Mohamed Lokbani

IFT1170 - Final - Solution

Nom:__________________________________| Prénom(s):____________________________|

Signature:______________________________| Code perm:____________________________|

Date:26 juillet 2001

Durée: 3 heures (de 18h30 à 21h30) Local: Z-110

Directives:

- Toute documentation permise.
- Calculatrice (simple) permise.

- Répondre directement sur le questionnaire.

1.____________________/13

2.____________________/16

3.____________________/14

4.____________________/15

5.____________________/20

6.____________________/22

Total:__________________/100

Question 1 (13 points)

Soit le programme suivant:

```
// Fichier exo1.java

public class exo1 {

    public static void QuefaisJe(int nombre,int rapport) {

       System.out.print(nombre % rapport);
       int valeur = nombre/rapport;

       if (valeur > 0)
           QuefaisJe(valeur,rapport);
       else
           System.out.println("");
    }

    public static void main(String [] args) {

       if (args.length == 2 )
           QuefaisJe(Integer.parseInt(args[0]),Integer.parseInt(args[1]));
       else
         if (args.length == 1 )
           QuefaisJe(Integer.parseInt(args[0]),10);
       else QuefaisJe(34,10);
    }

}
```

-Q1.1-

Que fait la méthode **QuefaisJe**?

Affiche de manière récursive, sur la même ligne, le reste de la division de valeur/rapport, tant que la variable valeur est supérieure à zéro.

Exemple: On veut faire le calcul suivant: QuefaisJe(45,10);

45/10 ; Résultat = 4 et le reste est 5.

affichage intermédiaire : 5

Résultat /10 = 4/10 ; Résultat = 0 et le reste est 4.

affichage intermédiaire : 4

Résultat = 0 on arrête l'opération.

Affichage final

54

-Q1.2-

Que va afficher le programme suite aux 3 appels suivants:

Appel	Affichage
`java exo1 39 5`	421
`java exo1 39`	93
`java exo1`	43

-Q1.3

Écrire une méthode itérative qui fait la même chose que la méthode récursive précédente.

```
public static void QuefaisJe(int nombre,int rapport) {

        System.out.print(nombre % rapport);
        int valeur = nombre/rapport;

        while (valeur > 0){

              System.out.print(valeur % rapport);

              valeur = valeur/rapport;

        }
        System.out.println("");



  }
```

Question 2 (16 points)

Soit les déclarations suivantes:

```
// Fichier exo2.java

class A {

    String s1;
    private String s2;
    protected void affiche(){
      System.out.println(s1);
    }

}

class B extends A {

    String s3;
    protected void affiche(){
      System.out.println(s3);
    }

}


class C extends A {}


public class exo2 {

    public static void main (String [] args) {


      /*

        on ajoute un seul bloc à la fois

      */

    }

}

// Fin du fichier
```

Si on ajoute dans la méthode main chacun des blocs suivants (un à la fois), indiquez si les lignes en gras sont correctes ou bien elles provoquent des erreurs. Nous entendons par correct: La syntaxe est correcte et l'exécution se fait sans erreur. Indiquez aussi l'affichage obtenu en sortie, dans le cas où la ligne doit afficher un résultat (bien sur si elle est correcte).

Blocs	Erreur	Correct	Résultat
A ref; **ref.affiche();** // pas d'objet instancié!	 X		 - pas d'aff. -
A ref4 = new A(); **ref4.s1 = "toto";** **ref4.s2 = "tata";** // private	 X	 X	
A ref1; B ref2; ref1 = new A(); **ref2 = (B) ref1;** // cast impossible	 X		
A ref3; **ref3 = new B();**		 X	
B ref5 = new B(); **ref5.s1 = "tata";** **ref5.affiche();** **ref5.s3 = "toto";**		 X X X	 --- null ---
A ref7; C ref8 = new C(); **ref7 = (A) ref8;** **ref7.s3 = "tata";** // s3 dans B	 X	 X	
C ref10 = new C(); ref10.affiche(); **((A) ref10).s1="toto";** **((A) ref10).affiche();**		 X X X	 --- null --- --- toto ----

Question 3 (14 points)

Soit le contenu complet du fichier **`Test.java`**

```
/*
 fichier Test.java
 attention: Le main n’est pas obligatoire pour la compilation.
*/
interface Interface1 {
    int i = 1;
    int j = 2;

    void f();
    void g();
}
interface Interface2 {
    int i = 1;
    int j = 2;

    void g();
    void h();
}
abstract class Test implements Interface1, Interface2 {
    public void f() {}
    public void g() {}
}

// Fin
```

L’instruction suivante **`javac Test.java`** provoque-t-elle des erreurs à la compilation?

(<u>encerclez</u> la réponse correcte, et expliquez pourquoi)

Oui, elle provoque des erreurs à la compilation?
Non, elle ne provoque pas d'erreurs à la compilation?

Pourquoi?

La classe Test est abstraite, elle n'est pas obligée de définir les méthodes f & g.

Dans la classe Test, il y a la méthode g, qui est partagée par deux interfaces. Puisque la signature de la méthode g dans Test correspond aux deux qui se trouvent dans les deux interfaces, donc pas d'erreur à ce niveau.

Pour les champs données, i et j, des deux interfaces. La classe Test ne mentionne aucun des deux explicitement. De ce fait les deux seront disponibles dans la classe qui va dériver de Test. S'il y a appel à l'un deux, il faut que cet appel se fasse de manière explicite en mentionnant le nom de l'interface: Interface1.i ou Interface2.i ; Interface1.j ou Interface2.j ; sinon nous serons en présence d'une ambiguïté à la compilation. Chose qui ne se produira pas avec le fichier Test.java, vu qu'il n'y a eu aucun appel à ces champs données.

Question 4 (15 points)

Écrire un programme (**exo4.java**) qui permettra de faire ce qui suit:

-1- lire avec un tampon à un flux texte d'entrée, un fichier donné comme argument au programme,
-2- compter le nombre de lignes dans ce fichier,
-3- et finalement, afficher le résultat sous le format suivant:

```
Nom du fichier: mettre_nom_du_fichier_ici
Nombre de lignes: mettre_nombre_de_lignes_ici

1 contenu_de_la_ligne_1
2 contenu_de_la_ligne_2
3 contenu_de_la_ligne_3

etc.
```

<u>Réponse</u>

```
import java.io.*;

class exo4
{

    public static void main(String[] args) throws IOException
    {

        if (args.length != 1 ) {
                System.err.println("Nombre d'arguments est incorrect: "
                                                     + args.length + " != 1");
                System.exit(1);
        }

        BufferedReader fin = new BufferedReader(new FileReader(args[0]));

        System.out.println("Nom du fichier: " + args[0]);

        int lineno = 0;
        String line = "";

        String interm = "";
        while ( (interm = fin.readLine()) != null )
            line += (++lineno + " " + interm + "\n");


        fin.close();


        System.out.println("Nombre de lignes dans le fichier: " + lineno);

        System.out.print(line);

        System.exit(0);

    }
}
```

Question 5 (20 points)

Soit le programme suivant:

```
 1        import java.applet.applet;
 2        import java.awt.*;
 3
 4        public class exo5
 5        {
 6           private final int LARG_MAX = 300;
 7           private final int NBRE_ANN = 5;
 8           private final int LARG_ANN = 25;
 9
10           public void paint (Graphics page)
11           {
12              int x = 0, y = 0, diameter;
13
14              setBackground (Color.blue);
15
16              diameter = LARG_MAX;
17              page.setColor (Color.white);
18
19              for (int count = 0; count < NBRE_ANN; count++)
20              {
21                 if (page.Color() == Color.black)
22                    page.setColor (Color.white);
23                 else
24                    page.setColor (Color.black);
25
26                 page.fillRect (x, diameter, diameter);
27
28                 diameter -= (2 * LARG_ANN);
29                 x += LARG_ANN;
30                 y += LARG_ANN;
31              }
32
33              page.setColor (red);
34              page.fillRect (x, diameter, diameter);
35           }
36        }
```

-Q5.1- 6 erreurs de syntaxe (pas de logique) se ont glissées dans le programme précédent (pas besoin de détailler).

Ligne	**Erreur**	**Correction**
1	`java.applet.applet;`	`java.applet.Applet;`
4	`public class exo5`	`public class exo5 extends Applet`
21	`if (page.Color() == Color.black)`	`if (page.getColor() == Color.black)`
26	`page.fillRect (x, diameter, diameter);`	`page.fillRect (x,y diameter, diameter);`
33	`page.setColor (red);`	`page.setColor (Color.red);`
34	`page.fillRect (x, diameter, diameter);`	`page.fillRect (x,y diameter, diameter);`

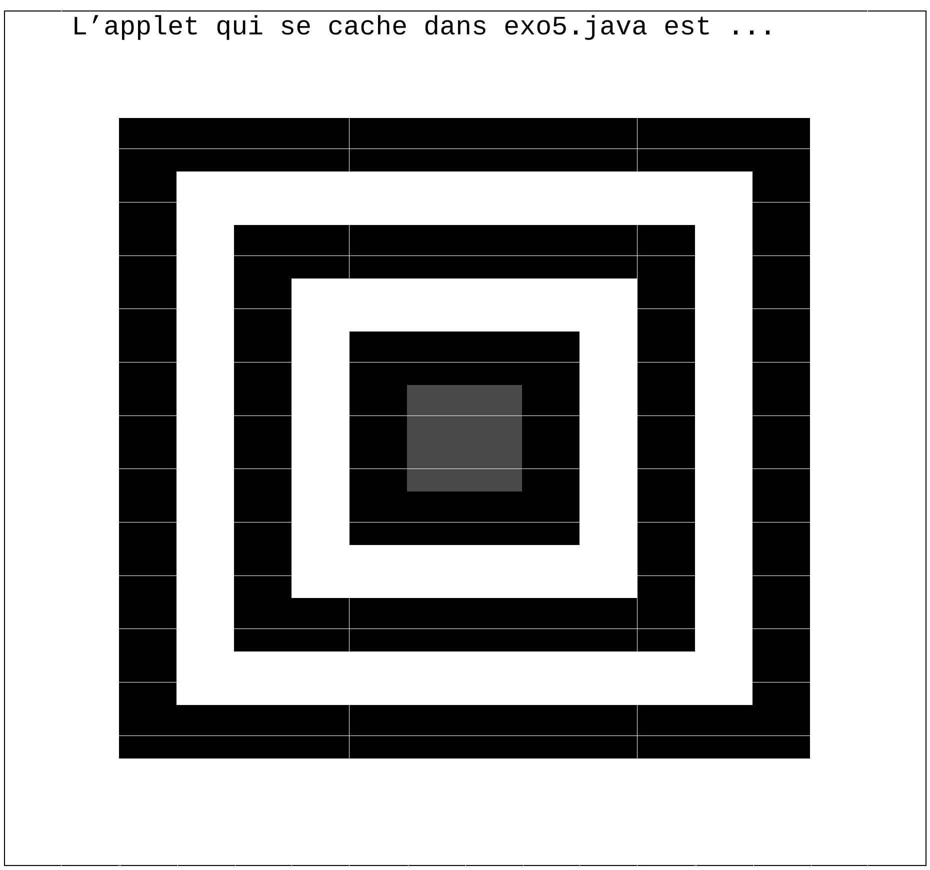
-Q5.2-

Soit le fichier **`exo5.html`** suivant:

```
<! exo5.html>
<HTML>
      <HEAD>
            <TITLE>exo5 ; final H-2001</TITLE>
      </HEAD>
      <BODY>
            <center>
                  <H3>L'applet qui se cache dans exo5.java est ...</H3>
                  <APPLET CODE="exo5.class" WIDTH=300 HEIGHT=300>
                  </APPLET>
                  <HR>
            </center>
      </BODY>
</HTML>
```

Qu'est-ce que vous allez obtenir comme résultat après avoir chargé dans votre navigateur le fichier **`exo5.html`**? (si c'est un dessin, dessinez-le).

__**Réponse:**__ L'applet est centrée dans le cadre de la page HTML. Au départ nous allons avoir un fond bleu du cadre dont les dimensions sont 300x300. Par la suite on dessine en réalité des carrés, en changeant en alternance la couleur du pinceau: tantôt noire, tantôt blanche. Puis dès qu'on arrive au centre du carré, cette couleur est rouge.



Question 6 (22 points)

Tout votre programme sera écrit dans le fichier **`exo6.java`**

- Définir une interface **`Fonction`** qui a une seule méthode **`Calcul`**.
La méthode **`Calcul`** accepte comme argument un **`int`** et retourne un **`int`**.

- Définir la classe **`Mi`** qui implémente **`Fonction`**. L'implémentation de la méthode **`Calcul`** consiste à diviser (division entière) l'argument par 2.

- Définir la classe **`affiche`** qui implémente **`Fonction`**. L'implémentation de la méthode **`Calcul`** consiste à afficher sur la sortie standard l'argument et à le retourner.

- Définir une méthode **`TestMaFonction`**, dans la classe **`exo6`** qui prend comme arguments:

- un tableau d'entiers et qui retourne un tableau qui a la même taille que celui passé en paramètre mais ses valeurs ont été toutes divisées par 2 par la méthode **`Calcul`**.
- un objet du type **`Fonction`**.

- Définir la fonction **`main`** qui doit réaliser ce qui suit:

L'exercice consiste à développer un programme qui:

-1- génère une série de 10 nombre aléatoire entre 0 et 200, comme suit:

On considère un générateur de nombre aléatoire spécifique (on n'utilise pas la classe Random), utilisant la formule:

r = ((r*25173) + 13849)%65536

Le programme commence avec une valeur initiale de r (par exemple, r=0), et à chaque itération, la formule est utilisée avec, comme valeur de r, le nombre aléatoire obtenu à l'itération précédente. La formule produit un nombre dans l'intervalle [0,65535]. Étant donné que l'on veut des nombres aléatoire dans l'intervalle [0,200], une conversion doit être effectuée.

-2- crée une instance de **`Fonction`**, du type **`affiche`**.

-3- appel la méthode **`TestMaFonction`** avec le tableau d'entiers crée aléatoirement et la méthode **`affiche`**.

-4- appel la méthode **`TestMaFonction`** avec le tableau d'entiers crée aléatoirement et une instance de la méthode **`Mi`**.

-5- appel la méthode **`TestMaFonction`** avec comme paramètre le tableau d'entiers obtenu en 4 et la méthode **`affiche`**.

```java
interface Fonction  {
    public int Calcul(int arg);

}

class Mi implements Fonction {
    public int Calcul(int arg) {
      return arg/2;
    }
}

class Affiche implements Fonction {
    public int Calcul(int arg) {
      System.out.println(arg);
      return arg;
    }
}

public class exo6 {

    public static void main(String [] args) {

      int[] myArr = new int[10];
      int valeur=0;

      for (int i=0;i<10;i++) {
          valeur = nbAleatoire(valeur);
          myArr[i] = valeur;
      }
      Fonction print = new Affiche();


      TestMaFonction(myArr,print);
      myArr = TestMaFonction(myArr, new Mi());
      TestMaFonction(myArr, print);
    }


    public static int[] TestMaFonction(int[] arrIn, Fonction func) {

      int length = arrIn.length;

      int[] arrOut = new int[length];

      for (int i=0; i<length; i++)
          arrOut[i] = func.Calcul(arrIn[i]);

      return arrOut;

    }

    static public  int nbAleatoire(int valeur) {
      return ((((valeur * 25173)+13849)%65536)%200);

    }

}
```