Axel Tetzlaff
21. August 2012

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Aufgaben zur Klausur Objektorientierte Programmierung im SS 2015

Zeit: 90 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.
Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 7 Seiten.

Die folgende Klassenhierarchie stellt eine spezielle Darstellung für zusammengesetzte Zeichenketten dar.
Die Idee hinter der Datenstruktur ist es eine speichereffiziente Arbeit mit großen Zeichenketten zu ermöglichen. Anstatt immer wieder neue `String` Objekte zu erzeugen, sollen beim Zusammensetzen (z.B. von Buchkapiteln) bereits erzeugte Zeichenketten wiederverwendet werden.

An einigen Stellen fehlen Implementierungen, die in den Aufgaben von Ihnen erarbeitet werden sollen.

```
1  import java.util.*;
2
3  class MyString {
4
5    public static abstract class Str implements Iterable<Character> {
6
7      public static Str empty()                   { // Aufgabe 1 }
8      public static Str fromString(String string) { return new ConstStr(string); }
9
10     public abstract int length();
11     public abstract Character charAt(int i);
12
13     public Str reverse()                        { // Aufgabe 5 }
14     public Str append(Str other)                { return new CompoundString(this, other); }
15     public Str append(String string)            { return append(fromString(string)); }
16     public boolean isEmpty()                    { return false; }
17     public Iterator<Character> iterator()       { return new StrIterator(); }
18
19     // Aufgabe 4
20
21   }
22
23   private static class EmptyStr extends Str {
24
25     // Aufgabe 1
26
27     public int length()            { return 0; }
28     public Character charAt(int i) { throw new NoSuchElementException(); }
29     public Str reverse()           { return this; }
30
31     public boolean isEmpty()       { return true; }
32     public Str append(Str other)   { return other; }
33     public String toString()       { return ""; }
34
35   }
36
37   private static class ConstStr extends Str {
38
39     private final String string;
40
41     public ConstStr(String string) { this.string = string; }
42     public String toString()       { return string; }
43     public Character charAt(int i) { return string.charAt(i); }
44     public int length()            { return string.length(); }
45
46   }
47
48   private static class CompoundString extends Str {
49
50     /* Aufgabe 2 */ Str head;
51     /* Aufgabe 2 */ Str tail;
52
53     public CompoundString(Str head, Str tail) {
54       this.head = head;
55       this.tail = tail;
56     }
57
58     // Aufgabe 3
59   }
60
61   public static void main(String args[]) {
62     final Str space = Str.fromString(" ");
63     Str myStr = Str.empty();
64
65     for(String arg : args) {
66       myStr = myStr.append(arg).append(space);
67     }
```

```
68      if (myStr.isEmpty()) {
69        System.out.println("Langweilig...");
70        return;
71      }
72
73      System.out.println("input: '" + myStr + "' :" + myStr.length()) ;
74
75      final Str reversed = myStr.reverse();
76      System.out.println("reverse:  " + reversed);
77
78      if (myStr.length() > 4) {
79        System.out.println("5. Buchstabe: " + myStr.charAt(4));
80        System.out.println("5. Buchstabe von hinten : " + reversed.charAt(4));
81      }
82
83      for(Character c : myStr ) {
84        System.out.println(c);
85      }
86    }
}
```

Die `main`-Methode zeigt die bestimmungsgemäße Verwendung der `Str`-Klasse. Der folgende Absatz zeigt die Ausgaben für den Aufruf '`java MyString Hello World!`'. Achten Sie bei den von Ihnen angegebenen Implementierungen darauf, dass die Aufrufe aus der `main`-Methode diese Ausgaben ergeben.

```
#> java MyString Hello World!
input: 'hello world! ' :13
reverse:   !dlrow olleh
5. Buchstabe: o
5. Buchstabe von hinten : r
h
e
l
l
o

w
o
r
l
d
!
```

Geben Sie für jeden Codeblock den Sie schreiben an, an welcher Stelle dieser im obenstehenden Listing eingefügt werden soll.

Aufgabe 1

Die leere Zeichenkette wird durch ein spezielles Objekt der Klasse `EmptyStr` repräsentiert. Um eine Referenz auf die leere Zeichenkette zu erlangen ist die Methode `empty()` in der `Str`-Klasse vorgesehen. Geben Sie den Code an, der für die Implementierung dieser Methode sowie in der Klasse `EmptyStr` fehlt. Vermeiden Sie die Erzeugung unnötiger Objekte auf dem Heap.

Aufgabe 2

Die Klasse `CompoundStr` repräsentiert eine aus zwei Teilen zusammengesetzte Zeichenkette.

Unterstreichen sie die Modifier die Sie bei der Deklaration der Referenzen auf die Teilzeichenketten in Zeile 50 und 51 nutzen würden:

public	static	private	abstract	protected	final

Begründen Sie ihre Wahl.

Aufgabe 3

Implementieren Sie die fehlenden Methoden der `CompoundStr`-Klasse. Speichern Sie dabei keine Referenzen auf `String`-Objekte in Instanz-Variablen. Erzeugen Sie keine neuen Instanzen der `ConstStr`-Klasse.

Aufgabe 4

Um - wie in Zeile 82 gezeigt - zeichenweise über eine Zeichenkette unserer Klasse `Str` iterieren zu können, implementiert `Str` die Schnittstelle `Iterable<Character>`. Diese schreibt uns eine Operation `iterator()` vor wie, sie in Zeile 17 implementiert ist.

Geben Sie eine Implementierung für die Klasse `StrIterator` an wie man Sie in Zeile 19 beginnend einfügen kann.

Für das Erreichen der vollen Punktzahl darf diese nur eine einzige Instanz-Variable deklarieren.
Das Iterator Interface ist wie folgt deklariert:

```
interface Iterator <E> {
  // Returns true if the iteration has more elements.
   boolean hasNext();

  // Returns the next element in the iteration.
  // Throws:
  //  NoSuchElementException – if the iteration has no more elements
   E next();

  //Removes from the underlying collection the last element returned by this iterator (optional operation).
  // Throws:
  //  UnsupportedOperationException – if the remove operation is not supported by this iterator
```

```
  //  IllegalStateException – if the next method has not yet been called, or the remove method has
                                already been called after the last call to the next method
   void remove();
}
```

Aufgabe 5

Die Methode `reverse()` in Zeile 13 soll eine Referenz auf ein Objekt mit der Schnittstelle `Str` zurückgeben, welches die übergebene Zeichenkette in umgekehrter Reihenfolge repräsentiert.
Implementieren Sie den Methodenrumpf der Methode `reverse()`. Falls Code außerhalb der Methode notwendig ist, geben Sie an in welcher Zeile dieser eingefügt werden soll.
Speichern Sie keine Referenzen auf `String`-Objekte in Instanz-Variablen. Erzeugen Sie keine neuen Instanzen der Klasse `ConstStr`.

Aufgabe 6

Die Klasse `Str` und ihre Unterklassen sind als `static` deklariert. Ist das sinnvoll? ☐ ja ☐ nein

Begründung:

Die Unterklassen von `Str` sind als `private` deklariert. Ist das sinnvoll? ☐ ja ☐ nein

Begründung:

Stimmt die folgende Aussage? Die Klasse `Str` enthält zwei unterschiedliche Methoden `append(…)`. Beim Aufruf von `append(..)` wird zur Laufzeit über dynamisches Binden ermittelt welche der beiden Methodenrümpfe in `Str` ausgeführt wird. ☐ ja ☐ nein

Begründung:

In der Klasse EmptyStr wird die Methode append überschrieben. Ist diese Implementierung sinnvoll? ☐ ja ☐ nein

Begründung:

Aufgabe 7

Angenommen folgende Zeilen werden am Ende der `main`-Methode, hinter Zeile 85 einfügt. Welche davon werden abgelehnt?

```
1 Iterator i1 = myStr.iterator();
2 Iterator<Object> i2 = myStr.iterator();
3 Iterator<?> i3 = myStr.iterator();
4 reversed = Str.fromString("foo");
5 Str s1 = new Str();
```

Zeile	kompiliert nicht, weil..

Aufgabe 8

Angenommen folgende Zeilen werden in der `main`-Methode hinter Zeile 85 einfügt.

```
1     Str a42 = Str.fromString("42");
2     Str b42 = Str.fromString("42");
3     Str empty1 = Str.fromString("");
4     Str empty2 = Str.fromString("");
5     Str empty3 = Str.empty();
6     Str empty4 = empty3;
7     Str empty5 = Str.empty();
8
9    boolean b;
10    b = a42 == b42;
11    b = empty1 == empty2;
12    b = empty1.equals(empty2);
13    b = empty1 == empty3;
14    b = empty1.equals(empty3);
15    b = empty3 == empty4;
16    b = empty5.equals(empty4);
```

Welche der Ausdrücke, die auf `b` zugewiesen werden, werden zu `true` ausgewertet?

Zeilennummern: