UNIVERSITÉ LE HAVRE

UFR
Sciences et
Techniques

Licence 2

Examen de Spécialisation Informatique

Mercredi 24 Mai 2017
Durée : 3 heures
Documents, téléphones et calculatrices non autorisés.
Il est possible d'utiliser les réponses à une question non traitée pour résoudre les autres questions.

I *Fichier et collection*

I.1 Écrire un programme qui lit un fichier texte et le stocke dans une collection dont le nom est spécifié au niveau de la ligne de commande. Le programme doit afficher un certain nombre de lignes aléatoirement, le nombre est spécifié par le deuxième argument de la ligne de commande Écrire le programme de façon à ce que le dimentionnement initial de la collection soit le plus juste possible.
Indication : pour déterminer le nombre de ligne approximatif vous utiliserez `java.io.File.length`, que vous diviserez par la taille moyenne d'une ligne (50 p.e.).

II *Lambda*

On considère le code suivant qui calcule le minimum des valeurs d'un tableau :

```
public static int min(int[] values) {
    if (values.length == 0) {
      throw new IllegalArgumentException();
    }
    int min = values[0];
    for(int i = 1; i < values.length; i++) {
      min = Math.min(min, values[i]);
    }
    return min;
  }
```

II.1 Écrire le code (une méthode) permettant de calculer le maximum.

II.2 Transformer le code de la méthode `min` pour utiliser une lambda.

II.3 Utiliser une référence de méthode plutôt qu'une lambda.

III *Exceptions*

```
class Exemple {

    /*
      Expliquer pourquoi ce code ne compile pas
    */
```

25 Rue Philippe Lebon
B.P. 540
76058 Le Havre Cedex

```java
public void m1() {
        foo();
}

public int foo() throws Exception {
        throw new Exception();
}
//-------------------------------------
/*
  Expliquer pourquoi ce code n'est pas consid r  comme c
*/
public void m2() {
        try {
                // Un peu de travail ...
        } catch (Exception e) {

        }
}
//-------------------------------------
/*
  Expliquer pourquoi ce code ne compile pas
*/
public void m3() {
        try {
                // Un peu de travail ...
        } catch (Exception e) {

        } catch (NullPointerException e) {

        }
}
//-------------------------------------
/*
  Expliquer pourquoi ce code ne compile pas
*/
public void m4() {
        throw new CustomCheckedException();
}

private class CustomCheckedException extends Exception {

        private static final long serialVersionUID = -7944

        public CustomCheckedException() {
                //nothing
        }

}
//-------------------------------------
```

```
        /*
          Expliquer pourquoi ce code ne compile pas
        */
        public int m5() {
                int age;
                String s = "24";
                try {
                        age = getAccessCode();
                } catch (IllegalAccessException e) {
                        e.printStackTrace();
                }
                return age;
        }

        public int getAccessCode() throws IllegalAccessException {
                throw new IllegalAccessException();
        }
        //-------------------------------------
        /*
          Expliquer pourquoi ce code COMPILE
        */
        public void m6() {
                bar();
        }

        public int bar() {
                throw new RuntimeException();
        }
}
```

III.1 Fournir les explications pour chaque cas.

Annexe

```
for (Personne p : liste) {
    if (p.getGenre() == Personne.Sexe.MALE) {
        System.out.println(p.getNom());
    }
}
```

`File(String pathname)`

Creates a new File instance by converting the given pathname string into an abstract pathname.

`long length()`

Returns the length of the file denoted by this abstract pathname.