Compte Rendu TP1 CPOO

Paul Chaignon, Xavier Fraboulet

INSA de Rennes
4INFO, groupe 2.2

18 septembre 2013

Listing 1 – Carte.h

```
1  /**
2   * \file Carte.h
3   * \brief Delcarations pour la classe Carte.
4   * \author Paul Chaignon
5   * \author Xavier Fraboulet
6   * \version 1.0
7   * \date 11/09/13
8   */
9
10
11 #ifndef CARTE_H
12 #define CARTE_H
13 #include <string>
14 #include <cstdlib>
15 #include <iostream>
16 #include <cassert>
17
18 /**
19  * \enum Couleur
20  * \brief Couleur possible pour une carte.
21  */
22 enum Couleur {
23   Pique = 1,
24   Trefle,
25   Coeur,
26   Carreau
27 };
28
29 /**
30  * \enum Hauteur
```

```
31  * \brief Hauteurs possibles pour une carte.
32  */
33 enum Hauteur {
34   As = 1,
35   Deux,
36   Trois,
37   Quatre,
38   Cinq,
39   Six,
40   Sept,
41   Huit,
42   Neuf,
43   Dix,
44   Valet,
45   Dame,
46   Roi
47 };
48
49 /**
50  * \enum Joueur
51  * \brief Deux joueurs possibles.
52  */
53 enum Joueur {
54   nord = 1,
55   sud
56 };
57
58 /**
59  * \class Carte
60  * \brief Decrit une carte et ses liaisons au paquet.
61  */
62 class Carte {
63 private:
64   Joueur joueur;
65   Hauteur hauteur;
66   Couleur couleur;
67   /**
68    * Successeur de la carte dans le paquet.
69    */
70   Carte *succ;
71   /**
72    * Tetes et queues des deux paquets de cartes.
73    */
74   static Carte *teteN, *teteS, *queueN, *queueS;
75
76 public:
77   Carte(Couleur, Hauteur, char);
78   Hauteur getHauteur() const;
79   Joueur getJoueur();
```

```
80    Couleur getCouleur() const;
81    static Carte* getNTete();
82    static Carte* getSTete();
83    static Carte* getNQueue();
84    static Carte* getSQueue();
85    Carte* getSucc();
86    bool supAbs(Carte);
87    bool egale(Carte);
88    void afficher();
89    static void afficherN();
90    static void afficherS();
91    void passerDerriere();
92    void changerProp();
93
94  };
95
96  /**
97   * \fn inline Hauteur Carte::getHauteur() const
98   * \brief Accesseur pour la hauteur.
99   * \return La hauteur de la carte.
100  */
101 inline Hauteur Carte::getHauteur() const {
102   return this->hauteur;
103 }
104
105 /**
106  * \fn inline Joueur Carte::getJoueur()
107  * \brief Accesseur pour le joueur proprietaire.
108  * \return Le joueur proprietaire de la carte.
109  */
110 inline Joueur Carte::getJoueur() {
111   return this->joueur;
112 }
113
114 /**
115  * \fn inline Couleur Carte::getCouleur() const
116  * \brief Accesseur pour la couleur.
117  * \return La couleur de la carte.
118  */
119 inline Couleur Carte::getCouleur() const {
120   return this->couleur;
121 }
122
123 /**
124  * \fn inline Carte* Carte::getNTete()
125  * \brief Accesseur pour la tete du paquet N.
126  * \return La tete du paquet N.
127  */
128 inline Carte* Carte::getNTete() {
```

```
129   return Carte::teteN;
130 }
131
132 /**
133  * \fn prototype de la fontion
134  * \brief Accesseur pour la tete du paquet S.
135  * \return La tete du paquet S.
136  */
137 inline Carte* Carte::getSTete() {
138   return Carte::teteS;
139 }
140
141 /**
142  * \fn prototype de la fontion
143  * \brief Accesseur pour la queue du paquet N.
144  * \return La queue du paquet N.
145  */
146 inline Carte* Carte::getNQueue() {
147   return Carte::queueN;
148 }
149
150 /**
151  * \fn prototype de la fontion
152  * \brief Accesseur pour la queue du paquet S.
153  * \return La queue du paquet S.
154  */
155 inline Carte* Carte::getSQueue() {
156   return Carte::queueS;
157 }
158
159 /**
160  * \fn prototype de la fontion
161  * \brief Accesseur pour le successeur.
162  * \return Le successeur de la carte.
163  */
164 inline Carte* Carte::getSucc() {
165   return this->succ;
166 }
167
168 #endif
```

Listing 2 – Carte.cpp

```
1 /**
2  * \file Carte.cpp
3  * \brief Methodes et initialisation de la classe Carte.
4  * \author Paul Chaignon
5  * \author Xavier Fraboulet
6  * \version 1.0
7  * \date 11/09/13
```

```
 */

#include "Carte.h"

Carte *Carte::teteN = 0;
Carte *Carte::teteS = 0;
Carte *Carte::queueN = 0;
Carte *Carte::queueS = 0;

/**
 * Tableau avec les correspondances structures - noms pour la methode afficher.
 */
std::string hauteurs[] = {"As", "Deux", "Trois", "Quatre", "Cinq", "Six", "Sept",
    "Huit", "neuf", "Dix", "Valet", "Dame", "Roi"};
std::string couleurs[] = {"Pique", "Trefle", "Coeur", "Carreau"};

/**
 * \fn Carte(Couleur, Hauteur, char)
 * \brief Constructeur de Carte.
 * \param[in] c La couleur.
 * \param[in] h La hauteur.
 * \param[in] proprio Le proprietaire sous forme de charactere.
 */
Carte::Carte(Couleur c, Hauteur h, char proprio): couleur(c), hauteur(h) {
  if(proprio == 'N') {
    joueur = Joueur::nord;
    if(teteN == 0) {
      teteN = this;
      queueN = this;
    } else {
      queueN->succ = this;
      queueN = this;
    }
  } else if(proprio == 'S') {
    joueur = Joueur::sud;
    if(teteS == 0) {
      teteS = this;
      queueS = this;
    } else {
      queueS->succ = this;
      queueS = this;
    }
  } else {
    assert(false);
  }
  this->succ = 0;
}

/**
```

```
56  * \fn bool Carte::supAbs(Carte)
57  * \brief Compare la hauteur d'une carte avec la carte courante.
58  * \param[in] carte La carte a comparer avec la carte courante.
59  * \return Vrai si la carte courante est "superieure" a la carte en parametre.
60  */
61 bool Carte::supAbs(Carte carte) {
62   return this->hauteur > carte.hauteur;
63 }
64
65 /**
66  * \fn bool Carte::egale(Carte)
67  * \brief Compare la hauteur d'une carte avec la carte courante.
68  * \param[in] carte La carte a comparer avec la carte courante.
69  * \return Vrai si les deux carte sont de meme hauteur.
70  */
71 bool Carte::egale(Carte carte) {
72   return this->hauteur == carte.hauteur;
73 }
74
75 /**
76  * \fn void Carte::afficher()
77  * \brief Affiche la carte.
78  */
79 void Carte::afficher() {
80   std::cout << hauteurs[this->hauteur-1] << " de " << couleurs[this->couleur - 1]
        << " (" << this->hauteur << ", " << this->couleur << ")";
81 }
82
83 /**
84  * \fn void Carte::afficherN()
85  * \brief Affiche l'ensemble des cartes du paquet N.
86  */
87 void Carte::afficherN() {
88   Carte *carte = teteN;
89   while(carte != 0) {
90     carte->afficher();
91     std::cout << std::endl;
92     carte = carte->getSucc();
93   }
94 }
95
96 /**
97  * \fn void Carte::afficherS()
98  * \brief Affiche l'ensemble des cartes du paquet S.
99  */
100 void Carte::afficherS() {
101   Carte *carte = teteS;
102   while(carte != 0) {
103     carte->afficher();
```

```
104     std::cout << std::endl;
105     carte = carte->getSucc();
106   }
107 }
108 
109 /**
110  * \fn void Carte::passerDerriere()
111  * \brief Passe la carte a la fin du paquet.
112  */
113 void Carte::passerDerriere() {
114   if(this->joueur == Joueur::nord) {
115     queueN->succ = this;
116     queueN = this;
117     teteN = teteN->succ;
118     queueN->succ = 0;
119   } else {
120     queueS->succ = this;
121     queueS = this;
122     teteS = teteS->succ;
123     queueS->succ = 0;
124   }
125 }
126 
127 /**
128  * \fn void Carte::changerProp()
129  * \brief Change le proprietaire de la carte.
130  */
131 void Carte::changerProp() {
132   if(this->joueur == Joueur::nord) {
133     this->joueur = Joueur::sud;
134     teteN = teteN->succ;
135     queueS->succ = this;
136     queueS = this;
137     queueS->succ = 0;
138   } else {
139     this->joueur = Joueur::nord;
140     teteS = teteS->succ;
141     queueN->succ = this;
142     queueN = this;
143     queueN->succ = 0;
144   }
145 }
```