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	
String functions	7	0	
Static and classes	5	0	
Monte-Carlo estimation	5	0	
Histogram frequencies	6	0	
Students abroad with the MOVE office	17	0	
Total:	50	0	

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

Rev 1.0.1 Ω

Question 1 – Short questions (10 points)

Diese Frage ist in unabhängige Aufgaben unterteilt. Die Anzahl Punkte jeder Aufgabe ist am Rand vermerkt. Cette question est séparée en plusieurs exercices indépendants. Le nombre de point pour chaque exercice est indiqué dans la marge.

[1 Pt] (a) *Was zeigt der nachstehende Code auf der Konsole an?*
Qu'affiche le code suivant sur la console ?

```
1 final String s = "ddpssfdu";
2
3 for(int i = 1; i < s.length(); i+=1){
4   System.out.print((char)(s.charAt(i)-1));
5 }
```

Solution: *correct*

[1 Pt] (b) *Erklären Sie, **wo** man den folgenden Code finden kann und was das **Ziel** dieses Codes ist:* `this(0,0)`
Expliquez **où** dans un code l'on peut trouver la ligne de code suivante et **à quoi** elle sert : `this(0, 0)`.

Solution: This line can only be written in a constructor to call another constructor which takes two parameters, presumably of the `int` type.

[2 Pt] (c) *Geben Sie an, ob die nachstehenden Ausdrücke korrekt sind oder nicht. Wenn sie korrekt sind, bestimmen Sie den **Typ** und den **Wert**. Zur Errinerung, der Wert von 'a' ist 97 und von '0' ist 48.*
Indiquez si les expressions suivantes sont correctes. Si elles le sont, donnez le **type** et la **valeur** des expressions suivantes. Pour rappel, la valeur de 'a' est 97 et celle de '0' est 48.

(a) `'hello'.charAt(1)`

(a) **incorrect**

(b) `'1'+1`

(b) **int, 50**

(c) `! 10 > 4 && true`

(c) **incorrect**

(d) `(double) 9 / 2 / 2 * 3`

(d) **double, 6.75**

[1 Pt] (e) *Welche Ausgabe erzeugt die folgende Schleife (mit Leerzeichen ␣)?*
Que va afficher la boucle suivante sur la console (indiquez également les espaces avec ␣) ?

```
1 int i = 1;
2 int j = 4;
3
4 for (j = 5; i <= 7 || j < 7; i *= 2) {
5   j++;
6   System.out.print(i + " " + j + " + ");
7 }
```

Lösung | Solution:

Solution:
`1 6 + 2 7 + 4 8 +`

[1 Pt] (f) *Geben Sie ein gültiges Beispiel für zwei Methoden, die denselben Namen und dieselbe Anzahl Parameter haben.*
Donnez un exemple valide de deux méthodes ayant le même nom et le même nombre de paramètres.

Solution:

```
1 int foo(int a){return a;}
2 int foo(String a){return 3;}
```

[4 Pt] (g) *Wahr oder falsch?* | Vrai ou faux ?

	True	*False*
Gegeben ist die nachstehende Deklaration `int[] foo = new int[10]`*. Alle Elemente dieser Tabelle enthalten den Wert* `null`. Soit la déclaration suivante `int[] foo = new int[10]`. Tous les éléments de ce tableau contiennent la valeur `null`.	☐	⊗
Eine Methode kann als formales Argument einen Parameter haben, der denselben Namen hat wie ein Attribut der Klasse. Une méthode peut avoir comme argument formel un paramètre qui a le même nom qu'un attribut de la classe.	⊗	☐
`(!true && false) != (!((true && false) || true))` `(!true && false) != (!((true && false) || true))`	☐	⊗
Klassen können Objekte als Attribute haben. Les classes peuvent avoir comme attributs des objets.	⊗	☐
Dieselben Methoden sind verfügbar für eine Zeichenkette (`String`*) und eine Tabelle vom Typ* `char`*.* Les mêmes méthodes sont disponibles pour un `String` et un tableau de `char`.	☐	⊗
Ein Attribut, das als `static private final int` *deklariert ist, ist eine ganze Konstante, von der eine Instanz pro Objekt besteht.* Un attribut déclaré comme `static private final int` est une constante entière dont il existe une instance par objet.	☐	⊗
Es ist nicht möglich, eine Klasse zu instanziieren, die als `static` *deklarierte Attribute enthält.* Il n'est pas possible d'instancier une classe qui contient des attributs `static`.	☐	⊗
Eine statische Methode kann nicht immer den gleichen Wert zurückgeben. Une méthode statique ne peut pas retourner toujours la même valeur.	☐	⊗

Question 2 – String functions (7 points)

[4 Pt] (a) *Eine gültige Mailadresse darf nur ein einziges Zeichen `'@'` enthalten. Vor und nach diesem Zeichen `'@'` müssen andere Zeichen stehen (beachten Sie, dass diese Bedingungen in Wirklichkeit nicht ausreichend sind). Schreiben Sie eine Funktion, die prüft, ob eine als Argument übergebene Adresse gültig ist.*

On considère qu'une adresse email valide doit posséder un et un seul caractère `'@'` qui doit être entouré d'autres caractères avant et après (notez que ces conditions ne sont pas suffisantes en réalité). Écrivez une fonction vérifiant qu'une adresse passée en argument est valide.

Solution:

```
1  public static boolean checkEmail(String email) {
2    int count = 0;
3
4    // @ must be somewhere in the middle, count number of @ as well
5    for (int i = 0; i < email.length(); i++) {
6      if (email.charAt(i) == '@') {
7        count++;
8
9        if (i == 0 || i == email.length() - 1)
10         return false;
11     }
12   }
13
14   // Count must be one
15   return count == 1;
16 }
```

[3 Pt] (b) *Schreiben Sie eine Funktion `doubleDetect`, die angibt, wie viele* Doppelbuchstaben *(d. h. der gleiche Buchstaben zweimal direkt nebeneinander) in einem Wort vorkommen. Beispiel: Für das Wort* Haus *gibt die Funktion 0 zurück, für das Wort* Gruss *gibt die Funktion 1 zurück und für das Wort* Mittlerrolle *gibt die Funktion 3 zurück.*

Écrivez une fonction `doubleDetect` permettant de détecter combien de doublons (c'est-à-dire la même lettre se trouvant deux fois côte à côte) apparaissent dans un mot. Par exemple, pour le mot *lapin* elle doit retourner 0, pour *saucisse* 1 et pour *commissionné* elle retourne 3.

Solution:

```
1  public static int doubleDetect(String s) {
2    int counter = 0;
3    for (int i = 0; i < s.length() - 1; i++)
4      if (s.charAt(i) == s.charAt(i + 1))
5        counter++;
6    return counter;
7  }
```

Question 3 – Static and classes (5 points)

Was zeigt der folgende Code an? | Qu'affiche le code suivant ?

```
1  public class Foo {
2    public int a;
3    public static int b = 0;
4
5    public Foo(int c) {
6      int a = 5;
7      b = a + c;
8    }
9
10   public void bar(int a) {
11     a += b;
12   }
13
14   public void inc() {
15     a = b++;
16   }
17
18   public String toString() {
19     return "" + a + " " + b + " ";
20   }
21
22   public static void main(String[] args) {
23     Foo inst1 = new Foo(4);
24     System.out.println(inst1);
25     inst1.inc();
26     System.out.println(inst1);
27     Foo inst2 = new Foo(2);
28     inst2.bar(4);
29     System.out.println("" + inst1 + inst2);
30     Foo inst3 = new Foo(1);
31     System.out.println("" + inst2 + inst3);
32     inst3.inc();
33     inst3.bar(5);
34     System.out.println("" + inst2 + inst3);
35   }
36 }
```

Solution:

```
1  0 9
2  9 10
3  9 7 0 7
4  0 6 0 6
5  0 7 6 7
```

Question 4 – Monte-Carlo estimation (5 points)

Das Ziel dieser Übung ist es, den Wert von π mit einem Java-Programm statistisch zu schätzen (Monte-Carlo Algorithmus).

Betrachten wir dazu ein Quadrat mit der Seitenlänge 1 und die in diesem Quadrat enthaltene Viertelscheibe mit dem Radius 1, wie in der folgenden Abbildung dargestellt.

Die Fläche des Quadrats ist 1, während die Fläche der Viertelscheibe $\pi/4$ ist. Der Wert von π kann geschätzt werden, indem eine grosse Anzahl von zufälligen Punkten erzeugt wird und überprüft wird, ob sich diese Punkte in der Scheibe befinden.

Schreiben Sie eine Funktion namens `monteCarloPI`. Diese Funktion verwendet einen Eingabeparameter `n`, der bestimmt, wie viele Zufallszahlen erzeugt werden, um π zu schätzen. Diese Funktion gibt eine reelle Zahl aus, die der Pi-Schätzung entspricht. Für Ihren Code müssen Sie `n` Paare `(x,y)` von Zufallspunkten zwischen 0 und 1 erzeugen. Ihre Funktion muss dann die Anzahl Punkte zählen, die sich in der Scheibe befinden (mit Radius 1).

Die Approximation von π ist dann gegeben durch:

$$4 \cdot \frac{\textit{Anzahl Punkt auf der Scheibe}}{\textit{Gesamtzahl der erzeugten Punkte}}$$

Le but de cet exercice est d'estimer statistiquement la valeur de π à l'aide de Java selon une technique dite de *Monte-Carlo*.

Pour cela, considérons un carré de côté 1 et le quart de disque de rayon 1 inscrit dans ce carré, comme illustré dans la figure ci-dessous.

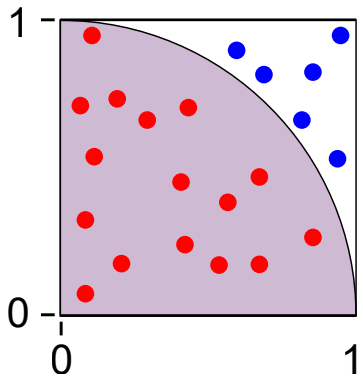
L'aire du carré est de 1 alors que l'aire du quart de disque est de $\pi/4$. En générant un grand nombre de points aléatoires dans le carré et en regardant si ces points sont dans le disque ou non, il est possible d'estimer la valeur de π.

Écrivez une fonction nommée `monteCarloPI`. Cette fonction prend un paramètre `n` en entrée qui détermine combien de nombres aléatoires seront générés pour estimer π. Cette fonction a comme valeur de retour un nombre réel correspondant à l'estimation de π.

Pour votre code, vous devez générer `n` paires `(x,y)` de points aléatoires compris entre 0 et 1 et votre fonction doit ensuite compter le nombre de points se trouvant sur le disque (avec rayon 1).

L'approximation de π est ensuite donnée par :

$$4 \cdot \frac{\text{Nombre de points dans le disque}}{\text{Nombre total de points}}$$



Solution:

```java
class EstimationPI {
  static double monteCarloPI(final int N) {
    double cntOnDisk = 0;

    for (int i = 0; i < N; i++) {
      double x = Math.random();
      double y = Math.random();
      if (x * x + y * y <= 1)
        cntOnDisk++;
    }
    return 4.0 * cntOnDisk / N;
  }

  public static void main(String[] arg) {
    System.out.println(monteCarloPI(100000));
  }
}
```

⚠*Turn page* →

Question 5 – Histogram frequencies (6 points)

In dieser Übung werden wir eine Funktion schreiben, die für die Erstellung von Histogrammen nützlich ist. Ein Histogramm ist ein Diagramm, das die Häufigkeitsdichte für verschiedene Kategorien anzeigt.

Diese Funktion, `countFrequencies` genannt, soll die Auftreten in einem Array zählen und angeben, wie viele Werte in `n` verschiedenen Kategorien zu gleichen Teilen über den Bereich der Eingangswerte verteilt sind. Diese Funktion kann dann verwendet werden (aber nicht in dieser Übung), um ein Histogramm anzuzeigen.

Die Funktion `countFrequencies` erhält ein Array `t` mit reellen Zahlen als Eingabe, eine ganze Zahl `n`, eine untere Grenze `a`, eine obere Grenze `b`. Als Ausgabe gibt sie ein Array zurück, das die Auftreten repräsentiert. `n` definiert die Anzahl der Kategorien (gleicher Grösse) zwischen der unteren und der oberen Grenze.

Anwendungsbeispiel:

Gegeben ist eine Tabelle `t` mit Zahlen $t \in [5, 25[$. Wir definieren `a=5`, `b=25` und `n=5` als die anderen Eingabeparameter.

Der Rückgabewert der Funktion `countFrequencies` ist die Tabelle `[4,5,2,6,1]`. Dies bedeutet, dass die erste Kategorie `[5,9[` vier Elemente umfasst, die zweite Kategorie `[9,13[` fünf Elemente, die Kategorie `[13,17[` zwei Elemente, die Kategorie `[17,21[` sechs Elemente und die Kategorie `[21,25[` 1 Element. Wir stellen fest, dass:

- *es 5 Kategorien gibt (da `n=5`)*
- *die untere Grenze, von der aus wir zählen, 5 ist, einschliesslich der 5 (da `a` = 5)*
- *dass die obere Grenze 25 ist, ohne die 25 (da `b` = 25)*
- *jede Kategorie gleich gross ist (hier 4)*

Dans cet exercice nous allons écrire une fonction utile à la création d'histogrammes. Un histogramme est un graphique montrant, pour différentes catégories, combien d'occurrences se retrouvent dans chaque catégorie.

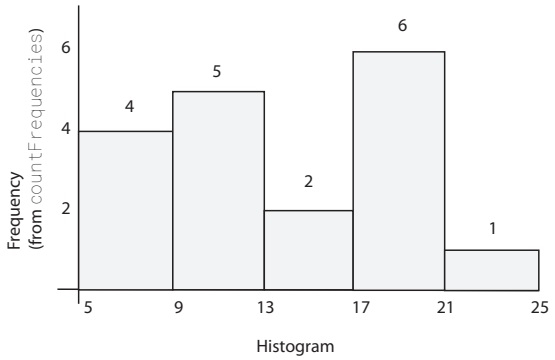
Cette fonction, nommée `countFrequencies`, a pour but de compter des occurrences d'un tableau et de donner combien de valeurs se retrouvent dans `n` différentes catégories réparties également sur l'intervalle des valeurs en entrée. On pourra ensuite (mais pas dans cet exercice) utiliser cette fonction pour afficher un histogramme.

La fonction `countFrequencies` prend un tableau `t` de nombres réels en entrée, un nombre entier `n`, une borne inférieure `a`, une borne supérieure `b`. En sortie, elle retourne un tableau représentant les occurrences. Le `n` définit le nombre de catégories (de même taille) entre l'élément minimal et l'élément maximal à créer.

Exemple d'utilisation :

Soit un tableau avec des nombres $t \in [5, 25[$. On définit `a=5`, `b=25` et `n=5` comme étant les autres paramètres d'entrée. La sortie de la fonction `countFrequencies` est alors le tableau `[4,5,2,6,1]`, signifiant qu'il y a 4 éléments dans la catégorie `[5,9[`, 5 éléments dans la catégorie `[9,13[`, 2 éléments dans la catégorie `[13,17[`, 6 éléments dans la catégorie `[17,21[` et finalement 1 élément dans `[21,25[`. On note que (voir dessin) que :

- Il y a 5 catégories (parce que `n=5`)
- Que la borne inférieure à partir d'où l'on compte est 5, 5 inclus (parce que `a=5`)
- Que la borne supérieure est de 25, 25 exclu (parce que `b=25`)
- Que chaque catégorie a la même taille (ici 4)



Solution:

```
1  static int[] countFrequencies(double[] t, int N, int min, int max) {
2    int[] occ = new int[N];
3    final int LEN = t.length;
4
5    final double CATLEN = (max - min) / N;
6
7    for (int i = 0; i < LEN; i++) {
8      // Lets see in what bin this element belongs
9      for (int j = 0; j < N; j++) {
10       double lowerTick = min + j * CATLEN;
11       double upperTick = min + (j + 1) * CATLEN;
12       if (t[i] >= lowerTick && t[i] < upperTick)
13         occ[j]++;
14     }
15   }
16   return occ;
17 }
```

Question 6 – Students abroad with the MOVE office (17 points)

[4 Pt] (a) *Gegeben ist die nachstehende Klasse* `Student`*, die einen Studenten darstellt.*

```
class Student {
  String firstname, surname, uni, country;
  double grade;
}
```

Wie Sie sehen können, hat ein Objekt dieser Klasse einen Vornamen, einen Nachnamen, eine Nationalität, eine Heimuniversität (oder -Ingenieurschule) (z. B. "Harvard University" oder "MIT") sowie den Notendurchschnitt (ausgedrückt zwischen 0 und 20).

Fügen Sie einen Konstruktor zu dieser Klasse hinzu, um die Werte der verschiedenen Attribute anzugeben. Dieser Konstruktor muss auch überprüfen, ob die Note gültig ist. Wenn dies nicht der Fall ist, ist die Note des Schülers 10. In allen Fällen muss die Note auf den Zehntel genau geschrieben werden, ohne sie auf- oder abzurunden (z. B. 15.678 muss als 15.6 gespeichert werden).

Fügen Sie in der Klasse auch den Code hinzu, der notwendig ist, damit:

```
System.out.println(new Student("Sheldon", "Cooper", "US", "CalTech", 19.554);
```

genau *"*`Student: Sheldon Cooper, 19.5`*" anzeigt.*

Soit la classe `Student` qui représente un étudiant et qui est donnée ci-dessous.

```
class Student {
  String firstname, surname, uni, country;
  double grade;
}
```

Comme vous le voyez, un objet de cette classe a un prénom, un nom de famille, une nationalité, une université (ou école d'ingénieur) d'origine (e.g. "Harvard University" ou "MIT"), ainsi que la moyenne des notes (exprimé entre 0 et 20).

Ajoutez un constructeur à cette classe permettant de spécifier les valeurs des différents attributs. Ce constructeur doit également contrôler que la note est valide. Si tel n'est pas le cas, la note de l'étudiant vaut 10. Dans tous les cas, la note doit être coupée au $10^{\text{ème}}$ sans arrondi (par exemple 15.678 doit être stockée sous la forme 15.6). Ajoutez également le code nécessaire à ce que :

```
System.out.println(new Student("Sheldon", "Cooper", "US", "CalTech", 19.554);
```

affiche **exactement** le texte "`Student : Sheldon Cooper,19.5`"

Solution:

Solution:

```java
public Student(String firstName, String surName, String country, String uni, double grade
    ) {
  this.firstname= firstName;
  this.surname = surName;
  this.country = country;
  this.uni = uni;

  if(grade < 0 || grade > 20) {
    grade = 10;
  }

  // Rounding
  this.grade = ((int) (grade * 10)) / 10.0;
}

public String toString() {
  return "Student : " + firstname + " " + surname + ","+ grade;
}
```

(b) *Schreiben Sie eine Klasse namens* `Move`*. Diese Klasse stellt das Büro für internationale Beziehungen einer bestimmten Universität dar. Diese Klasse hat die (privaten) Attribute des Namens des Büros und der Universität, zu der das Büro gehört. Das dritte Attribut (*`public`*) dieser Klasse ist eine Tabelle der ausländischen Studierenden in ihrem Austauschjahr. Erstellen Sie einen Konstruktor, um diesen Attributen einen Wert zu geben.*

Écrivez une classe nommée `Move`. Cette classe représente le bureau des relations internationales d'une université. Cette classe possèdes les attributs (privés) du nom du bureau et l'université à laquelle le bureau appartient. Le troisième attribut (`public`) de cette classe est un tableau d'étudiants étrangers faisant leur année d'échange. Faites un constructeur permettant de donner une valeur à ces attributs.

Solution:

```java
class Move {
  private String officeName;
  private String uni;
  public Student[] exchangeStudents;

  public Move(String officeName, String uni, Student[] exchangeStudents) {
    this.officeName = officeName;
    this.uni = uni;
    this.exchangeStudents = exchangeStudents;
  }
}
```

[1 Pt] (c) *Fügen Sie eine Methode namens* `changeName` *zu dieser Klasse hinzu, mit der der Name des Büros geändert werden kann.*

Incluez dans cette classe une méthode permettant de changer le nom du bureau nommée `changeName`.

Solution:

```
1  public void changeName(String newName) {
2    this.officeName = newName;
3  }
```

[5 Pt] (d) *Eine zweite in dieser Klasse vorhandene Methode ist die Methode* `numberOfCountries`*. Sie gibt die Anzahl den verschiedenen Nationalitäten im Büro zurück und nimmt keine Eingabeparameter an.* Achtung: *Jedes Land darf nur einmal aufgeführt werden.*

Une deuxième méthode présente dans cette classe est la méthode `numberOfCountries`. Cette méthode retourne le nombre de différentes nationalités présentes dans le bureau et ne prend aucun paramètre en entrée. *Attention :* chaque pays ne doit y figurer qu'une fois.

Solution:

```
1  int numberOfCountries() {
2    int cnt = 0;
3    for (int i = 0; i < exchangeStudents.length; i++) {
4      String ctry = exchangeStudents[i].country;
5      if (i == 0)
6        cnt++;
7      else {
8        int occurences = 0;
9        for (int j = 0; j < i; j++) {
10         String currCountry = exchangeStudents[j].country;
11         if (currCountry.equals(ctry))
12           occurences++;
13       }
14       if (occurences == 0)
15         cnt++;
16     }
17   }
18   return cnt;
19 }
```

[5 Pt] (e) *Die dritte Methode in dieser Klasse heisst* `meanGEP`*. Diese Methode erhält die Abkürzung eines Landes (z. B. "US") als Eingabe und gibt als Parameter den Notendurchschnitt der Austauschstudierenden aus dem jeweiligen Land zurück. Wenn das betreffende Land nicht vertreten ist, muss diese Funktion -1 zurückgeben.*

La troisième méthode de cette classe s'appelle `meanGEP`. Cette méthode prend en entrée l'abbréviation d'un pays (p.ex. "US") et retourne la note moyenne des étudiants d'échange provenant du pays donné en paramètre. Si le pays en question n'est pas représenté, cette fonction doit retourner -1.

Solution:

```java
double meanGEP(String country) {
  final int N = exchangeStudents.length;

  double grd = 0;
  int cnt = 0;
  for (int i = 0; i < N; i++) {
    String currCnty = exchangeStudents[i].country;
    if (currCnty.equals(country)) {
      grd += exchangeStudents[i].grade;
      cnt++;
    }
  }

  if (grd == 0)
    return -1;
  else {
    // round grade to a tenth
    grd /= cnt;
    grd *= 10;
    double remainder = grd % 1;
    if (remainder >= 0.5)
      grd += 1;
    else
      grd = (int) grd;

    return (grd / 10);
  }
}
```

[2 Pt] (f) *Gegeben ist die nachstehende Methode* `main`*:* | Soit la méthode `main` suivante:

```java
public static void main(String[] args) {
    Student s1 = new Student("Sheldon","Cooper","US","CalTech",18.0);
    Student s2 = new Student("Leonard","Hofstadter","US","CalTech",16.2);
    Student s3 = new Student("Jens","Van Acker","BE","KULeuven",15.3);
    Student s4 = new Student("Florian","Luisier","CH","EPFL",16.26);
    Student s5 = new Student("Francois","Aguet","CH","EPFZ",16.0);
}
```

Vervollständigen Sie sie, damit eine Instanz eines Büros für internationale Beziehungen namens "MOVE" erstellt wird, das zur HES-SO Valais-Wallis gehört. Erfassen Sie die vorhandenen Studierenden und sorgen Sie dafür, dass die Anzahl der vorhandenen Nationalitäten sowie der Notendurchschnitt für die Schweizer angezeigt werden.

Ihr Programm muss dann in der Kommandozeile anzeigen, wie viele verschiedene Nationalitäten im Büro MOVE vertreten sind und wie hoch der Notendurchschnitt der Schweizer ist. Nach der Ausführung Ihres Programms sollte das Ergebnis wie in der folgenden Abbildung aussehen.

```
Number of different countries present: 3
Mean grade of swiss students: 16.1
```

Complétez-le de manière à créer une instance d'un bureau de relations internationales nommée "MOVE" qui appartient à la HES-SO Valais-Wallis. Inscrivez-y les étudiants présents et faites en sorte de montrer combien de nationalités sont présentes ainsi que la moyenne pour les suisses.

Votre programme doit ensuite montrer dans la ligne de commande combien de nationalités différentes sont présentes au sein du bureau MOVE ainsi que la note moyenne pour les suisses. Après avoir exécuté votre programme, le résultat doit ressembler à la figure dessous.

```
Number of different countries present: 3
```

Listing continues on next page →

. . . continuing from previous page

```
Mean grade of swiss students: 16.1
```

Solution:

```
Student[] exchange = new Student[5];
exchange[0] = s1;
exchange[1] = s2;
exchange[2] = s3;
exchange[3] = s4;
exchange[4] = s5;

// Alternative Student[] exchange = {s1,s2,s3,s4,s5};

Move exchangeOffice = new Move("MOVE", "HEVS", exchange);
System.out.println("Number of different countries present: " + exchangeOffice.
    numberOfCountries());
System.out.println("Means grade of swiss students: " + exchangeOffice.meanGEP("CH"));
```

Leerseite

◇

Cette page a été laissée vierge intentionnellement

Fin|Ende