Dr. Annette Bieniusa
Mathias Weber, M. Sc.
Peter Zeller, M. Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Lösungshinweise/-vorschläge zum Übungsblatt 10: Software-Entwicklung 1 (WS 2016/17)

Die Hinweise und Vorschläge in diesem Dokument sollen der Lösungsfindung dienen und erheben demnach weder Anspruch auf Vollständigkeit noch Korrektheit. Sollten Sie Fehler finden, würden wir uns freuen, wenn Sie uns diese mitteilen. (Kontaktdaten finden Sie auf unserer Webpräsenz.)

Zusätzliche Fragestunde: Wir haben das Feedback erhalten, dass der aktuelle Termin der zusätzlichen Fragestunde für einige Interessierte ungünstig ist. Wenn Sie Interesse an der Fragestunde haben, füllen Sie bitte die folgende Doodle-Umfrage aus, damit wir einen besseren Termin finden können: http://doodle.com/poll/d3v8wy9ivbgydg57

Aufgabe 1 Fehlerbehandlung, IO (Sinnvoll bearbeiten)

In dieser Aufgabe sollen Sie das Behandeln von Exceptions am Beispiel vom Datei- und Netzwerk-Zugriff üben. In diesem Kontext werden Exceptions in Java oft verwendet, um Fehler zu signalisieren. Weitere Informationen zu den genauen Typen und Methoden-Signaturen finden Sie in der Dokumentation zur Java-Standardbibliothek unter https://docs.oracle.com/javase/8/docs/api/overview-summary.html. Die hier genannten Klassen befinden sich in den Paketen `java.io`, `java.nio.file` und `java.net`.

Ziel dieser Aufgabe ist es ein Programm zu schreiben, welches eine URL einer Internet-Seite als Parameter nimmt und prüft, ob sich diese seit dem letzten Aufruf des Programms verändert hat. Dazu soll das Programm immer den Inhalt vom letzten Aufruf abspeichern.

Wir wollen dieses Programm mit Hilfe von Streams implementieren, so dass nie der gesamte Inhalt einer Webseite oder Datei im Arbeitsspeicher gehalten werden muss und keine Datei oder Webseite mehrmals gelesen wird.

a) Schreiben Sie eine Klasse `TeeInputStream`, welche im Konstruktor einen `InputStream is` und einen `OutputStream os` nimmt. Die Klasse soll die Klasse `InputStream` erweitern. Implementieren Sie die Methoden **`public int`** `read()` **`throws`** `IOException` und **`public void`** `close()` **`throws`** `IOException`. Die Methode `read` liest vom `InputStream is`, schreibt das gelesene **`byte`** in den `OutputStream os` und gibt es außerdem zurück. Die Methode `close` liest alle verbleibenden Daten vom `InputStream is` und schreibt sie in den `OutputStream os`; anschließend schließt sie beide Ströme.

   Wir werden diese Klasse benutzen, um Daten von der Internetseite zu lesen und gleichzeitig in eine Datei zu schreiben.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class TeeInputStream extends InputStream {
    private InputStream in;
    private OutputStream out;
    private boolean closed = false;

    public TeeInputStream(InputStream in, OutputStream out) {
        this.in = in;
```

```
        this.out = out;
    }

    @Override
    public int read() throws IOException {
        int i = in.read();
        if (i >= 0) {
            out.write(i);
        }
        return i;
    }

    @Override
    public void close() throws IOException {
        if (closed) {
            return;
        }
        while (read() >= 0);
        out.close();
        in.close();
        closed = true;
    }
}
```

b) Schreiben Sie eine Methode `streamsEqual` in einer Klasse `WebChecker`, welche zwei `InputStreams` nimmt und prüft, ob diese den gleichen Inhalt haben. Verwenden Sie die `read`-Methode von `InputStream` und vergleichen Sie die Streams Byte für Byte.

```
private static boolean streamsEqual(InputStream a, InputStream b) throws
IOException {
    while (true) {
        int readA = a.read();
        int readB = b.read();
        if (readA != readB) {
            // contents not equal
            return false;
        } else if (readA < 0) {
            // both at end
            return true;
        }
    }
}
```

c) Schreiben Sie eine Methode `check` in der Klasse `WebChecker`, welche die URL (Typ `java.net.URL`) einer Internet-Seite und den Datei-Pfad (Typ `java.nio.file.Path`) der Backup-Datei nimmt und prüft, ob sich der Inhalt der Internetseite vom Inhalt der Backup-Datei unterscheidet. Gehen Sie dazu wie folgt vor:

- Öffne einen `InputStream` der Backup-Datei (siehe Methode `Files.newInputStream`). Falls keine Backup-Datei existiert, erstelle einen `ByteArrayInputStream` mit einem leeren **`byte`**-Array.
- Öffne einen `OutputStream` zu einer temporären Datei (siehe Methoden `Files.newOutputStream` und `Files.createTempFile`)
- Öffne einen `InputStream` der URL (siehe Methode `openStream` von `URL`).
- Erstelle einen `TeeInputStream`, welcher den `InputStream` der URL in den `OutputStream` der temporären Datei umleitet.
- Verwende die Methode `streamsEqual` um den Inhalt des `TeeInputStream` mit dem `InputStream` der Backup-Datei zu vergleichen.
- Wenn kein Fehler aufgetreten ist, schließe den `TeeInputStream`, so dass die Website komplett gespeichert wird und ersetze dann die alte Backup-Datei durch die temporäre Datei. Verwende dazu den Aufruf `Files.move(tempFile, backup, StandardCopyOption.ATOMIC_MOVE)`, welcher dafür sorgt,

dass Abstürze während dem verschieben der Datei die Backup-Datei nicht beschädigen können.

```
private static boolean check(URL url, Path backup) throws IOException {
    Path tempFile = Files.createTempFile("temp", ".html");
    // Hinweis: Unter Windows kann das verwenden von temporären Dateien zu
    // Problemen führen. Statt dessen kann auch eine Datei im lokalen
    // Verzeichnis verwendet werden:
    // Path tempFile = Paths.get(".","webchecker-temp.html");
    try (InputStream file = openBackup(backup);
         InputStream wepage = url.openStream();
         OutputStream tempFileStream = Files.newOutputStream(tempFile);
         InputStream tee = new TeeInputStream(wepage, tempFileStream)) {
        boolean result = streamsEqual(file, tee);
        tee.close();
        tempFileStream.close();

        Files.move(tempFile, backup, StandardCopyOption.ATOMIC_MOVE);

        return !result;
    }
}

private static InputStream openBackup(Path backup) throws IOException {
    try {
        return Files.newInputStream(backup);
    } catch (NoSuchFileException e) {
        return new ByteArrayInputStream(new byte[0]);
    }
}
```

d) Schreiben Sie die `main`-Methode für Ihr Programm. Das Programm nimmt die URL der Webseite und den Pfad der Backup-Datei als Programm-Parameter und ruft dann entsprechend die Methode `check` auf. Das Programm soll auf der Konsole ausgeben, ob sich die Webseite verändert hat.

Wenn Fehler auftreten soll eine nette Fehlermeldung auf der Konsole ausgegeben werden. Behandeln Sie die Ausnahmen `UnknownHostException`, `FileNotFoundException`, `NoSuchFileException` und `MalformedURLException` mit entsprechenden Nachrichten. Behandeln Sie auch fehlerhafte Programm-Parameter.

Führen Sie Ihr Programm mit verschiedenen Fehler-Situationen aus, zum Beispiel:

```
java WebChecker https://softech.cs.uni-kl.de ordner_existiert_nicht/backup.html
java WebChecker https://softech.cs.uni-kl.de/seite_existiert_nicht backup.html
java WebChecker https://domain_gibt_es_nicht.de backup.html
java WebChecker gar_keine_url backup.html
```

```
public static void main(String[] args) {
    try {
        if (args.length < 2) {
            System.out.println("Nicht genug Programm-Parameter angegeben!");
            return;
        }
        URL webpage = new URL(args[0]);
        Path backupFilePath = Paths.get(args[1]);
        boolean res = check(webpage, backupFilePath);
        if (res) {
            System.out.println("Webseite hat sich geaendert!");
        } else {
            System.out.println("Keine Aenderungen.");
        }
    } catch (MalformedURLException e) {
        System.out.println("Erster Parameter ist keine gueltige URL.");
    } catch (UnknownHostException e) {
        System.out.println("Der angegebene Server konnte nicht erreicht werden"
);
```

```
        } catch (FileNotFoundException e) {
            System.out.println("Webseite " + e.getMessage() + " wurde nicht
    gefunden.");
        } catch (NoSuchFileException e) {
            System.out.println("Datei nicht gefunden. " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Es ist ein unbekannter Fehler aufgetreten : " + e.
    getMessage());
            e.printStackTrace();
        }
    }
```

Aufgabe 2 Defensive Programmierung (Einreichaufgabe, 4 Punkte)

```
/*
 * requires entries != null
 *       && key != null
 *       && für alle e in entries: e != null
 *       && entries enthält einen Eintrag e mit key.equals(e.getKey())
 * returns e.getValue(), wobei e der erste Eintrag in entries
 *                       mit key.equals(e.getKey()) ist
 */
public static Object find(List<Entry> entries, String key) {
    if (entries == null) {
        throw new IllegalArgumentException("entries ist null");
    }
    if (key == null) {
        throw new IllegalArgumentException("key ist null");
    }
    for (Entry e : entries) {
        if (e == null) {
            throw new IllegalArgumentException("entries enthaelt einen null-
Eintrag");
        }
        if (key.equals(e.getKey())) {
            return e.getValue();
        }
    }
    throw new IllegalArgumentException("Kein Eintrag mit gesuchtem Schluessel
vorhanden");
}
```

Aufgabe 3 Warteschlange (Einreichaufgabe, 15 Punkte)

```
public class ElementNullException extends Exception {}
public class QueueEmptyException extends Exception {}
public class QueueFullException extends Exception {}


public class ArrayQueue<T> implements Queue<T> {
    private final T[] buffer;
    private int size = 0;
    private int headPos = 0;

    ArrayQueue(int capacity) {
        @SuppressWarnings("unchecked")
        T[] a = (T[]) new Object[capacity];
        buffer = a;
    }

    public void add(T e) throws QueueFullException, ElementNullException {
        if (e == null) {
            throw new ElementNullException();
        }
        if (size >= buffer.length) {
            throw new QueueFullException();
        }
        buffer[(headPos + size) % buffer.length] = e;
        size++;
    }

    public T remove() throws QueueEmptyException {
        if (size == 0) {
            throw new QueueEmptyException();
        }
        T e = buffer[headPos];
        buffer[headPos] = null; // allow garbage collection to clean element
        headPos = (headPos + 1) % buffer.length;
        size--;
        return e;
    }

    public T element() throws QueueEmptyException {
        if (size == 0) {
            throw new QueueEmptyException();
        }
        return buffer[headPos];
    }

    public boolean isEmpty() {
        return size == 0;
    }

}
```

Aufgabe 4 Hallo C (Sinnvoll bearbeiten)

In der nächsten Woche werden wir die Programmiersprache C behandeln. Als Vorbereitung sollen Sie in dieser Aufgabe ein einfaches C-Programm übersetzen und ausführen.

Falls Sie einen eigenen Computer verwenden, installieren Sie sich bitte einen C-Compiler (Anleitungen dazu finden Sie spätestens zur Vorlesung am Donnerstag auf der SE1-Webseite). Auf den Linux-Rechnern des SCI ist der C-Compiler bereits installiert. Die folgenden Befehle beziehen sich auf den C-Compiler clang. Wenn Sie einen anderen Compiler verwenden, müssen Sie entsprechend andere Befehle zum Übersetzen verwenden.

Erstellen Sie eine Datei `hello.c` mit dem folgenden Inhalt:

```
#include <stdio.h>

int main() {
    printf("Hallo C!\n");
    return 0;
}
```

Übersetzen Sie dann die Datei mit dem folgenden Befehl:

```
$ clang hello.c -o hello
```

Führen Sie dann das Programm `hello` wie folgt aus:

```
$ ./hello
```