

Au cours de cette séance de TD, nous allons concevoir nos premières classes. Afin de nous aider dans notre processus de réflexion et son résultat nous allons utiliser un outil : les cartes CRC (*Class, Responsibilities, Collaborators*)¹. Ces petites cartes permettent de décrire chaque famille d'objets (chaque classe donc) en faisant abstraction de leur implémentation (du langage qui sera utilisé, de la syntaxe de ce langage, etc.). Une carte CRC est constituée de trois parties :

Un nom de la classe : Ce nom permet de créer un vocabulaire entre les différents acteurs de la conception d'une application. Il est donc important de trouver un nom qui décrit bien la classe. Ce nom doit être le plus évocateur possible car il a vocation à être utilisé dans un contexte de conception plus large.

Des responsabilités : Celles-ci identifient les problèmes qui doivent être résolus par la classe. Une manière de définir les responsabilités d'un objet est de déterminer ce que l'objet doit *savoir* et ce qu'il doit *faire*.

Les responsabilités qui répondent à *que doit savoir cet objet ?* incluent notamment :

- les valeurs que cet objet doit conserver ;

Les responsabilités qui répondent à *que doit faire cet objet ?* incluent notamment :

- effectuer un calcul particulier ;
- modifier son état interne d'une certaine façon ;
- créer et initialiser d'autres objets ;
- contrôler et coordonner les activités d'autres objets.

Des collaborateurs : Ce sont les noms des classes avec lesquelles la classe que l'on décrit devra collaborer pour assurer ses responsabilités. Autrement, les noms des classes auxquelles notre classe enverra des messages (appels de méthode) ou desquelles elle recevra des messages.

La figure ci-dessous donne une illustration d'une carte CRC.

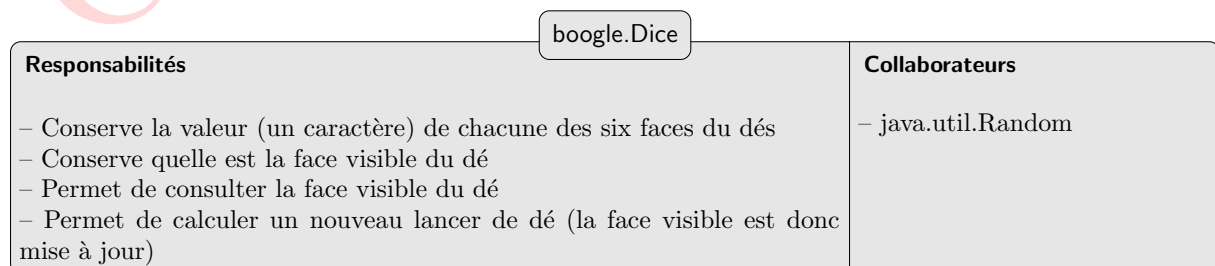


FIGURE 1 – Un exemple de carte CRC.

★ **Exercice 1.** L'objectif de cet exercice est de déterminer les classes nécessaires à la réalisation d'un interpréteur rudimentaire LOGO.

Le langage LOGO a été créé au dans les années 1960 au Massachusetts Institute of Technology (MIT) par Wally Feurzeig et Seymour Papert. C'est un bon langage d'initiation à la programmation en particulier pour les enfants grâce au côté ludique de la tortue graphique. La tortue graphique peut effectuer les actions suivantes : avancer de N pixels, tourner à droite de N degrés, tourner à gauche de N degrés, reculer de N pixels, se cacher, se montrer, lever le crayon, poser le crayon, changer la couleur du crayon. La classe `tortue.Screen` représente l'écran graphique de notre application. L'interface publique de cette classe est la suivante :

```

1 public Screen(int width, int height) ;
2 public void setForegroundColor(Color c) ;
3 public void drawLine(int xA, int yA, int xB, int yB) ;
4 public void fillRectangle(int xA, int yA, int xB, int yB) ;
5 public void setBackgroundColor(Color c) ;
6 public void clear() ;

```

1. [Benck89] K. Beck and W. Cunningham. A Laboratory for Teaching Object-Oriented Thinking, in Proceedings of OOPSLA'89. pp. 1–6, 1989. ACM Press. <http://doi.acm.org/10.1145/74877.74879>

▷ **Question 1.** Réaliser la carte CRC correspondant à la classe `tortue.Screen`.

Réponse

tortue.Screen

Responsabilités

- Permet de définir la couleur avec laquelle on dessine
- Permet de dessiner une ligne à l'écran
- Permet de définir la couleur de fond d'écran
- Offre la possibilité d'effacer l'écran

Collaborateurs

Fin réponse

▷ **Question 2.** Réaliser la carte CRC correspondant à la classe `tortue.Tortoise` qui modélisera à la tortue.

Réponse

tortue.Tortoise

Responsabilités

- Permet de définir la couleur du crayon
- Permet de poser/lever le crayon
- Permet de déplacer la tortue (avancer, reculer)
- Permet de faire tourner la tortue (droite, gauche)
- Permet de cacher/montrer la tortue
- Permet de savoir si la tortue est visible
- Conserve la position actuelle de la tortue (x,y)

Collaborateurs

- tortue.Screen

Fin réponse

▷ **Question 3.** À partir de la carte CRC déterminer une interface publique possible pour la classe `tortue.Tortoise`.

Réponse

```

1 public Tortoise() ;
2 public Tortoise(int initialX, int initialY) ;
3 public void forward(int pixelCount) ;
4 public void backward(int pixelCount) ;
5 public void turnLeft(int angleInDegree) ;
6 public void turnRight(int angleInDegree) ;
7 public int getX() ;
8 public int getY() ;
9 public void setBrushColor(Color c) ;
10 public Color getBrushColor() ;
11 public void setBrushDown() ;
12 public void setBrushUp() ;
13 public void hide() ; // void setVisible(boolean isVisible)
14 public void show() ;
15 public void boolean isVisible() ;

```

Fin réponse

▷ **Question 4.** Déterminer la carte CRC pour la classe principale dénommée `tortue.Main`. Sachant que l'application fonctionne de la manière suivante. Une fois exécutée, l'écran graphique apparaît et il est demandé à l'utilisateur de saisir une commande. Une fois la commande saisie, si celle-ci est valide, elle est exécutée et l'application demande à nouveau à l'utilisateur de saisir une commande.

Les commande possibles sont les suivantes :

- FD `x` pour faire avancer la tortue de `x` pixels
- BD `x` pour faire reculer la tortue de `x` pixels

- LT d pour faire tourner à gauche la tortue de d degrés
- RT d pour faire tourner à droite la tortue de d degrés
- PENUP pour lever le crayon
- PENDOWN pour poser le crayon
- CLEAR pour effacer l'écran
- BC c pour choisir la couleur numéro c du crayon (0 : blanc, 1 : noir, 2 : bleu, 3 : rouge, 4 : vert)
- EXIT pour quitter l'application

Réponse

tortue.Main

Responsabilités

- Doit instancier un nouvel écran et le conserver
- Doit instancier une nouvelle tortue et la conserver
- Capable de lire les commandes données par l'utilisateur
- Valide les commandes saisie par l'utilisateur
- Exécute les commandes valides

Collaborateurs

- tortue.Screen
- tortue.Tortoise

Fin réponse

▷ **Question 5.** Déterminer une interface publique de la classe principale de l'application.

Réponse

```
1 public static void main(String args[]) ;
```

Le reste n'est pas visible ; b (mais je vous le donnerai l'an prochain ...)

Fin réponse

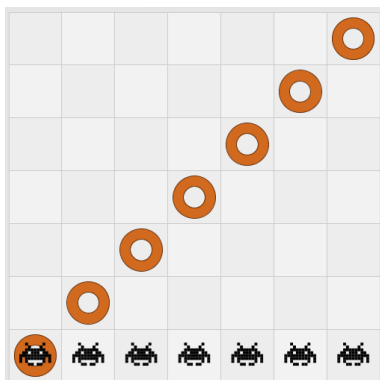
▷ **Question 6.** (*facultative*) Écrivez le code source des classes Tortue.Main et Tortue.Tortoise en supposant que la classe Tortue.Screen vous est fournie.

Réponse

C'est surtout pour occuper ceux qui font les malins (qui a dit les "Dossiers" ?)

Fin réponse

★ **Exercice 2.** Les buggles sont de retour ;)



Les buggles ? Qu'est ce que c'est ?

Les buggles sont de petites bêtes qui obéissent aux ordres que vous leur donnez. Ils habitent sur un monde en forme de grille. Tout comme les tortues, ils peuvent se déplacer sur ce monde (avancer, reculer, tourner d'un quart de tour à droite, tourner d'un quart de tour à gauche, lever un crayon, baisser un crayon, définir la couleur d'un crayon). Le monde des buggles peut quand à lui contenir des Baggles (les fameux biscuits tant appréciés par les buggles), mais également des murs qui empêchent les buggles d'avancer. Les buggles doivent donc être capables de savoir si ils sont faces à un mur ou non. De même ils peuvent savoir si ils sont au dessus d'un baggle, le prendre et le déposer.

Extrait de Java Learning Machine, (JLM)

<http://www.loria.fr/~quinson/JLM.html>.

▷ **Question 7.** Réaliser les différentes cartes CRC nécessaires à la conception de cette application. On réutilisera la carte CRC de la classe tortue.Screen vue dans l'exercice précédent.

Réponse

```
not_yet_implemented;d
```

Désolé, mais cela ne devrait pas poser de problème pour ceux qui connaissent JLM (niark niark pour une fois les “Dossiers” n’ont pas l’avantage)

Fin réponse

▷ **Question 8.** Déterminer les interfaces publiques des classes correspondantes aux cartes CRC que vous avez réalisées.

Réponse

```
not_yet_implemented;d
```

Voici rapidement ce que cela pourrait donner (extrait du code source de JLM).

```

1 class Buggle
2     public Buggle(BuggleWorld w) {
3         public Buggle(BuggleWorld world, String name, int x, int y, Direction direction, Color c, Color brushColor)
4
5         public boolean isBrushDown()
6         public void brushDown()
7         public void brushUp()
8
9         public Color getBrushColor()
10        public void setBrushColor(Color c)
11
12        public Color getColor()
13        public void setColor(Color c)
14
15        public Direction getDirection()
16        public void setDirection(Direction direction)
17
18        public void turnLeft()
19        public void turnRight()
20        public void turnBack()
21
22        public int getWorldHeight()
23        public int getWorldWidth()
24
25        public int getX()
26        public void setX(int x)
27        public int getY()
28        public void setY(int y)
29
30        public void setPos(int x, int y)
31
32        public void forward()
33        public void forward(int count)
34        public void backward()
35        public void backward(int count)
36
37        public boolean isFacingWall()
38
39        public boolean isOverBaggle()
40        public boolean isCarryingBaggle()
41        public void pickUpBaggle()
42        public void dropBaggle()
43
44        public void run()
45
46        public String toString()
47        public int hashCode()
48        public boolean equals(Object obj)
49
50
51 class BuggleWorld
52     public BuggleWorld(String name, int x, int y)
53
54     public int getWidth()
55     public int getHeight()
56
57     public void setCell(BuggleWorldCell c, int x, int y)
58     public BuggleWorldCell getCell(int x, int y)
59
60     public void setColor(int x, int y, Color c)
61
62     public String toString()
63     public int hashCode()
64     public boolean equals(Object obj)

```

```

65 class BuggleWorldCell
66     public BuggleWorldCell(BuggleWorld w, int x, int y)
67     public BuggleWorldCell(BuggleWorld w, int x, int y, Color color, boolean leftWall, boolean topWall)
68
69     public int getX()
70     public int getY()
71     public void setX(int newX)
72     public void setY(int newY)
73
74     public void setWorld(GridWorld w)
75     public GridWorld getWorld()
76
77     public void setColor(Color c)
78     public Color getColor()
79
80     public boolean hasTopWall()
81     public boolean hasLeftWall()
82     public void putLeftWall()
83     public void removeLeftWall()
84     public void putTopWall()
85     public void removeTopWall()
86
87     public boolean hasBaggle()
88     public Baggle getBaggle()
89     public void setBaggle(Baggle b)
90     public Baggle pickUpBaggle()
91
92     public int hashCode()
93     public boolean equals(Object obj)
94
95 class Baggle
96     public Baggle(BuggleWorldCell c)
97     public void setCell(BuggleWorldCell c)
98     public String toString() {
99

```

Fin réponse

★ Exercice 3. Schéma mémoire

Considérons le code source java donné ci-dessous.

```

1 public class A {
2     public int a1;
3     public int a2;
4
5     public A(int u, int v) {
6         a1=u;
7         a2=v;
8     }
9
10    public void met1(int u, int v) {
11        a1 = a1 + u;
12        a2 = a2 + v;
13    }
14
15    public A met2(int u, int v) {
16        return new A(a1+u, a2+v);
17    }
18
19    public boolean egale(A other) {
20        return ((this.a1 == other.a1) && (this.a2 == other.a2));
21    }
22
23    public String toString() {
24        return "("+ a1 + ","+ a2+")";
25    }
26 }
27

```

```

1 public class B {
2     public A b1;
3     public A b2;
4     public A b3;
5
6     public B(A x, A y, A z) {
7         b1=x;
8         b2=y;
9         b3=z;
10    }
11

```

```

12
13 public void met1(int u, int v) {
14     b1.met1(u,v);
15     b2.met1(u,v);
16     b3.met1(u,v);
17 }
18
19 public B met2(int u, int v) {
20     return new B(b1.met2(u,v), b2.met2(u,v), b3.met2(u,v));
21 }
22
23 public B met3(int u, int v) {
24     b1.met1(u,v);
25     b2.met1(u,v);
26     b3.met1(u,v);
27     return new B(b1,b2,b3);
28 }
29
30 public boolean egale(B other) {
31     return ( (this.b1).egale(other.b1) &&
32             (this.b2).egale(other.b2) &&
33             (this.b3).egale(other.b3) );
34 }
35
36 public String toString() {
37     return "[" + b1 + " ; " + b2 + " ; " + b3 + "]" ;
38 }
39 }

```

```

1 public class Test {
2
3     public static void main(String[] args) {
4         int n1;
5         int n2;
6         A p1, p2, p3;
7         B f1, f2, f3;
8
9         n1 = 1;
10        n2 = 7;
11        p1 = new A(n1,n2);
12        System.out.println("p1="+p1);
13        p2 = p1.met2(28,11);
14        p1.met1(28,11);
15        System.out.println("p1="+p1);
16        System.out.println("p2 (ou p1)="+p2);
17        System.out.println(p1==p2);
18        System.out.println(p1.egale(p2));
19        n1 = n1 + n2;
20        n2 = n1 + n2;
21        p3 = new A (n1, n2);
22        System.out.println("p3 (ou p2)="+p3);
23        p3.met1(21,3);
24        System.out.println("p3 (ou p2)="+p3);
25        System.out.println(p1 == p3);
26        System.out.println(p1.egale(p3));
27        p1=p2;
28        System.out.println(p1==p2);
29        System.out.println(p1.egale(p2));
30        System.out.println("");
31
32        // point_1
33        System.out.println("---- POINT 1 ----");
34
35        p1 = new A(1,4);
36        p2 = new A(5,7);
37        p3 = new A(8,9);
38        f1 = new B(p1,p2,p3);
39        System.out.println("f1="+f1);
40        f1.met1(10,20);
41        System.out.println("f1="+f1);
42        f2 = f1.met2(5,10);
43        System.out.println("f1="+f1);
44        System.out.println("f2="+f2);
45        System.out.println(f1==f2);
46        f1.met1(5,10);
47        System.out.println("f1="+f1);
48        System.out.println(f1==f2);
49        System.out.println(f1.egale(f2));
50        System.out.println("");
51
52        // point_2
53        System.out.println("---- POINT 2 ----");
54

```

```

55     p1 = new A(2,4);
56     p2 = new A(3,6);
57     p3 = new A(4,8);
58     f1 = new B(p1,p2,p1);
59     f2 = new B(new A(2,4), new A(3,6), new A(2,4));
60     System.out.println(f1==f2);
61     System.out.println(f1.egale(f2));
62     System.out.println("f1="+f1);
63     f1.met1(10,20);
64     System.out.println("f1="+f1);
65     System.out.println("f2="+f2);
66     f3 = f2.met3(100,200);
67     System.out.println("f2="+f2);
68     System.out.println("f3="+f3);
69     System.out.println(f2==f3);
70     System.out.println(f2.egale(f3));
71
72     // point_3
73     System.out.println("---- POINT 3 ----");
74 }
75 }

```

▷ **Question 9.** Après avoir extrait l'interface publique de chaque classe, essayez de déduire ce que décrivent les classes fournies.

Réponse

L'objectif recherché est de faire le travail inverse du travail habituellement demandé. On fera remarquer aux étudiants la difficulté de maintenir du code qui n'est pas proprement nommé (la classe A et B). Un petit rappel rapide sur les concepts de pile et de tas serait le bien venu également.

Fin réponse

▷ **Question 10.** Calculez et donnez le résultat de l'exécution de la classe **Test**. Pour cela, vous réaliserez les schémas de la mémoire aux trois étapes identifiées dans le code source de la méthode **main** de la classe **Test**.

Réponse

En annexe ;-)

Fin réponse