Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Informatik I (D-BAUG)

Hermann Lehner, Felix Friedrich

ETH Zürich, 12.08.2019.

Name, Vorname: . . . . .
Legi-Nummer: . . . . .

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte, und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:

1. Dauer der Prüfung: 60 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für von Ihnen gesprochene Sprachen). 4 A4 Seiten handgeschrieben oder $\geq$ 11pt Schriftgrösse.
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen sind deutlich durchzustreichen! Korrekturen bei Multiple-Choice Aufgaben bitte unmissverständlich anbringen!
5. Es gibt keine Negativpunkte für falsche Antworten.
6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.
7. Wir sammeln die Prüfung zum Schluss ein. Wichtig: Stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: Bitte melden Sie sich lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.
8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

1. *Exam duration: 60 minutes.*
2. *Permitted examination aids: dictionary (for languages spoken by yourself). 4 A4 pages hand written or $\geq$ 11pt font size.*
3. *Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.*
4. *Solutions must be written directly onto the exam sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Provide corrections to answers of multiple choice questions without any ambiguity!*
5. *There are no negative points for wrong answers.*
6. *If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.*
7. *We collect the exams at the end. Important: You must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: Please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.*
8. *If you need to go to the toilet, raise your hand and wait for a supervisor.*
9. *We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.*

Question:	1	2	3	4	5	6	Total
Points:	15	10	12	10	10	3	60
Score:							

Aufgabe 1: Java Grundlagen (15P)

Gegeben ist folgender Codeabschnitt. *Consider the following code segment.*

```
5  // PRE: 'digits' contains an arbitrary number of elements
6  // PRE: each number in 'digits' is in the interval [0..9]
7  // POST: ??? (todo)
8  private static int x_sum_rec(int[] digits){
9    int result(X) = 0;            Task a) (1P) only if all 4 spots correct
10   for (int i = 0; i < digits.length; ++i){
11     result(X) += digits[i];
12   }              Task b) (1P) correct idea of casting to int.
13   if (result > 9){          (1P) correct syntax. There can be parentheses.
14     // create an array with just enough cells to hold all digits of 'result'
15     int[ ] result_digits = new int[ (int)Math.log10(result) + 1 ];
16     for (int j = 0; j < result_digits.length; ++j){
17       result_digits[j] = result % 10;
18       result(X) /= 10;
19     }
20     // perform the above computation again on 'result_digits'
21     result(X) = x_sum_rec(result_digits);
22   }
23   return result;
24 }
```

/1P (a) Aufwärmübung: Markieren Sie im Code die Stellen (Einkreisen!), an denen der Wert von `result` geschrieben wird.

Warming-up: Mark the places (Circles!) in the code where the value of `result` is written.

/2P (b) Der Compiler gibt folgenden Fehler aus. Ändern Sie Zeile 15 direkt im Programm oben, um den Fehler zu beseitigen.

The following compiler error is reported. Change line 15 directly in the program above to fix that error.

```
Main.java:15: error: incompatible types: possible lossy conversion from double to int
      int[] result_digits = new int[ (Math.log10(result) + 1) ];
                                                             ^
1 error
```

/2P (c) (i) Was ist der Gültigkeitsbereich der Variable `result_digits`?
(ii) Welchen Wert nimmt Variable `i` während der Ausführung von `x_sum_rec` zuletzt an (solange sie existiert)?

(i) What is the scope of variable `result_digits`?
(ii) What is the last value of variable `i` during an execution of `x_sum_rec` (as long as it exists)?

(i) Zeilen / *Lines* von / *from*: 15 bis / *to*: 21 or 22 (1P)

(ii) ◯ `digits.length-1` √ `digits.length` ◯ `digits.length+1` (1P)

(d) Angenommen wir rufen die Methode `x_sum_rec` mit folgendem Argument auf:

Given we call the method `x_sum_rec` with the following argument:

/5P

digits =

3	1	4	7	1	2
0	1	2	3	4	5

Was sind die Werte der Variablen `result` und `result_digits` beim Erreichen der Zeile 20?

What are the values of the variables `result` and `result_digits` upon reaching line 20?

result = `0 (1P)`

result_digits =

8	1
0	1

```
(1P) correct digits
(1P) correct order
```

Was ist der Rückgabewert der Methode? Achtung! Der Wert der Variable `result` wird auf Zeile 21 noch einmal verändert!

What is the return value of the method? Caution! The value of variable `result` is being changed once more on line 21!

Rückgabewert / *Return-Value*: `9 (2P)`

(e) Die Nachbedingung auf Zeile 7 fehlt noch! Schreiben Sie eine möglichst prägnante Nachbedingung welche den Rückgabewert der Methode so präzise wie möglich beschriebt.
Beschreiben Sie auch das kleinstmögliche Zahlenintervall, in dem der Rückgabewert der Methode immer liegt.

The post-condition on line 7 is still missing! Write a post-condition that is as consise as possible and that describes the return value of the method as precisely as possible.
Also describe the smallest possible number-range in which the return value always lies.

/3P

```
// POST:
(2P) The value is the result of repeatetly computing the cross sum of
the input value or the previous cross sum until only a single digit is
left.
(1P) The return value is in the interval [0..9] (single digit)
```

(f) Zur Zeit werden in der Methodensignatur `int` Datentypen genutzt. Das geht besser!
Was ist der kleinstmögliche Datentyp für den Rückgabewert und das Argument, so dass die angegebenen Wertebereiche darstellbar sind? Schreiben Sie die verbesserte Methodensignatur hin.

At the moment, the method signature uses `int` data types. We can do better! What is the smallest-possible data type for the return value and the argument so that the given number ranges can be represented? Write the improved method signature.

/2P

Methodensignatur:

```
byte x_sum_rec(byte[] digits)
(1P) for using 2 x byte, (1P) for correct syntax
```

Aufgabe 2: Methoden und Rekursion (10P)

/3P (a) **Wie versprochen:** Die folgende Aufgabe ist direkt aus dem Buch kopiert!

Rekursive Ausgabe einer Zahl. Schreiben Sie eine rekursive Methode `print_rec(n)` zur Ausgabe aller Ziffern der positiven ganzen Zahl `n`. Die letzte Ziffer kann mit `n % 10` abgespalten werden, der vordere Teil der Zahl entspricht `n / 10`. Die Methode soll ausser `n` keine zusäzlichen Variablen verwenden. `print_rec(123)` soll nacheinander die Zeichen `1`, `2` und `3` ausgeben.

As promised: *The following task is directly copied from the book!*

Outputting a number recursively. Write a recursive method `print_rec(n)` *to output all digits of a positive integer number* `n`*. The last digit can be split off using* `n % 10`*, the front part of the number corresponds to* `n / 10`*. The method must not use any additional variables besides* `n`*.* `print_rec(123)` *shall output one after the other* `1`*,* `2`*, and* `3`*.*

```
void print(int n){
    if (n > 9) print_rec(n / 10);
    Out.print(n % 10);
}
- termination (1P)
- correct recursive call (1P)
- correct output after recursive call (1P)
```

/7P (b) Das Spiel "Jump It" besteht aus einem Spielbrett mit n positiven Ganzzahlen in einer Reihe von Feldern, wobei das erste Feld immer 0 enthält. Diese Zahlen entsprechen den Kosten, ein Feld zu betreten. Hier ein Beispiel fuer $n = 6$.

The game of "Jump It" consists of a game board with n positive integers in a row of fields, whereas the first field always contains 0. These numbers represent the cost to enter a field. Here is a sample game board where $n = 6$.

board =

0	3	80	6	57	10
0	1	2	3	4	5

Das Ziel des Spiels ist es, vom ersten Feld zum letzten Feld zu gelangen. Es gibt nur zwei mögliche Züge:

- Zum nächsten Feld nach rechts rücken
- Direkt zum übernächsten Feld nach rechts springen

The goal of the game is to get from the first field to the last field. There are only two possible moves:

- *Move to the next field to the right*
- *Jump to the next but one field to the right*

Die Kosten der Reise entsprechen der Summe der besuchten Feld-Kosten und sollen minimiert werden!

The costs of the journey consists of the sum of the costs of the visited fields and must be minimized!

Die günstigste Reise wäre also folgende Felder zu besuchen: 0, 1, 3, 5 (Kosten $0+3+6+10=19$). Eine schlechtere Variante wäre 0, 2, 4, 5 (Kosten $0+80+57+10=147$).

The cheapest journey would be to use the following fields: 0, 1, 3, 5 (costs $0+3+6+10=19$). A worse variant would be 0, 2, 4, 5 (costs $0+80+57+10=147$).

Aufgabe

Vervollständigen Sie die folgende Implementierung des oben beschriebenen Spiels. Die Methode `jump_it` soll für ein gegebenes Spielbrett (Argument `board`) die minimalen Kosten zurückgeben, welche nötig sind, um vom ersten Feld zum letzen zu gelangen.

Tip: `Math.min(a, b)` gibt die kleinere der beiden Zahlen `a` und `b` zurück.

Task

Complement the following implementation of the game described above. The method `jump_it` shall return the minimal costs that are required to get from the first to the last field for a given game board (argument `board`).

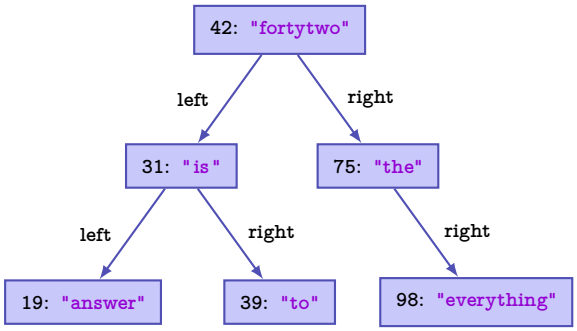
Hint: `Math.min(a, b)` returns the smaller of the two numbers `a` and `b`.

```
public class JumpIt {

  // PRE: board: the game board. Contains positive integers.
  // PRE: pos: the current (zero based) position on the board.
  //       pos must be >= 0.
  // POST: returns the minimal costs from pos to the last field.
  //       If pos is too large, return the cost of the last field.
  int solve(int[] board, int pos){
    if (pos >= board.length - 1 (1P)){
      return board[board.length - 1] (1P);
    }
    return board[pos] + Math.min(
        solve(board, pos + 1),
        solve(board, pos + 2))
    (1P) - Cost of the curret cell
    (1P) - Compute the minimum of something
    (2P) - Recursive calls for pos+1, pos+2
  }

  public static void main(String[] args) {
    JumpIt game = new JumpIt();
    int[] board = {0, 3, 80, 6, 57, 10};
    int min_costs = game.solve(board, 0 (1P));
    Out.println("Minimum costs are " + min_costs);
  }
}
```

Aufgabe 3: Dynamische Datenstrukturen (12P)

Aus der Vorlesung kennen Sie bereits binäre Suchbäume und deren Implementierung in Java. Wir wollen nun Suchbäume verwenden, um einen `Map` Datentypen zu implementieren. Wir wollen Werte nach Schlüsseln ablegen können und effizient wieder nach Schlüssel finden können.
Wir nehmen an, dass Schlüssel immer vom Typ `int` sind und Werte vom Typ `String`.

You already know binary search trees and their implementation in Java. Now we want to use search trees to implement a `Map` *data type. We want to store values assigned to keys and be able to efficiently retreive them again by key.*
We assume that keys are of type `int` *and values are of type* `String`.



Ein Beispiel eines Suchbaumes / *An example of a search tree*

```
public class Node {
  int key;       // Schluessel
  String value; // Wert
  Node left;     // linker Teilbaum
  Node right;    // rechter Teilbaum

  Node(int k, String v) {
    key = k; value = v;
    left  = right = null;
  }
}
```

Der Suchbaum besteht aus Knoten vom Typ `Node` / *The search tree consists of nodes of type* `Node`

/2P (a) Beschreiben Sie genau die Suchbaumeigenschaft. Also die Invariante, welche welche jederzeit erfüllt sein muss.

Describe the search tree property. That is, the invariant that must always hold.

```
Each node has a key.
All keys in the left subtree are smaller.
All keys in the right subtree are bigger.

Grading:
(1P) if partly correct, vague, etc.
(2P) if precisely defined.
```

/10P (b) Vervollständigen Sie die Implementation der Methoden `put` und `get` der Klasse `TreeMap`. Beachten Sie die Vor- und Nachbedingungen.

Complement the implementation of the methods `put` *and* `get` *of class* `TreeMap`. *Mind the pre- and postconditions.*

```
class TreeMap {        Each box is (1P) if 100% correct, (0P) otherwise.
  private Node root;
  // POST: the value is stored in the map an can be accessed by key
  // POST: if key already existed in the map, the value has been replaced
  public void put (int key, String value) {
    if ( root == null ) { // tree is empty
      root = new Node(key, value);
      return;
    }
    Node node = root;
    while (true) {
      if (key == node.key) {
        node.value = value;
        return;
      }
      if (key < node.key) {
        if ( node.left == null ) {
          node.left = new Node(key, value);
          return;
        } else {
          node = node.left;
        }
      } else { // key > node.key
        /* omitted, analogous to key < node.key */
      }
    }
  }
  // POST: yields the value currently assigned to key or null if key doesnt exist.
  public String get(int key) {
    Node node = root;
    while ( node != null ) {
      if (key == node.key){
        return node.value;
      } else if (key < node.key) {
        node = node.left;
      } else {
        node = node.right;
      }
    }
    return null;
  }
}
```

Aufgabe 4: Klassen, Kapselung und Floating Points (10P)

Rationale Zahlen. Eine rationale Zahl besteht aus einem Zähler und einem Nenner, beide vom Typ `int`. Vervollständigen Sie die Implementierung der Klasse `Rational` auf der gegenüberliegenden Seite. Eine Instanz dieser Klasse repräsentiert eine rationale Zahl und bietet diverse hilfreiche Operationen an.
Der folgende Test-Code zeigt die Anwendung von unserem Datentypen `Rational` sowie die erwartete Ausgabe in der Konsole.

Rational Numbers. A rational number consists of a numerator and a denominator, both of type `int`. Complement the implementation of the class `Rational` on the opposite side. An instance of the class represents a ration number and provides several helpful operations.
The following test-code shows the application of our data type `Rational` as well as its expected output in the console.

```
public class Main {
  public static void main(String[] args) {
    Rational a = new Rational(1, 3);
    Rational b = new Rational(2, 4);
    Rational c = a.add(b);          // add two rationals: a + b = c
    double d = 0.833333;            // this is reasonably close to 5/6
    Out.print(c.toString());        // Outputs "5/6"
    Out.print(c.equals(d));         // Compare against a double. Outputs "true"
    Rational b1 = new Rational(1, 2);
    Out.print(b.equals(b1));        // Compare against a rational. Outputs "true"
  }
}
```

Denken Sie bei der Implementierung an die korrekte Handhabung von Fliesskomma-Zahlen. Und vergessen Sie nicht, die Daten zu kapseln. Berücksichtigen Sie alle Nachbedingungen.

When implementing this class, always remember the correct handling of floating-point numbers. And don't forget to encapsulate the data. Take all postconditions into account.

/2P (a) Definieren Sie zwei Felder `n` (Zähler) und `d` (Nenner) vom Typ `int`.

Define two fields `n` (nominator) and `d` (denumerator) of type `int`.

/1P (b) Implementieren Sie den Konstruktor.

Implement the constructor.

/3P (c) Implementieren Sie die Methode `add`.

Implement the method `add`.

Hint: $$\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d + c \cdot b}{b \cdot d}$$

/4P (d) Implementieren Sie die beiden `equals` Methoden.

Implement both `equals` methods.

Hints: $\frac{a}{b} = \frac{c}{d} \Leftrightarrow a \cdot d = c \cdot b$ $\quad |a| \mathrel{\widehat{=}}$ `Math.abs(a)`

```
public class Rational {
  private static final double epsilon = 0.00001;

  // Fields / Task (a)
  private int n; // nominator
  private int d; // denumerator
  (1P) encapsulation (1P) correct names and decl.

  // Reduces this rational (e.g. 1/2 instead of 2/4) (zu Deutsch: Kuerzen)
  private void reduce() { /* implementation omitted */ }

  // POST: The newly created rational contains the value n / d. Does not reduce.
  public Rational(int n, int d) {
    this.n = n;
    this.d = d;
    (1P) correct use of this
  }

  // POST: no changes to the current object or 'other', returns a new rational.
  // containing the *reduced* sum of both rationals.
  public Rational add(Rational other) {
    Rational result = new Rational(n*other.d + other.n * d, d * other.d);
    result.reduce();
    return result;
    (1P) create new object (1P) correct formula (1P) call reduce
  }
  // POST: returns the value of the current object as double.
  public double toDouble() { /* implementation omitted */ }

  // POST: true iff the current object is "equal" to the argument of type double.
  public boolean equals(double d) {
    return Math.abs(d - toDouble()) < epsilon;
    (1P) Correct formula with abs (1P) use epilon
  }
  // POST: true iff the current object is exactly equal to the argument.
  public boolean equals(Rational r) {
    return n * r.d == r.n * d;
    (1P) correct formula (1P) no use of epsilon!
  }
  //POST: Provides a string representation of the form "n/d"
  public String toString() { /* omitted, */ }
}
```

/10P

Aufgabe 5: Vererbung und Polymorphie (10P)

```
abstract class Item {
  String name;
  double price;
  Item(String name, double price){
    this.name = name;
    this.price = price;
  }
  abstract String description();
  void print(){
    Out.println(description() + ": CHF " + price);
  }
}
```

```
abstract class NonFood extends Item {
  NonFood(String name, double price){
    super(name, price);
  }
  String description(){
    return "(NF)";
  }
}
```

```
abstract class Food extends Item {
  String expiresDate;
  Food(String name, double price, String expiresDate){
    super(name, price);
    this.expiresDate = expiresDate;
  }
  String description(){
    return " [exp. " + expiresDate + "] ";
  }
}
```

```
class Banana extends Food {
  Banana(double price, String expiresDate){
    super("Banana", price, expiresDate);
  }
  String description(){
    return name + super.description();
  }
}
```

```
class Stone extends NonFood {
  Stone(){
    super("Stone", 21.0);
  }
}
```

Geben Sie für die folgenden Codeabschnitte an ob sie kompilieren. Wenn ja, schreiben Sie hin, was der Code ausgibt. Wenn nein, erklären Sie, warum nicht.

Indicate for the following code fragments if they compile. If yes, write down the output of the code. If no, explain why not.

```
Banana b;
b = new Banana(1.05, "yesterday");
b.print();
```

√ Ausgabe / *Output*: ◯ Compiler Error:

```
Banana [exp. yesterday] : CHF 1.05
```

```
Item i;
i = new Banana("Banana", 1.05);
i.print();
```

◯ Ausgabe / *Output*: √ Compiler Error:

```
Error: wrong constructor arg types
```

```
Banana b = new Banana(1.35, "today");
Food f = b;
Out.print(f.description());
```

√ Ausgabe / *Output*: ◯ Compiler Error:

```
Banana [exp. today]
```

```
Item i = new Stone();
NonFood nf = i;
nf.print();
```

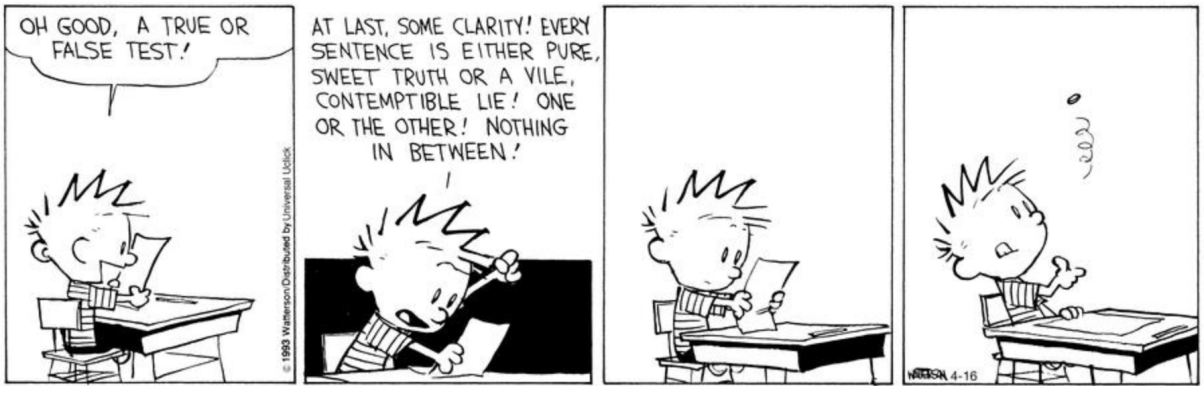
◯ Ausgabe / *Output*: √ Compiler Error:

```
Error: Can't assign Item to NonFood
```

```
Banana banana;
banana = new Banana(1, "Banana");
Out.print(banana.name);
```

√ Ausgabe / *Output*: ◯ Compiler Error:

```
Banana
```



Aufgabe 6: Java Collections (3P)

Sie haben in der Vorlesung diverse Java Collections kennen gelernt. Für Listen, Sets und Maps wurden jeweils mehrere Implementierungen diskutiert, mit unterschiedlichen Eigenschaften. Ziel ist es, je nach Situation die beste Datenstruktur zu wählen.
Für die folgenden drei Szenarien, geben Sie jeweils an, welche Java Datenstruktur sich am besten eignet fuer den Job. In allen drei Situation wollen Sie möglichst effizient auf gespeicherte Werte zugreifen.

You were introduced to various Java collections in the lecture. For lists, sets and maps, multiple implementations have been discussed, each with different properties. The goal is to choose the best data structure for a given situation.
For the following three scenarios, name the Java data structure that is best suited for the job. In all three situations, you want to access stored values as efficiently as possible.

/1P (a) Sie wollen Schlüssel und Werte speichern. Weder die Einfügereihenfolge noch die Sortierung der Schlüssel ist relevant. Welche Java Datenstruktur wählen Sie?

You want to store keys and values. Neither the insertion order nor the sorting of the keys is relevant. What Java data structure do you choose?

```
Hashmap / Treemap
```

/1P (b) Sie wollen Werte ablegen und dabei Duplikate vermeiden. Die Einfügereihenfolge ist relevant und muss erhalten bleiben.

You want to store values without duplicates. The insertion order is relevant and must be preserved.

```
LinkedHashSet / LinkedTreeSet
```

/1P (c) Sie wollen Werte ablegen. Duplikate sollen möglich sein. Zugriffe auf Werte erfolgen meist wahlfrei.

You want to store values. Duplicates should be possible. Typically, random access to values is required.

```
ArrayList
```