

# Créer une classe Menu pour appeler les exercices demandés qui contient un méthode execute() qui fait l'exemple suivant

Quelle exercice voulez vous faire :

0 -> finir programme

1 -> TP 1.1

2 -> TP 1.2

Saisir votre choix:

Etc.....





# La Programmation C++

**Exercices 4** 





## **TP 4.1.1 – Jeu de carte**

## class Domain Model

### Carte

- \_couleur: Couleur
- \_valeur: std::string
- + Carte(Couleur, std::string&)
- + Carte(Carte&)
- + ~*Carte()*
- + setType(Couleur): void
- + setValeur(std::string&): void
- + afficher(): void {query}
- + equal(Carte&): bool {query}
- + affecter(Carte&): void

«Enumeration» **Couleur** 

PIQUE COEUR CARREAU TREFLE





## **TP 4.1.2 – Jeu de carte**

```
#include <iostream>
#include "Carte.h"
using namespace std;
//enum Couleur{ PIQUE, COEUR, CARREAU, TREFLE};
int main()
    cout << "Jeu de carte" << endl;
   Carte cl(PIQUE, "As");
    cl.afficher();
    Carte c2 (c1);
    c2.afficher():
    c2.setType(TREFLE);
    c2.setValeur("Queen");
    c2.afficher();
    Carte c3(PIQUE, "2");
    c2.affecter(c3);
    c2.afficher();
    c3.afficher();
   if ( c1.equal(c2) ) {
        cout << "is ok :-)" << endl;
    } else {
       cerr << " problem bug" << endl;
       cl.afficher();
       c2.afficher();
    return 0;
```





# TP 4.2.1 – Jeu de carte

#### class Domain Model

#### Carte

- + NbCreation: unsigned
- \_couleur: Couleur
- \_valeur: std::string
- + Carte(Couleur, std::string&)
- + Carte(Carte&)
- + ~Carte()
- + operator=(Carte&): Carte&
- + operator==(Carte&): bool {query}
- + operator!=(Carte&): bool {query}
- + setType(Couleur): void
- + setValeur(std::string&): void

#### «property get»

+ GetNbCreation(): unsigned

#### «friend»

+ operator<<(std::ostream&, Carte&): std::ostream&

### class Domain Model

«Enumeration» **Couleur** 

PIQUE COEUR CARREAU TREFLE





# TP 4.2.2 – Jeu de carte

```
#include <iostream>
#include "Carte.h"
using namespace std;
int main()
    cout << "Jeu de carte" << endl;
    Carte c1 (PIQUE, "As");
    cout << c1 << endl;
    Carte c2 (c1);
    cout << c2 << endl;
    c2.setType(TREFLE);
    c2.setValeur("Queen");
    cout << c2 << endl;
    if ( c1 != c2 ) {
        cout << "is ok :-)" << endl;
    } else {
        cout << " problem bug" << endl;
    return 0;
```





### TP 4.3 - Personne

# Transformer la structure Personne en Classe Personne avec les méthodes qui permettent de :

- créer la classe Personne
- détruire la classe Personne
- initialiser les champs de la classe Personne
- afficher les champs de la classe Personne
- récupérer le nom de la classe Personne
- récupérer le sexe de la classe Personne
- récupérer le numero de la classe Personne
- Test\_Fonction qui test les méthodes de personnes

```
enum Sexe { INCONNUE=0,MASCULIN=1,FEMININ=2};
struct Personne {
  int numero;
  char nom[10];
  Sexe sexe;
};
```

