

TP #2 Langage Java / Corrigé

Applications Réparties 2009-2010
Université Paris VIII / Parcours SRM / M1

Mise en place de l'environnement de développement :

Dans eclipse :

1. Définir un nouveau projet Java (File->New->Java Project) et lui donner le nom "tp2".
2. Pour chaque exercice, définir un package tp2.ex# où # correspond au numéro de l'exercice (click droit sur le nom du projet->New->Package). Le code lié à chaque exercice devra se trouver dans un package (espace de nommage) distinct.
3. Si un exercice fait référence à du code qui aurait déjà été implémenté dans un exercice précédent, il est possible d'importer les sources contenues dans ce package de manière à ne pas devoir tout réécrire.

Exercice 1 : Classes

I - A partir du code de la classe ci-dessous, répondez aux questions suivantes :

```
public class IdentifyMyParts {
    public static int x = 7;
    public int y = 3;
}
```

1. Quelles sont les variables de classes ? → **x**
2. Quelles sont les variables d'instance ? → **y**
3. Quelle est la sortie affichée par ce segment de code ?

```
IdentifyMyParts a = new IdentifyMyParts();
IdentifyMyParts b = new IdentifyMyParts();
a.y = 5;
b.y = 6;
a.x = 1;
b.x = 2;
System.out.println("a.y = " + a.y);
System.out.println("b.y = " + b.y);
System.out.println("a.x = " + a.x);
System.out.println("b.x = " + b.x);
System.out.println("IdentifyMyParts.x = " + IdentifyMyParts.x);
```

Correction :

La sortie affichée par ce segment de code est la suivante :

```
a.y = 5
b.y = 6
a.x = 1
b.x = 2
IdentifyMyParts.x = 7
```

Exercice 2 : Jeux de cartes

Ecrivez une classe *Card* dont les instances représentent une carte à jouer extraite d'un jeu de carte.

Vous devez prendre en compte les considérations suivantes :

- Une carte dispose de deux propriétés principales : son rang ("rank") et sa couleur ("suit").
- Ses 2 propriétés sont à définir dans la classe sous forme de **constantes** de la classe, pour vous aider le code ci-dessous vous est fournis.
- Le constructeur de la classe permet de fixer le "rank" et la "suit".
- La classe dispose de 2 accesseurs `getSuit()` et `getRank()`.
- La classe dispose d'une méthode statique `String rankToString(int rank)` permettant de convertir un rang en chaîne de caractères.
- La classe dispose d'une méthode statique `String suitToString(int suit)` permettant de convertir une couleur en chaîne de caractères.

Pour finir, définissez dans la classe *Card* deux méthodes statiques `isValidRank` et `isValidSuit` permettant de s'assurer par retour de booléen que le rang et la couleur passés en argument du constructeur de la classe sont valides.

- Vous pouvez utiliser l'instruction `assert` pour tester la valeur de retour des méthodes `isValidRank` et `isValidSuit`. Si l'argument de `assert` est faux (*false*) alors une erreur est levée et l'exécution du programme s'arrête.
- Par exemple : `assert isValidRank(rank);`

Correction :

```
public class Card {
    private final int rank;
    private final int suit;

    // Kinds of suits
    public final static int DIAMONDS = 1;
    public final static int CLUBS = 2;
    public final static int HEARTS = 3;
    public final static int SPADES = 4;
}
```

```

// Kinds of ranks
public final static int ACE = 1;
public final static int DEUCE = 2;
public final static int THREE = 3;
public final static int FOUR = 4;
public final static int FIVE = 5;
public final static int SIX = 6;
public final static int SEVEN = 7;
public final static int EIGHT = 8;
public final static int NINE = 9;
public final static int TEN = 10;
public final static int JACK = 11;
public final static int QUEEN = 12;
public final static int KING = 13;

public Card(int rank, int suit) {
    assert isValidRank(rank);
    assert isValidSuit(suit);
    this.rank = rank;
    this.suit = suit;
}

public int getSuit() {
    return suit;
}

public int getRank() {
    return rank;
}

public static boolean isValidRank(int rank) {
    return ACE <= rank && rank <= KING;
}

public static boolean isValidSuit(int suit) {
    return DIAMONDS <= suit && suit <= SPADES;
}

public static String rankToString(int rank) {
    switch (rank) {
    case ACE:
        return "Ace";
    case DEUCE:
        return "Deuce";
    case THREE:
        return "Three";
    case FOUR:
        return "Four";
    case FIVE:
        return "Five";
    case SIX:
        return "Six";
    case SEVEN:
        return "Seven";
    case EIGHT:
        return "Eight";
    case NINE:
        return "Nine";
    case TEN:
        return "Ten";
    case JACK:
        return "Jack";
    case QUEEN:
        return "Queen";
    case KING:
        return "King";
    default:
        //Handle an illegal argument. There are generally two
        //ways to handle invalid arguments, throwing an exception
        //(see the section on Handling Exceptions) or return null
        return null;
    }
}

public static String suitToString(int suit) {
    switch (suit) {
    case DIAMONDS:
        return "Diamonds";
    case CLUBS:
        return "Clubs";
    case HEARTS:

```

```

        return "Hearts";
    case SPADES:
        return "Spades";
    default:
        return null;
    }
}

public static void main(String[] args) {
    assert rankToString(ACE) == "Ace";
    assert rankToString(DEUCE) == "Deuce";
    assert rankToString(THREE) == "Three";
    assert rankToString(FOUR) == "Four";
    assert rankToString(FIVE) == "Five";
    assert rankToString(SIX) == "Six";
    assert rankToString(SEVEN) == "Seven";
    assert rankToString(EIGHT) == "Eight";
    assert rankToString(NINE) == "Nine";
    assert rankToString(TEN) == "Ten";
    assert rankToString(JACK) == "Jack";
    assert rankToString(QUEEN) == "Queen";
    assert rankToString(KING) == "King";

    assert suitToString(DIAMONDS) == "Diamonds";
    assert suitToString(CLUBS) == "Clubs";
    assert suitToString(HEARTS) == "Hearts";
    assert suitToString(SPADES) == "Spades";
}
}

```

Une fois la classe *Card* mise au point, écrire une classe *Deck* dont les instances représentent un paquet de carte entier. Cette classe va reposer sur un tableau à deux dimensions pour stocker les 52 cartes du paquet (une dimension pour la couleur et une dimension pour le rang).

Correction :

```

import java.util.*;

public class Deck {

    public static int numSuits = 4;
    public static int numRanks = 13;
    public static int numCards = numSuits * numRanks;

    private Card[][] cards;

    public Deck() {
        cards = new Card[numSuits][numRanks];
        for (int suit = Card.DIAMONDS; suit <= Card.SPADES; suit++) {
            for (int rank = Card.ACE; rank <= Card.KING; rank++) {
                cards[suit-1][rank-1] = new Card(rank, suit);
            }
        }
    }

    public Card getCard(int suit, int rank) {
        return cards[suit-1][rank-1];
    }
}

```

Une fois les classes *Card* et *Deck* mises au point, écrire une classe *DisplayDeck* contenant une méthode *main* pour les tester : ce programme va créer un paquet de cartes et afficher chaque carte.

Correction :

```

import java.util.*;

public class DisplayDeck {
    public static void main(String[] args) {
        Deck deck = new Deck();
        for (int suit = Card.DIAMONDS; suit <= Card.SPADES; suit++) {
            for (int rank = Card.ACE; rank <= Card.KING; rank++) {
                Card card = deck.getCard(suit, rank);
                System.out.format("%s of %s%n",
                    card.rankToString(card.getRank()),
                    card.suitToString(card.getSuit()));
            }
        }
    }
}

```

Exercice 3 : Objets

Question 1 - Trouver l'erreur dans cette portion de code :

```
public class SomethingIsWrong {
    public static void main(String[] args) {
        Rectangle myRect;
        myRect.width = 40;
        myRect.height = 50;
        System.out.println("myRect's area is " + myRect.area());
    }
}
```

Correction :

Le code ne crée pas un objet Rectangle, il indique uniquement sa déclaration. Cette erreur simpliste est détectée à la compilation, mais il existe des cas plus réalistes où myRect aurait pu être initialisé à null (par exemple dans un constructeur) puis utilisé dans cette méthode sans que le compilateur ne puisse détecter l'erreur. Dans ce cas, la ligne myRect.width = 40; aurait levé une exception NullPointerException à l'exécution.

Question 2 - Le code suivant crée une instance Point et une instance de Rectangle :

- Combien de références à ces objets existent après l'exécution de cette portion de code ?
- Est-ce que les instances sont susceptibles d'être traitées par le Garbage-Collector (ramasse-miettes) du runtime Java ?

```
...
Point point = new Point(2,4);
Rectangle rectangle = new Rectangle(point, 20, 20);
point = null;
...
```

Correction :

Il existe une référence à l'instance de Point créée ainsi qu'une référence à l'instance de Rectangle créée. Du fait que ces références soient stockées dans des variables (point et rectangle), les instances ne sont pas éligibles pour être supprimés par le Garbage-Collector.

Question 3 - Expliquez si il possible de détruire des instances dans un programme, et comment ?

Correction :

Un programme ne peut pas détruire explicitement les instances qu'il manipule. Par contre, il est possible de passer toutes les variables référençant un objet à null ou les faire pointer vers une autre instance, dans ce cas l'objet deviens éligible pour être supprimés par le Garbage-Collector.

Exercice 1 - Corrigez la classe SomethingIsWrong vue précédemment.

Correction :

```
public class SomethingIsWrong {
    public static void main(String[] args) {
        Rectangle myRect = new Rectangle();
        myRect.width = 40;
        myRect.height = 50;
        System.out.println("myRect's area is " + myRect.area());
    }
}
```

Exercice 2 - A partir de la classe donnée ci-dessous, écrivez du code pour créer une instance de cette classe, initialiser et afficher les valeurs de ses variables d'instances.

```
public class NumberHolder {
    public int anInt;
    public float aFloat;
}
```

Correction :

```
public class NumberHolderDisplay {
    public static void main(String[] args) {
        NumberHolder aNumberHolder = new NumberHolder();
        aNumberHolder.anInt = 1;
        aNumberHolder.aFloat = 2.3f;
        System.out.println(aNumberHolder.anInt);
        System.out.println(aNumberHolder.aFloat);
    }
}
```