Institut für Informatik
Prof. Dr.-Ing. Elke Pulvermüller
Lars Huning, M.Sc.

Universität Osnabrück, 07.06.2021
Testat bis 16.06.2021, 18:00 Uhr

Übungen zu Einführung in die Software-Entwicklung

Sommersemester 2021

Blatt 8

Aufgabe 8.1: Theorie: Decorator-Entwurfsmuster (10 Punkte)

Bereiten Sie sich darauf vor, Ihrem Tutor das Decorator-Entwurfsmuster zu erklären, sowie dessen Vor- und Nachteile zu erläutern. Erklären Sie außerdem die Begriffe *System.in*, *System.out* und *System.err*, sowie das Vorgehen um etwas von der Kommandozeile während der Ausführung des Programms einzulesen (Hinweis: Verwendung von *System.in*).

Aufgabe 8.2: Ströme (30 Punkte)

Machen Sie sich mit den Klassen im Paket `java.io` der Java-API vertraut. Nutzen Sie das in der Vorlesung vorgestellte *Decorator*-Pattern, um eine eigene `Reader`-Klasse zu implementieren mit der man

- über die Methode `readLine()` alle Zeichen von einem `Reader` bis zum nächsten Zeilenumbruch einlesen und als einen `String` zurückgeben kann. Der Zeilenumbruch soll nicht mit zurückgegeben werden. Beim Erreichen des Dateiendes soll `null` zurückgegeben werden.
- über die Methode `getLineNumber()` die Nummer der zuletzt gelesenen Zeile ermitteln kann.
- über die Methode `getAmountOfMatches()` ermitteln kann, wie oft ein dem Konstruktor übergebener regulärer Ausdruck in der zuletzt gelesenen Zeile gefunden wurde.

Sie können zur Verarbeitung der regulären Ausdrücke die Klassen `java.util.regex.Pattern` und `java.util.regex.Matcher` verwenden.

Implementieren Sie anschließend ein Kommandozeilenprogramm, das mit einem regulären Ausdruck aufgerufen wird und den Inhalt einer Datei über den *Pipe-Operator* < zugewiesen bekommt. Das Programm soll jede Zeile der Datei, die den regulären Ausdruck mindestens einmal enthält, zusammen mit der Zeilennummer auf der Standardkonsole ausgeben. Zusätzlich soll für jede ausgegebene Zeile die Anzahl der Vorkommen des regulären Ausdrucks ausgeben werden.

Ein Aufruf des Programms kann beispielsweise wie folgt aussehen:

```
java io/SearchLines "pu.*c" < Beispiel.java
```

Der Inhalt von `Beispiel.java` wird damit in den Standard-Eingabestream `System.in` geschrieben.

Sofern Sie Eclipse verwenden, können Sie statt dem Pipe-Operator eine Eingabedatei unter RunConfigurations → Common → Input-File angeben. Den Reiter Commons finden Sie, wenn Sie die Zeile, in der auch Main, Arguments und JRE steht, ausklappen.

Aufgabe 8.3: Standard-Serialisierung (30 Punkte)

Betrachten Sie das Programm zum Berechnen der Fibonacci-Zahlen `Fibonacci.java`, das bereits berechnete Fibonacci-Zahlen in einer `HashMap` vorhält.

Verändern Sie dieses Programm dahingehend, dass die eingesetzte `java.util.HashMap` beim Start aus einer Datei gelesen und nach der Beendigung des Programms wieder in diese Datei zurückgeschrieben wird. Sollte das Programm zum ersten Mal überhaupt aufgerufen werden, soll eine neue `java.util.HashMap` erzeugt werden.

Aufgabe 8.4: Spezielle Threads (30 Punkte)

Implementieren Sie ein Programm, das darauf horcht, ob sich die Größe eines in den Kommandozeilenargumenten angegeben Verzeichnisses oder einer Datei verändert hat. Nutzen Sie dazu einen `java.util.TimerTask`, der einmal pro Sekunde überprüft, ob sich die Größe einer `File`-Instanz verändert hat und im Falle einer Änderung die Größe auf der Kommandozeile ausgibt. Die Größe eines Verzeichnisses sei hier die Größe aller Dateien und Unterverzeichnisse zusammengenommen. Das Programm soll nur über den Konsolenbefehl *ctrl + C* beendet werden können und im Falle des Beendens noch eine Meldung auf der Kommandozeile ausgeben.