Name/Nom : .....

Klasse/Classe: .....

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

TEST SEMESTRIEL – SEMESTERPRÜFUNG

SOLUTION

Informatique 1 | Informatik 1

Anweisung / Consigne :

*Lesen Sie die Fragen gut durch und beantworten Sie diese **leserlich** auf den Aufgabenblättern. Für diese Prüfung dürfen Sie 2 Blätter (mit Vor- und Rückseite) mitnehmen, jedoch keine elektronischen Hilfen.*
Tipp: Verlieren Sie bei einzelnen Fragen nicht zu viel Zeit. Beantworten Sie zuerst die Fragen, die Ihnen keine Probleme stellen, und kommen Sie später auf die für Sie schwierigeren Fragen zurück. Die Skala ist unverbindlich

Lisez attentivement la donnée et répondez de manière **lisible** aux questions. Vous avez droit pour cet examen à un aide-mémoire de 4 pages (2 feuilles recto-verso). Aucun moyen électronique n'est permis.
Un conseil : ne restez pas bloqués sur une question. Répondez tout d'abord aux questions avec lesquelles vous êtes à l'aise et revenez ensuite aux questions posant problème. Le barème indiqué est indicatif.

Question	Points	Bonus	Score
Short questions	10	0	
Working with strings	12	0	
Home Lighting Automation	6	0	
Some functions	6	0	
Des petits tableaux	8	0	
Gotta catch 'Em all	8	0	
Total:	50	0	

This exam has 6 questions, for a total of 50 points.

Rev 1.01

Question 1 – Short questions (10 points)

Cette question est séparée en plusieurs exercices indépendants. Le nombre de points pour chaque exercice est indiqué dans la marge.

[2 Pt] (a) Soient les déclarations suivantes :

```
int a = 5;
int b = 6;
double c = 7.0;
double d = 8.0;
String e = "karaoke";
```

Donnez le **type** et la **valeur** des expressions suivantes :

1) `a+b`

1) **int, 11**

2) `a/b`

2) **int, 0**

3) `a + b + e`

3) **String, "11karaoke"**

4) `e.charAt(e.length()-3) + "" + e.charAt(0)`

4) **String, "ok"**

[2 Pt] (b) Soit le code suivant:

```
1  class Foo {
2    public static int a;
3    private int b;
4
5    public static void fun1()
6    {
7      // Loc1
8    }
9  }
10
11 class Bar {
12   private static int c;
13   public int d;
14
15   private void fun2()
16   {
17     Foo inst1 = new Foo();
18
19     // Loc 2
20   }
21 }
```

Si les instructions suivantes sont écrites à la position `Loc1`, sont-elles valides ?

`b = 3;`

True ☐ | *False* ⊗

`a = 4;`

True ⊗ | *False* ☐

`Bar.d = 2;`

True ☐ | *False* ⊗

Même question pour la position `Loc2` :

	True	*False*
`d = 3;`	⊗	☐
`c = 4;`	⊗	☐
`inst1.b = 5;`	☐	⊗

[2 Pt] (c) Cochez **la (ou les)** affirmations correctes (il peut donc y avoir plus d'une réponse).

1) `bar = ' ' + 12.0;` est une instruction possible si
   - ☐ `bar` de type `String`
   - √ **`bar` est déclaré comme `static double`**
   - ☐ `bar` est `final`
2) Après l'exécution de `String[] x = new String[10];`
   - ☐ `x` contient un mot de 10 lettres
   - √ **`x` peut contenir au plus 10 chaînes de caractères**
   - √ **`x` est un tableau vide qui contient `null` dans chaque case**
3) `Foo.bar` est valide si
   - √ **`bar` est statique**
   - ☐ `Foo` est une classe privée
   - ☐ `bar` est une instance de `Foo`
4) Un objet
   - √ **peut contenir d'autres objets**
   - √ **est une instance d'une classe**
   - √ **peut être un attribut statique d'une classe**

[4 Pt] (d) Vrai ou faux ?

	True	*False*
Soit la déclaration suivante `public int bar()` se trouvant dans la classe `Bar`. Il est possible d'appeler cette méthode avec l'instruction `Bar.bar()`.	☐	⊗
Il est possible de faire une boucle `for` infinie.	⊗	☐
Un tableau possède une taille variable.	☐	⊗
Un constructeur peut retourner une valeur.	☐	⊗
`(new Seat[10])[0] != null`	☐	⊗
Il est possible de déclarer deux méthodes ayant le même nom, le même type et le même nombre de paramètres pour autant que le type de la valeur de retour soit différent.	☐	⊗
Pour pouvoir utiliser `FunGraphics` correctement, il est nécessaire de créer une instance de cette classe.	⊗	☐

	True	*False*
Lorsque l'on souhaite connaître la taille d'un tableau, on appelle la méthode `length` appartenant à la classe tableau.	☐	⊗

Question 2 – Working with strings (12 points)

[4 Pt] (a) Écrivez la fonction `camelCase` qui prend un `String` en entrée et renvoie un `String`. La valeur retournée doit avoir une majuscule au début ainsi qu'après chaque espace du `String` d'entrée. Ces espaces doivent être absents dans la sortie alors que les autres lettres ne sont pas touchées. Par exemple :

```
camelCase("camel␣case␣4␣wikipedia") → "CamelCase4Wikipedia"
camelCase("HTTP␣error␣␣list") → "HTTPErrorList"
```

Solution:

```
static String camelCase(String input) {
    String output = "";
    boolean last_was_space_or_first = true;

    for (int i = 0; i < input.length(); i++) {
        if (input.charAt(i) == ' ') {
            last_was_space_or_first = true;
        } else if (last_was_space_or_first) {
            output += input.toUpperCase().charAt(i);
            last_was_space_or_first = false;
        } else {
            output += input.charAt(i);
            last_was_space_or_first = false;
        }
    }

    return output;
}
```

[3 Pt] (b) Écrire une fonction qui double toutes les voyelles d'une chaîne de caractères (les voyelles étant les lettres `a`, `e`, `i`, `o`, `u`, `y`). Avant de doubler les voyelles, faites en sorte de convertir toutes les lettres de la chaîne en minuscules. Attention, cette conversion en minuscules doit se faire en une seule instruction !

Solution:

```java
// Direct solution
static String doubleVowels1(String input)
{
  String in = input.toLowerCase();
  String output = "";

  for(int i = 0 ; i < input.length(); i++)
  {
    char c = input.charAt(i);
    switch (c)
    {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
    case 'y':
      output += c;
    default:
      output += c;
    }
  }

  return output;
}

// Alternative solution
static String doubleVowels(String input)
{
    return input.toLowerCase()
            .replace("a", "aa")
            .replace("e", "ee")
            .replace("i", "ii")
            .replace("o", "oo")
            .replace("u", "uu")
            .replace("y", "yy");
}
```

Solution:

[5 Pt] (c) Perceval le Gallois n'est pas très malin, et il le sait. Il a par contre une botte secrète: à chaque fois qu'il ne comprend pas un mot dans une phrase, il dit "C'est pas faux". Pour faire comme lui, écrivez une méthode qui permet d'automatiser ce genre de réponse. Votre méthode reçoit en paramètre le vocabulaire connu de Perceval (sous forme d'un tableau de `String`) ainsi qu'une phrase sous la forme d'un `String`. La méthode doit retourner "C'est pas faux" si l'un des mots de la phrase n'est pas dans le vocabulaire. Si Perceval connaît tous les mots, la méthode retourne un `String` vide. Par exemple :

```
String[] voc = {"vous", "nous", "plus", "c'est", "les", "pas", "cotelettes", "une", "savez",
    "tavernier", "savoureux"};

perceVoc(voc, "Vous savez tavernier c'est pas une sinécure"); // Retourne "c'est pas faux"
perceVoc(voc, "Les cotelettes c'est plus savoureux"); // Retourne une chaine vide
```

Pour vous aider, vous pouvez utiliser la méthode `split(String seq)` de la classe `String` qui, à partir d'un `String` original, fabrique un tableau de `String` séparés par la séquence donnée en parenthèse. Par exemple, `"je,suis,content".split(",")` retourne le tableau `{"je", "suis", "content"}`. Si vous ne trouvez pas comment faire pour séparer les mots de la phrase, changez le prototype de manière à ce que la méthode reçoive plutôt deux tableaux de `String` et travaillez avec cela.

Solution:

```
static String vocPerceval1(String[] voc, String phrase) {
  String result;
  String[] ph = phrase.split(",");
  String concat = "";

  for (int i = 0; i < voc.length; i++) {
    concat += voc[i].toLowerCase();
  }

  for (int i = 0; i < ph.length; i++) {
    if (!concat.contains(ph[i].toLowerCase())) {
      return "C'est pas faux";
    }
  }

  return "";
}

// Solution alternative
static boolean contained(String s, String[] a) {
  for (int i = 0; i < a.length; i++)
    if (a[i].equalsIgnoreCase(s))
      return true;
  return false;
}

static String vocPerceval2(String[] voc, String phrase) {
  String result;
  String[] ph = phrase.split(",");

  for (int i = 0; i < ph.length; i++) {
    if(!contained(ph[i], voc))
      return "C'est pas faux";
  }

  return "";
}
```

Question 3 – Home Lighting Automation (6 points)

Afin de mettre à profit vos nouvelles compétences en informatique, vous souhaitez ajouter un peu de domotique chez vous. Pour ce faire, vous décidez de réaliser un programme permettant d'automatiser l'éclairage dans une pièce.

Écrivez une classe `HomeAutomation` contenant les attributs nécessaire à la gestion de cette automation. Notamment, vous souhaitez avoir les états possibles pour la lumière : `OFF`, `CHILL`, `READ`, `EAT`. Deux interrupteurs poussoir déclenchent les méthodes suivantes :

1. `void button1()`, permet d'éteindre la lumière mais pas de l'allumer.
2. `void button2()`, lorsque le système est allumé ce bouton permet de changer de mode selon la séquence ci-dessous (qui n'est pas un diagramme d'état). Lorsque le système est éteint, ce bouton sert à rallumer le système qui se retrouve alors dans l'état qu'il avait lorsqu'on l'a éteint.

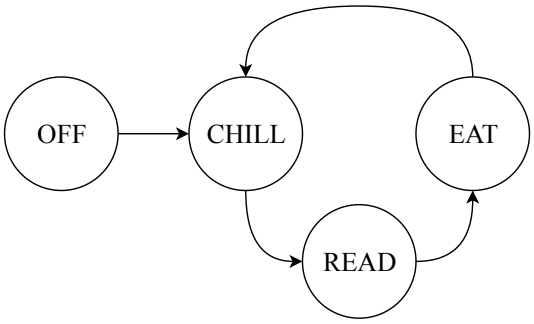

Figure 1 – Diagramme de séquence deuxième bouton

Lors de la création des instances de cette classe, il doit être possible de spécifier l'état dans lequel se trouve. Si aucun état n'est spécifié à la création de l'instance, 'éclairage doit se trouver en mode `CHILL`.

Ajoutez également *tout le code nécessaire* afin que l'exécution du main donne le même résultat que ci-dessous.

```
public static void main(String args[]) {
    HomeAutomation hal = new HomeAutomation();

    System.out.println(hal); → affiche sur la console "Current state : CHILL"
    hal.button1();
    System.out.println(hal); → affiche sur la console "Current state : OFF"
    hal.button1();
    System.out.println(hal); → affiche sur la console "Current state : OFF"
    hal.button2();
    System.out.println(hal); → affiche sur la console "Current state : CHILL"
    hal.button2();
    System.out.println(hal); → affiche sur la console "Current state : READ"
    hal.button1();
    System.out.println(hal); → affiche sur la console "Current state : OFF"
    hal.button2();
    System.out.println(hal); → affiche sur la console "Current state : READ"
    hal.button2();
    System.out.println(hal); → affiche sur la console "Current state : EAT"
    hal.button2();
    System.out.println(hal); → affiche sur la console "Current state : CHILL"
}
```

Solution:

```
public class HomeAutomation {
  enum Ambiance {
    OFF, CHILL, READ, EAT
  };

  public HomeAutomation() {
    this(Ambiance.CHILL);
  }

  public HomeAutomation(Ambiance starting_state) {
    state = starting_state;
  }

  public Ambiance state;
  public Ambiance old_state;

  void button1() {
    if (state != Ambiance.OFF) {
      old_state = state;
    }

    state = Ambiance.OFF;
  }

  void button2() {
    switch (state) {
    case CHILL:
      state = Ambiance.READ;
      break;
    case EAT:
      state = Ambiance.CHILL;
      break;
    case OFF:
      state = old_state;
      break;
    case READ:
      state = Ambiance.EAT;
      break;
    }
  }

  public String toString() {
    return "Current state: " + state;
  }
}
```

Question 4 – Some functions (6 points)

[2 Pt] (a) La fonction `double Math.random()` permet de générer un nombre aléatoire entre $[0, 1[$. Écrivez une fonction nommée `myRand` qui fait usage de cette fonction afin de générer un nombre aléatoire compris entre $[a, b[$. On suppose que a et b sont des `double` et que $a < b$.

Solution:

```java
/**
 * @param a a double number < b
 * @param b a double number > a
 * @return a random number between a and b
 */
public static double myRand(double a, double b){
    return a + (b - a) * Math.random();
}
```

[2 Pt] (b) Passionné d'histoire un peu monomaniaque, vous vous intéressez à une décennie particulière par siècle, à savoir les années 40. Par exemple, les années 1941, 845, 2443 sont intéressantes alors que les années 1952, 23, 8765 ne le sont pas.

Écrivez une fonction qui retourne si oui ou non une année – donnée sous forme d'entier – est intéressante. *Conseil :* deux opérations mathématiques simples suffisent.

Solution:

```java
public static boolean interestingYear(int x) {
  int tenth = x % 100 / 10;
  return tenth == 4;
}

// Alternative solution
int t = a % 100;
if(t >= 40 && t < 50)
    return true;
else
    return false;

// Yet another solution style
String s = "" + x;
return s.charAt(s.length()-2) == '4';
```

[2 Pt] (c) Soit la fonction `isPrime(int x)` qui retourne si un nombre passé en argument est un nombre premier ou non. Écrivez une fonction nommée `nbPrimes` qui prend un entier `n` en paramètre et qui retourne combien de nombres premiers se trouvent entre `1` et `n` compris

Solution:

```java
public static int nbPrimes(int n) {
  int s = 0;
  for (int i = 1; i <= n; i++) {
    if (isPrime(i))
      s++;
  }
  return s;
}
```

Question 5 – Des petits tableaux (8 points)

[4 Pt] (a) Il est possible de représenter des nombres binaires sous forme de `String`. Par exemple, `"01011"` représente le nombre décimal 11. Dans cette représentation, le nombre de bits n'est pas forcément donné, c'est-à-dire que `"11"` et `"00000011"` représentent le même nombre.

Écrivez une fonction qui reçoit un tableau de tels nombres en arguments et qui retourne la position du premier nombre du tableau qui est une puissance de deux. Si aucune puissance de deux n'est présente dans le tableau, la fonction retournera alors -1.

Solution:

```
1  public static int containsPower2(String[] t) {
2    for (int i = 0; i < t.length; i++) {
3      int nOnes = 0;
4      String s = t[i];
5
6      // Count n of 1 in current word
7      for (int j = 0; j < s.length(); j++) {
8        char c = s.charAt(j);
9        if (c == '1')
10         nOnes++;
11     }
12
13     if (nOnes == 1)
14       return i;
15   }
16
17   return -1;
18 }
```

[4 Pt] (b) Écrivez une fonction qui sépare les nombres pairs et impairs dans un tableau reçu en argument. Votre fonction doit retourner un tableau contenant les nombres pairs au début du tableau et les nombre impairs à la fin du tableau. L'ordre des nombres n'est pas important.

Si le tableau contient uniquement des nombres pairs ou impairs, la fonction doit en plus de son comportement normal afficher sur la console `Array is monotonous`.

Solution:

```
1  public static int[] evenThenOdd(int[] a) {
2    int b[] = new int[a.length]; // array containing first the even numbers, then the odd
3    int evenCnt = 0, oddCnt = 0;
4
5    // first , sort the even/odd
6    for (int i = 0; i < a.length; i++) {
7      // even number
8      if ((a[i] % 2) == 0) {
9        b[evenCnt] = a[i];
10       evenCnt++;
11     }
12     // odd number
13     else {
14       b[(b.length - 1) - oddCnt] = a[i];
15       oddCnt++;
16     }
17   }
18
19   boolean monotonous = (evenCnt == 0 || oddCnt == 0) && evenCnt + oddCnt != 0;
20
21   if(monotonous)
22       System.out.println("Array is monotonous");
23
24   return b;
25 }
```

Question 6 – Gotta catch 'Em all (8 points)

Écrivez une classe `Pokemon` qui enregistre les caractéristiques suivantes d'un Pokémon: le nom (`name`), le nombre de points de vie (`health`) ainsi qu'un numéro d'identification (`id`). Ces trois paramètres doivent être déclarés comme étant `private`. L'`id` commence à 1 pour le premier Pokémon et est incrémenté lorsqu'un nouveau Pokémon est crée. Votre classe doit posséder deux constructeurs :

- Le premier, prenant deux paramètres permettant de fixer le nom et les points de vie du Pokémon.
- Le second, sans paramètres, fixe le nombre de points de vie à 0 et le nom du Pokémon à `Unknown`.

Ajoutez ensuite trois méthodes :

1. La première, nommée `modify`, permet de changer le nom du Pokémon.
2. La seconde, nommée `hit`, permet de réduire le nombre de points de vie d'un nombre donné.
3. Finalement, ajoutez une méthode `toString` afin d'afficher le Pokémon, en montrant son nom, ses PVs ainsi que son ID. Le résultat sera sous la forme suivante: `Name: X, PV: Y, ID: Z` où `X`, `Y`, `Z` sont remplacés par les valeurs correspondantes du Pokémon. De plus, X est toujours écrit **en majuscules**. Par exemple:

```
1 Pokemon p1 = new Pokemon("Pikachu", 40);
2 System.out.println(p1);
```

Affiche le résultat suivant sur la console : `Name: PIKACHU, PV: 40, ID: 1`.

Solution:

```
1  public class Pokemon {
2    private int health;
3    private String name;
4    private static int id = 1;
5    private int myId;
6
7    Pokemon(String name, int health){
8      this.name =name;
9      this.health = health;
10     myId = id++;
11   }
12
13   public Pokemon() {
14     this("Unknown", 0);
15   }
16
17   public void modify(String name){
18     this.name = name;
19   }
20
21   public void hit(int x){
22     health -= x;
23   }
24
25   public String toString(){
26     return "Name: "+name + ", PV: " + health + ", ID: " + myId;
27   }
28 }
```

Fin|Ende