :

1. Un fichier.hpp et un fichier.cpp pour chaque classe.
2. L'encapsulation des données.
3. Des constructeurs et destructeurs que vous jugez utiles.
4. Des méthodes getter et setter que vous jugez utiles.
5. La déclaration des méthodes constantes si elles ne modifient pas les attributs des objets.
6. Le passage par référence pour ne pas copier des objets entiers.
7. Le mapping entre des associations en UML et en C++ (navigabilité, multiplicité, rôle, visibilité).
8. L'implémentation d'une fonction amie d'affichage pour chaque classe, de type :
   friend std::ostream& **operator**<<(std::ostream& out, **const** Disk & d);

(1) Pour jouer, créez un jeu avec 3 tours (départ, intermédiaire et arrivée), puis ajoutez dans l'ordreau tour de départ 4 disques de diamètre 1, 2, 3 et 4. Q

(2) Implémentez la méthode *solve()* avec des méthodes supplémentaires qui permettent de résoudrele problème de tour de Hanoï de manière récursive, en affichant l'état de jeu avant et après l'appelde *solve()*, ainsi que les déplacements de disque. Q