NFA032 Programmation objet avec Java

Centre CNAM de Paris

Première session 2015 semestre 2

Documents et calculatrice interdits. Le barème est donné à titre indicatif.

Exercice 1 : programmation objet (5 points)

On donne le squelette de la classe des dates. Vous pouvez utiliser cette classe dans les réponses à cet exercice.

```
public class Date{
  public Date(int jour, int mois, int annee){...}
  public boolean estAvant(Date d){...}
}
```

La méthode estAvant renvoie true si la date de l'objet sur lequel la méthode est appelée est antérieure à la date passée en paramètre.

Question 1

On veut écrire une classe représentant une bouteille de jus de fruit. La bouteille peut être fermée ou ouverte. Elle a une date limite de consommation (DLC). Par ailleurs, elle contient une certaine quantité de soda exprimée en centilitres.

Ecrivez cette classe avec les variables nécessaires, un constructeur, une méthode pour ouvrir la bouteille, une méthode pour verser une partie du contenu de la bouteille. Il n'y a pas de méthode pour remplir la bouteille : on considère que les bouteilles créées dans le programme représentent le produit tel qu'il sort de l'usine de fabrication.

Faites attention à assurer la cohérence de l'objet (par exemple, il n'est pas possible de verser avant d'avoir ouvert la bouteille).

Question 2

Modifiez la méthode qui verse le contenu de la bouteille pour éviter de servir une boisson périmée. La date du jour sera passée en paramètre à la méthode.

Question 3

Ecrivez une méthode main qui illustre la création d'un objet et l'appel aux différentes méthodes qu'il contient.

Exercice 2 : références (3 points)

```
class Grand{
    Petit premier;
    Petit second;
    Grand(Petit p1, Petit p2){
        premier = p1;
        second = p2;
    }
}
class Petit{
    int x;
    Petit(int xx){
        x = xx;
    }
}
public class Ref9{
    public static void main(String[] args){
        Petit lepetit = new Petit(5);
        Petit[] tableau = new Petit[3];
        Grand legrand;
        tableau[0] = new Petit(7);
        legrand = new Grand(tableau[0],tableau[0]);
        tableau[1] = new Petit(1);
    }
}
```

Question 1

Représentez au moyen d'un petit dessin les objets et tableaux existant à la fin de l'exécution du programme donné ci-dessus. Chaque objet sera représenté par un rectangle et chaque référence par une flèche ou une adresse. Attention : on ne vous demande pas un diagramme de classe. C'est chaque objet créé, chaque tableau et chaque référence qui doit être représenté.

Question 2

Donnez le code java permettant d'afficher à l'écran la valeur de la variable `x` de chaque objet instance de `Petit` créé par ce programme. Vous ne devez pas changer le code des classes fournies.

Exercice 3 : exceptions (4 points)

Un jeu de démineur est un jeu qui utilise un terrain avec des cases identifiées par un numéro de colonne et un numéro de ligne. On propose un programme qui représente ce terrain avec un tableau de boolean à deux dimensions. Le boolean est false si la case ne contient pas une mine et il est true si la case contient une mine. Le joueur doit entrer les coordonnées d'une case. S'il y a une mine dans cette case, il a perdu. Sinon,

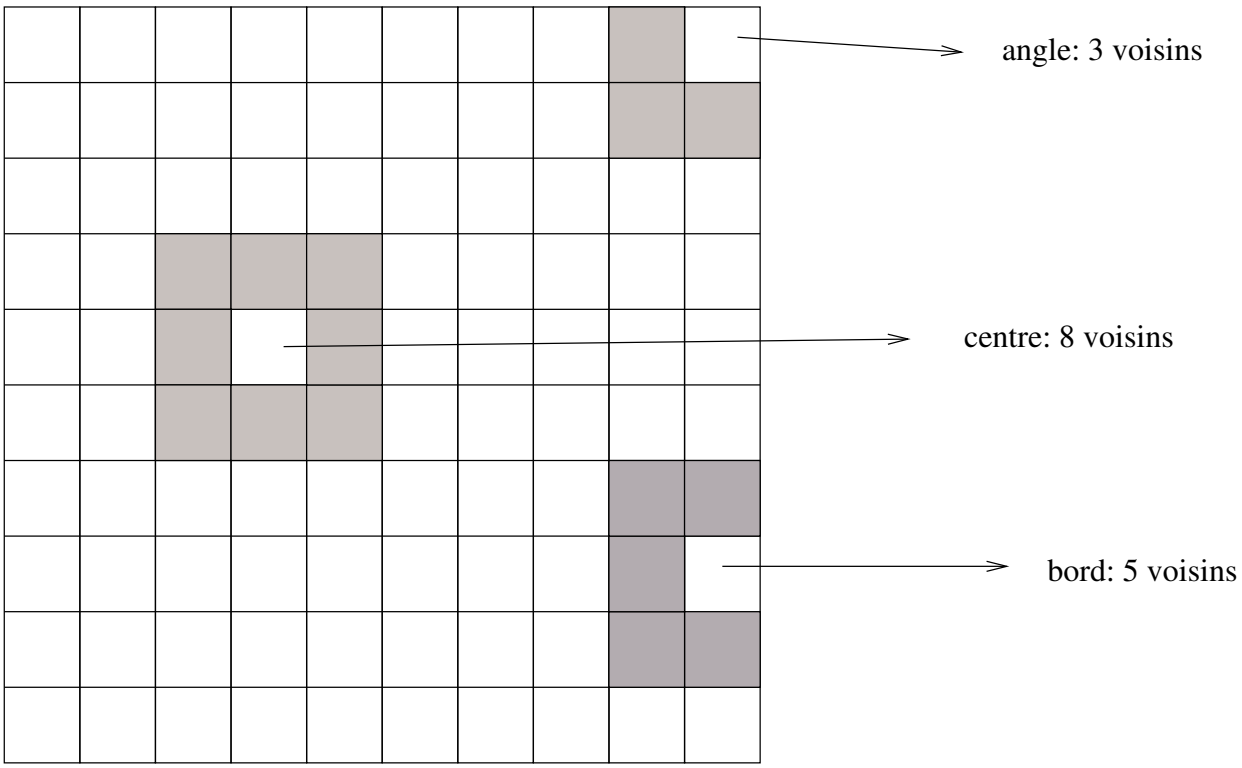
le programme lui affiche combien de mines il y a dans les cases adjacentes (les 8 cases qui entourent la case choisie).

On propose le programme suivant qu'il s'agit d'améliorer avec des exceptions.

```
public class Demineur{
    static boolean[][] champ;
    static boolean[][] placerMines(){
        return new boolean[10][10];
    }
    static void boumOuPasBoum(int col, int lig){
        if (champ[col][lig]){ // sous-entendu champ[col][lig]==true
                              // a compléter
        }
    }
    static int compteMines(int col, int lig){
        int res = 0;
        boumOuPasBoum(col,lig);
        // double-boucle qui parcourt les 8 voisins et la case choisie
        for (int i=col-1; i<=col+1; i++){
            for (int j=lig-1; j<=lig+1; j++){
                if (champ[i][j]){
                    res = res + 1;
                }
            }
        }
        return res;
    }
    public static void main(String[] args){
        champ=placerMines();
        int icol, ilig, nbmines;
        for (int i=0; i<10; i++){
            Terminal.ecrireStringln("Entrez le numéro de colonne: ");
            icol = Terminal.lireInt();
            Terminal.ecrireStringln("Entrez le numéro de ligne: ");
            ilig = Terminal.lireInt();
            nbmines = compteMines(icol,ilig);
            Terminal.ecrireStringln("Nombre de mines: " + nbmines);
        }
    }
}
```

On veut utiliser deux exceptions. La première doit être utilisée quand il y a une mine dans la case choisie par l'utilisateur. Le programme doit alors afficher le message BOUM et s'arrêter.

La deuxième exception à traiter est l'exception `ArrayIndexOutOfBoundsException` qui est levée automatiquement lorsqu'on essaie d'obtenir une case de tableau qui n'existe pas. Cette exception sera levée dans le programme lorsque l'utilisateur choisit un point qui a moins de 8 voisins : dans un angle (il n'y a que 3 voisins) ou sur un bord (il y a 5 voisins). Quand cette exception est levée dans la boucle de la méthode compteMines, cette boucle ne doit pas s'arrêter mais simplement passer à la case suivante sans ajouter de mine à la variable `res`.



Si l'exception ArrayIndexOutOfBoundsException est levée par boumOuPasBoum, il faut informer l'utilisateur que la case n'existe pas et que cela le prive d'un coup, mais le programme continue et l'utilisateur peut entrer la case suivante.

1. Déclarez l'exception dont on a besoin pour le cas d'explosion de mines en tant que sous-classe d'Exception (c'est-à-dire que ce ne doit pas être une sous-classe d'Error ou de RuntimeException).
2. Donnez le code de la méthode `BoumOuPasBoum` qui utilise l'exception déclarée à la question 1.
3. Modifiez le code de `compteMines` pour prendre en compte l'exception `ArrayIndexOutOfBoundsException`
4. Donnez les clauses throws des trois méthodes (`BoumOuPasBoum`, `compteMines` et `main`).

Exercice 4 : héritage (4 points)

```
class Personne{
    String nom, prenom;
    Personne(String n, String p){
        nom=n;
        prenom=p;
    }
    void affiche(){
        Terminal.ecrireStringln(nom + "␣" + prenom);
    }
}
class Monsieur extends Personne{
    Monsieur(....){
```

```
        ....
    }
    void affiche(){
        Terminal.ecrireStringln("Monsieur " + nom + " " + prenom);
    }
}
```

Question 1

Donnez le code du constructeur Monsieur. N'oubliez pas l'appel au constructeur de la super-classe.

Question 2

Pour chacune des instructions suivantes, dites si elle est correcte ou incorrecte. On la considère comme correcte si elle ne provoque aucune erreur, ni à la compilation, ni à l'exécution.

1. `Personne pers = new Personne("Hugo", "Victor");`
2. `Personne pers = new Monsieur("Hugo", "Victor");`
3. `Monsieur mons = new Personne("Hugo", "Victor");`
4. `Object obj = new Monsieur("Hugo", "Victor");`
5. `Monsieur mons = new Object("Hugo", "Victor");`
6. `Monsieur mons = new Object();`

Question 3

Dites ce qu'affiche à l'écran le programme suivant (qui est correct).

```
public class Herit{
    public static void main(String[] args){
        Personne pers1 = new Personne("Hugo","Victor");
        Personne pers2 = new Monsieur("Baudelaire","Charles");
        Monsieur mons = new Monsieur("Verlaine","Paul");
        pers1.affiche();
        pers2.affiche();
        mons.affiche();
    }
}
```

Exercice 5 : récursivité (4 points)

Question 1

On définit une fonction par :
- $f(0) = 1$
- $f(x) = f(x-1) * 5 + 1$ si $x > 0$

Ecrivez une méthode java qui calcule la valeur de la fonction `f` pour une valeur de x passée en paramètre.

Question 2

On donne la classe récursive suivante :

```
public class Route{
    String ville;
    int distance;
    Route suivante;
    Route(String v, int d, Route s){
        ville = v;
        distance = d;
        suivante = s;
    }
}
```

1. Utilisez cette classe pour créer une structure de 3 objets représentant une route partant de Paris, passant par Beaune et arrivant à Lyon, avec comme distances 330 kilomètres entre Paris et Beaune et 155 kilomètres entre Beaune et Paris. Donnez le code java nécessaire pour créer cette structure dans une méthode `main`.
2. Ecrivez une méthode à ajouter dans la classe Route permettant de calculer la distance totale de la route (pas seulement dans l'exemple de Paris-Lyon, mais pour toute route représentée avec la classe Route). Cette méthode pourra être récursive ou itérative au choix.