Le monde de l'objet
TD/TP n°2

ANNEXE POUR REUSSIR …

Sommaire

I. Langage Java …

Un problème sur un « concept objet » en Java vu en cours ?

Il suffit de regarder **le tutoriel officiel Java** (que vous avez pris soin d'installer sur votre machine pour gagner du temps) et de vous rendre ...

En ligne :

http://java.sun.com/docs/books/tutorial/java/concepts/index.html

Ou sur votre machine :

[HOME_TUTORIAL]/tutorial/java/concepts/index.html

Un problème de syntaxe Java ?

RAPPELEZ-VOUS QUE LA SYNTAXE DU JAVA EST TRES PROCHE DE CELLE DU C

Il suffit de regarder **le tutoriel officiel Java** (que vous avez pris soin d'installer sur votre machine pour gagner du temps) et de vous rendre ...

En ligne :

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/index.html

Sur votre machine :

[HOME_TUTORIAL]/tutorial/java/nutsandbolts/index.html

Comment utiliser une classe Java ?

Il suffit de regarder **la documentation officiel des API Java - la « JavaDoc » -** (que vous avez pris soin d'installer sur votre machine pour gagner du temps) et de consultez le manuel de la classe en question.

En ligne :

http://java.sun.com/javase/6/docs/api/index.html

Sur votre machine :

[HOME_JAVADOC]/docs/api/index.html

En Java tout est (presque) Object (cf. String , toString())

Comme vu en cours tout objet en Java hérite/dérive/étend « Object ».

Vérifiez donc dans la « JavaDoc » que vous trouvez la classe **Object** dans le package **java.lang.**

Recherchez également dans **Object** la méthode «**toString()** » et comprenez l'équivalence suivante :

```
System.out.print(vaisseau); ⇔ System.out.print(vaisseau.toString());
```

cf. ligne 20 du code source de « TestVaisseau » plus loin

Vous pouvez aussi observer **(java.lang.)String** et **(java.lang.)System**, vous verrez qu'elles étendent bien « Object » et qu'elles fournissent des services très utiles (ex. System.**out.print(...)**)

Pour faciliter vos tests et « debuggage » dans votre classe **Vaisseau**, vous comprendrez facilement l'intérêt de **redéfinir** la méthode toString() comme ci-dessous (exemple)

```
public abstract class Vaisseau {

[…]

    public String toString() {

        return "Vaisseau{ modele: '"+modele+"'"+

               ", force: "+forceAttaque+

               ", defense: "+niveauDefense+

               ", position: ["+position.x+","+position.y+"] }";

    }

[…]

}
```

II. Les Vaisseaux

Code de la classe de test « TestVaisseau »

```
1  import java.awt.Point;
2
3  public class TestVaisseau {
4
5      private Vaisseau[] vaisseaux;
6
7      public TestVaisseau() {
8          vaisseaux = new Vaisseau[6];
9          vaisseaux[0] = new XWing();
10         vaisseaux[1] = new TFighter();
11         vaisseaux[2] = new MilleniumFalcon();
12         vaisseaux[3] = new XWing(new Point(1, 2));
13         vaisseaux[4] = new TFighter(new Point(3, 4));
14         vaisseaux[5] = new MilleniumFalcon(new Point(5, 6));
15     }
16
17     public void testVaisseaux() {
18         for (int i = 0; i < vaisseaux.length; i++) {
19             Vaisseau vaisseau = vaisseaux[i];
20             System.out.print(vaisseau);/*System.out.print(vaisseau.toString())*/
21             System.out.println(" operationnel : " + vaisseau.estOperationnel());
22         }
23
24     }
25
26     public static void main(String[] args) {
27         System.out.println(" -=- TEST VAISSEAU -=-");
28         TestVaisseau test = new TestVaisseau();
29         test.testVaisseaux();
30     }
31 }
```

Affichage à l'exécution de « TestVaisseau »

```
-=- TEST VAISSEAU -=-

Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [4,0] } operationnel : true

Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [6,4] } operationnel : true

Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 10, position: [3,6] } operationnel : true

Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [1,2] } operationnel : true

Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [3,4] } operationnel : true

Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 10, position: [5,6] } operationnel : true
```

Utiliser un objet « Point » pour la position du vaisseau !

En observant la ligne 1 de la classe « TestVaisseau » vous remarquerez l'utilisation de la classe **Point** dans le package **java.awt**. Vous pouvez vérifier dans la « JavaDoc » que cette classe fonctionne de manière aussi simple que la classe Point de votre premier TP.

Les lignes 9 à 14 permettent de mettre en évidence la création de vaisseaux selon deux modes : aléatoire (lignes 9 à 11) ou imposé (lignes 12 à 14) pour sa position. Ceci implique que les sous-classes de Vaisseau possèdent **deux constructeurs** (et deux également pour Vaisseau)

Vous devriez sûrement avoir un attribut « position » de type « Point » dans votre classe Vaisseau ...

Comment générer une position aléatoire ?

Utilisez la « JavaDoc » et consultez le manuel de la classe **Math** dans le package **java.lang** et trouvez la méthode statique à utiliser.

Rappelez-vous que nous sommes dans une grille de 10x10 et qu'un flottant aléatoire choisi dans [0.0f, 1.0f[doit être multiplié par 10 pour obtenir un entier dans un intervalle [0, 10] ...

III. Les Belligérants

Code de la classe de test « TestBelligerant »

```
 1 public class TestBelligerant {
 2
 3     public static void tester(Belligerant b) {
 4         System.out.println(" - Belligerant - ");
 5         System.out.println("Camp de ralliement : "+b.campDeRalliement());
 6         System.out.println("Vaisseaux opérationnels : "+b.etatFlotte());
 7         System.out.println(b);
 8     }
 9
10     public static void main(String[] args) {
11         System.out.println(" -=- TEST BELLIGERANT -=- ");
12         EmpireGalactique empire = new EmpireGalactique();
13         AllianceRebelle alliance = new AllianceRebelle();
14         tester(empire);
15         tester(alliance);
16     }
17 }
```

Affichage à l'exécution « TestBelligerant »

```
-=- TEST BELLIGERANT -=-
 - Belligerant -
Camp de ralliement : Empire Galactique sur l'Etoile Noire
Vaisseaux opérationnels : 4
Détail du belligérant 'Empire Galactique sur l'Etoile Noire' (4 vaisseaux opérationnels)
        Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [8,1] }
        Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [6,8] }
        Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
        Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [8,7] }
Fin du détail
 - Belligerant -
Camp de ralliement : Alliance Rebelle sur Endor
Vaisseaux opérationnels : 4
Détail du belligérant 'Alliance Rebelle sur Endor' (4 vaisseaux opérationnels)
        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [6,4] }
        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [6,1] }
        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [5,1] }
        Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 10, position: [0,8] }
Fin du détail
```

IV. La confrontation entre Vaisseaux (Combattant / Cible)

Code de la classe de test « TestCombattantCible »

```
 1 import java.awt.Point;
 2
 3 public class TestCombattantCible {
 4
 5     public static void testConfrontation(Combattant combattant, Cible cible) {
 6         System.out.println(" a) - AVANT confrontation ->");
 7         System.out.println(" Combattant : "+combattant);
 8         System.out.println(" Cible : "+cible);
 9         combattant.attaquer(cible);
10         System.out.println(" b) - APRES confrontation ->");
11         System.out.println(" Combattant : "+combattant);
12         System.out.println(" Cible : "+cible);
13     }
14
15     public static void afficherRapportCombattant(Vaisseau v) {
16         System.out.println(" c) - RAPPORT d'attaque - "+v);
17         System.out.println(" Point attaque : "+v.donnerPointAttaque());
18         System.out.println(" Resultat attaque : "+v.donnerResultatAttaque());
19         System.out.println(" ( -1 = INDEMNE, 0 = TOUCHE, 1 = DETRUIT) ");
20     }
21
22     public static void main(String[] args) {
23         System.out.println(" -=- TEST COMBATTANT CIBLE -=- ");
24         Vaisseau vaisseauEmpire = new TFighter(new Point(0,0));
25         Vaisseau vaisseauRebelle = new MilleniumFalcon(new Point(1,1));
26
27         System.out.println("1- Le vaisseau de l'Empire attaque :");
28         vaisseauEmpire.prendreOrdreAttaque(1, 1);
29         testConfrontation(vaisseauEmpire, vaisseauRebelle);
30         afficherRapportCombattant(vaisseauEmpire);
31
32         System.out.println("2- Le Rebelle attaque :");
33         vaisseauRebelle.prendreOrdreAttaque(1, 1);
34         testConfrontation(vaisseauRebelle, vaisseauEmpire);
35         afficherRapportCombattant(vaisseauRebelle);
36
37         System.out.println("3- Le vaisseau de l'Empire attaque :");
38         vaisseauEmpire.prendreOrdreAttaque(0, 0);
39         testConfrontation(vaisseauEmpire, vaisseauRebelle);
40         afficherRapportCombattant(vaisseauEmpire);
41
42         System.out.println("4- Le Rebelle attaque :");
43         vaisseauRebelle.prendreOrdreAttaque(0, 0);
44         testConfrontation(vaisseauRebelle, vaisseauEmpire);
45         afficherRapportCombattant(vaisseauRebelle);
46     }
47 }
```

Affichage à l'exécution «TestCombattantCible »

```
-=- TEST COMBATTANT CIBLE -=-
1- Le vaisseau de l'Empire attaque :
 a) - AVANT confrontation ->
 Combattant : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 Cible : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 10, position: [1,1] }
 b) - APRES confrontation ->
 Combattant : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 Cible : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 c) - RAPPORT d'attaque de T-Fighter
 Point attaque : java.awt.Point[x=1,y=1]
 Resultat attaque : 0 ( -1 = INDEMNE, 0 = TOUCHE, 1 = DETRUIT)
2- Le Rebelle attaque :
 a) - AVANT confrontation ->
 Combattant : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 Cible : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 b) - APRES confrontation ->
 Combattant : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 Cible : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 c) - RAPPORT d'attaque de Millenium Falcon
 Point attaque : java.awt.Point[x=1,y=1]
 Resultat attaque : -1 ( -1 = INDEMNE, 0 = TOUCHE, 1 = DETRUIT)
3- Le vaisseau de l'Empire attaque :
 a) - AVANT confrontation ->
 Combattant : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 Cible : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 b) - APRES confrontation ->
 Combattant : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 Cible : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 c) - RAPPORT d'attaque de T-Fighter
 Point attaque : java.awt.Point[x=0,y=0]
 Resultat attaque : -1 ( -1 = INDEMNE, 0 = TOUCHE, 1 = DETRUIT)
4- Le Rebelle attaque :
 a) - AVANT confrontation ->
 Combattant : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 Cible : Vaisseau{ modele: 'T-Fighter', force: 2, defense: 4, position: [0,0] }
 b) - APRES confrontation ->
 Combattant : Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 8, position: [1,1] }
 Cible : Vaisseau{ modele: 'T-Fighter', force: 2, defense: -4, position: [0,0] }
 c) - RAPPORT d'attaque de Millenium Falcon
 Point attaque : java.awt.Point[x=0,y=0]
 Resultat attaque : 1 ( -1 = INDEMNE, 0 = TOUCHE, 1 = DETRUIT)
```

V. L'assaut entre Belligérant (Assaillant / Forteresse)

Code de la classe de test « TestAssaillantForteresse»

```
 1 public class TestAssaillantForteresse {
 2     public static void main(String[] args) {
 3         Forteresse f = new EmpireGalactique();
 4         Assaillant a = new AllianceRebelle();
 5         int nbAssauts = 0;
 6         int nbCibles;
 7         do {
 8             System.out.println("-------DEBUT ASSAUT---------");
 9             System.out.println("Assaillant avant assaut : "+a);
10             System.out.println("Forteresse avant assaut : "+f);
11             a.lanceAssaut(f);
12             System.out.println("Assaillant après assaut : "+a);
13             System.out.println("Forteresse après assaut : "+f);
14             System.out.println("----- -- FIN ASSAUT---------");
15             nbCibles = ((Belligerant)f).etatFlotte();
16             nbAssauts++;
17         } while (nbCibles > 0);
18
19         System.out.println("NB ASSAUTS = "+nbAssauts);
20     }
21 }
```

Affichage à l'exécution «TestCombattantCible »

```
...
-------DEBUT ASSAUT---------
Assaillant avant assaut : Détail du belligérant 'Alliance Rebelle sur Endor' (4 vaisseaux opérationnels)
        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [3,2] }
         - attaque [8,3] resultat 1

        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [8,6] }
         - attaque [1,0] resultat 1

        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [0,5] }
         - attaque [3,7] resultat 1

        Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 10, position: [6,6] }
         - attaque [8,0] resultat 1
Fin du détail
Forteresse avant assaut : Détail du belligérant 'Empire Galactique sur l'Etoile Noire' (1 vaisseaux opérationnels)
        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -8, position: [6,8] }
         - attaque [0,0] resultat -2

        Vaisseau{ modele: 'T-Fighter', force: 2, defense: 0, position: [0,8] }
         - attaque [0,0] resultat -2

        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -4, position: [7,4] }
         - attaque [0,0] resultat -2

        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -4, position: [2,5] }
         - attaque [0,0] resultat -2
Fin du détail
Assaillant après assaut : Détail du belligérant 'Alliance Rebelle sur Endor' (4 vaisseaux opérationnels)
        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [3,2] }
         - attaque [0,8] resultat 1

        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [8,6] }
         - attaque [8,7] resultat 1

        Vaisseau{ modele: 'X-Wing', force: 4, defense: 2, position: [0,5] }
         - attaque [0,5] resultat 1

        Vaisseau{ modele: 'Millenium Falcon', force: 8, defense: 10, position: [6,6] }
         - attaque [0,0] resultat 1
Fin du détail
Forteresse après assaut : Détail du belligérant 'Empire Galactique sur l'Etoile Noire' (0 vaisseaux opérationnels)
        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -8, position: [6,8] }
         - attaque [0,0] resultat -2

        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -4, position: [0,8] }
         - attaque [0,0] resultat -2

        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -4, position: [7,4] }
         - attaque [0,0] resultat -2

        Vaisseau{ modele: 'T-Fighter', force: 2, defense: -4, position: [2,5] }
         - attaque [0,0] resultat -2
Fin du détail
------- FIN ASSAUT---------
NB ASSAUTS = 57
```