Exercice 22 : jeu de cartes

> Cet exercice correspond à l'exercice n°65 (pages 164 et 358) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

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

class Couleur {
public:
  typedef enum { ROUGE, VERT, BLEU, BLANC, NOIR } Choix;
  Couleur(Choix c) : valeur(c) {}
  virtual ~Couleur() {}
  Choix valeur;
  void affiche(ostream&, bool feminin = false) const;
};

void Couleur::affiche(ostream& out, bool feminin) const {
  switch (valeur) {
  case ROUGE: out << "rouge"; break;
  case VERT:
    out << "vert";
    if (feminin) out << 'e';
    break;
  case BLEU:
    out << "bleu";
    if (feminin) out << 'e';
    break;
  case BLANC:
    out << "blanc";
    if (feminin) out << "he";
    break;
  case NOIR:
    out << "noir";
    if (feminin) out << 'e';
    break;
  }
}

// ---------------------------------------------------------------------
class Carte {
public:
  Carte(unsigned int cost = 0) : cost(cost) {
    // cout << "  une carte de cout " << cost << " : ";
  }
  virtual ~Carte(){}
  virtual void afficher(ostream& out) const {
    out << "de coût " << cost; }

protected:
  unsigned int cost;
};

ostream& operator<<(ostream& out, const Carte& c) {
  c.afficher(out);
  return out;
}

// ---------------------------------------------------------------------
class Terrain : public virtual Carte {
public:
  Terrain(Couleur c) : couleur(c) {
    cout << "Un nouveau terrain." << endl;
```

```
  }
  virtual ~Terrain() {}
  void afficher(ostream&) const;
protected:
  Couleur couleur;
};

void Terrain::afficher(ostream& out) const {
  out << "Un terrain ";
  couleur.affiche(out);
  out << "." << endl;
}

// ---------------------------------------------------------------------
class Creature : public virtual Carte {
public:
  Creature(unsigned int cost, string nom, unsigned int attaque,
           unsigned int defense) :
    Carte(cost), nom(nom), attaque(attaque), defense(defense) {
    cout << "Une nouvelle créature." << endl;
  }
  virtual ~Creature() {}
  void afficher(ostream&) const;
protected:
  string nom;
  unsigned int attaque;
  unsigned int defense;
};

void Creature::afficher(ostream& out) const {
  out << "Une créature " << nom << ' ' << attaque << "/"
      << defense << ' ';
  Carte::afficher(out);
  out << endl;
}

// ---------------------------------------------------------------------
class Sortilege : public virtual Carte {
public:
  Sortilege(unsigned int cost, string nom, string desc) :
    Carte(cost), nom(nom), description(desc) {
    cout << "Un sortilège de plus." << endl;
  }
  virtual ~Sortilege() {}
  void afficher(ostream&) const;
protected:
  string nom;
  string description;
};

void Sortilege::afficher(ostream& out) const {
  out << "Un sortilège " << nom << ' ';
  Carte::afficher(out);
  out << endl;
}

// ---------------------------------------------------------------------
class CreatureTerrain : public Creature, public Terrain {
public:
  CreatureTerrain(unsigned int cost, string nom, unsigned int attaque,
                  unsigned int defense, Couleur couleur)
    : Carte(cost), Creature(cost, nom, attaque, defense),
      Terrain(couleur)
  {
    cout << "Houla, une créature/terrain." << endl;
  }
  virtual ~CreatureTerrain() {}
```

```
  void afficher(ostream&) const;
};

void CreatureTerrain::afficher(ostream& out) const {
  out << "Une créature/terrain ";
  couleur.affiche(out, true);
  out << ' ' << nom << ' ' << attaque << "/" << defense << ' ';
  Carte::afficher(out);
  out << endl;
}

// ---------------------------------------------------------------------
class Jeu {
public:
  Jeu(){ cout << "On change de main" << endl; }
  virtual ~Jeu(){}
  void jette();
  void ajoute(Carte* carte) { contenu.push_back(carte); }
private:
  vector<Carte*> contenu;
friend ostream& operator<<(ostream&, const Jeu&);
};

ostream& operator<<(ostream& out, const Jeu& j) {
  for (auto carte : j.contenu)
    out << "  + " << *carte;
  return out;
}

void Jeu::jette() {
  cout << "Je jette ma main." << endl;
  for (auto& carte : contenu) { delete carte; }
  contenu.clear();
}

// ---------------------------------------------------------------------
int main()
{
  Jeu mamain;

  mamain.ajoute(new Terrain(Couleur::BLEU));
  mamain.ajoute(new Creature(6, "Golem", 4, 6));
  mamain.ajoute(new Sortilege(1, "Croissance Gigantesque",
           "La créature ciblée gagne +3/+3 jusqu'à la fin du tour"));
  mamain.ajoute(new CreatureTerrain(2, "Ondine", 1, 1, Couleur::BLEU));

  cout << "Là, j'ai en stock :" << endl;
  cout << mamain;

  mamain.jette();

  return 0;
}
```
