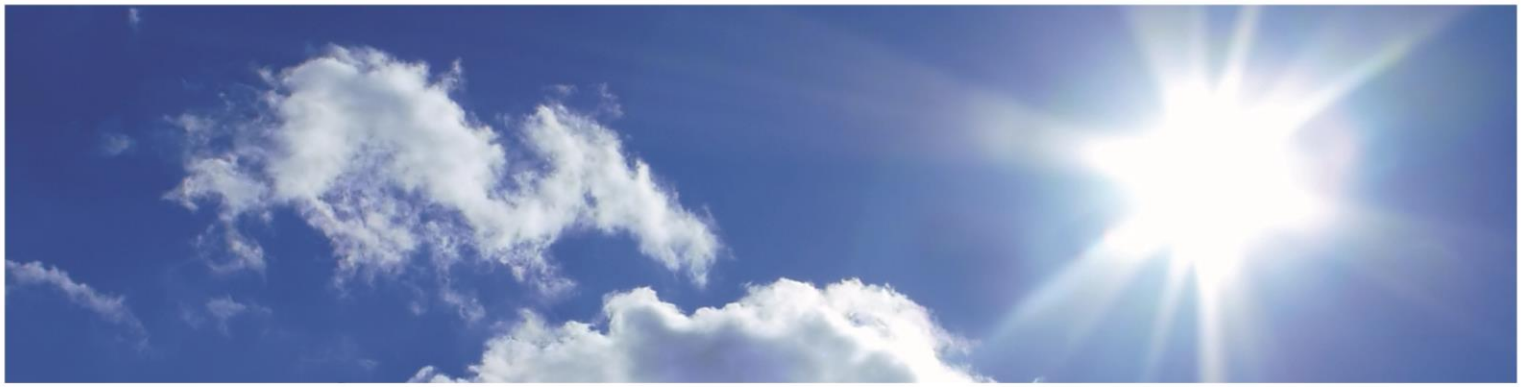


Self-study Exercises for Programming

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

PALUNO
The Ruhr Institute for Software Technology

SEP

SEP-Team

Tristan Kley, Philip Bohn, Julian Föckler, Mostapha Ahaduch, Jan Löber, Semra Burschik

Vorwort

Das Programmieren zu beherrschen, eine Tätigkeit zwischen Wissenschaft und Kunst, ist notwendig für jeden studierten Informatiker, sei er nun im konkreten Studiengang Wirtschaftsinformatik, Angewandte Informatik, Lehramt oder anderem. Zumindest ist die Universität Essen dieser Auffassung, da Module zur Programmierung in all diesen Studiengängen enthalten sind. Wenn Du dies liest – vielen Dank dafür – interessierst Du dich vermutlich für einen davon, Software Entwicklung und Programmierung, kurz SEP. In dieser Veranstaltung wirst Du im Verlauf eines Semesters begleitet ein Software-Projekt von Beginn bis Ende umsetzen. Dabei wirst Du selber Programmieren müssen, um die Veranstaltung zu bestehen. Wir hoffen, dass dieses Dokument Dir hilft, Deine bestehenden Kenntnisse zur Programmierung im Selbststudium aufzufrischen und zu erweitern, sodass Du perfekt vorbereitet bist auf SEP.

Was dieses Dokument nicht ist, ist ein Lehrbuch. Es ist nur eine Ansammlung von Übungen zu Themen, von denen Kenntnisse im SEP vorausgesetzt werden. Zu jedem Thema gibt es einen kurzen Einleitungstext, der aber explizit nicht alle Informationen beinhaltet, die gebraucht werden um die Übungen zu diesem Thema zu bearbeiten. Diese kurzen Einleitungen sollen lediglich dazu befähigen, weitere Informationsquellen zu dem Thema verstehen zu können. Um diese Übungen zu bearbeiten ist es notwendig darüber hinaus Informationsquellen eigenständig zu suchen.

Die Übungsaufgaben selbst sollen dabei einen Kontext zum Lernen geben, nicht Wissen abfragen oder möglichst schnell bearbeitet werden. Viele der Aufgaben haben Tipps um bei der Lösung zu helfen, diese sind in schwarzer Schrift auf schwarzem Grund geschrieben. [redacted] Wenn Du einen solchen Tipp anschauen möchtest, musst Du ihn nur markieren und in einen Text-Editor kopieren.

Zu allen Aufgaben gibt es online auch Lösungsvorschläge, so dass Du dir nach dem Bearbeiten eine alternative Lösung anschauen kannst, oder weitere Hilfe findest, solltest Du an einer Aufgabe gar nicht mehr weiter kommst. Diese findest Du alle in dem Gitlab-Repository:

https://git.uni-due.de/sktrkley/self-study-exercises-for-programming

Viel Spaß beim Lernen!

Inhaltsverzeichnis

01 Seperation of Concerns

Seperation of Concerns ist ein Software Design Paradigma, das vorsieht den entwickelten Quellcode in verschiedene, getrennte Abschnitte zu unterteilen. Jeder dieser Abschnitte soll dann genau eine klar definierte Aufgabe erfüllen, man sagt auch eine klar definierte Verantwortlichkeit haben. Objektorientierte Programmierung eignet sich gut für die Umsetzung dieses Paradigmas, da man die einzelnen Verantwortlichkeiten auf die verschiedenen Klassen und Objekte aufteilen kann.

Beispiel: Angenommen man möchte ein Programm schreiben, das die x- und y-Koordinaten von mehreren Punkten aus einer Datei ausliest und diese Punkte in einem Fenster zeichnet. Ein solches Programm ließe sich in die Verantwortlichkeiten aus der folgenden Tabelle aufteilen. Man könnte sicherlich auch andere Aufteilungen wählen oder diese Verantwortlichkeiten weiter aufteilen. Aus den Verantwortlichkeiten kann man dann direkt Klassen ableiten, aus denen man sein Projekt aufbauen könnte.

Verantwortlichkeit	abgeleitete Klasse
Auslesen der Koordinaten aus der Datei	PointReader
Behandlung von Fehlern beim Lesen	
Zusammenfassen von je zwei Koordinaten zu einem Punkt (Datenmodell)	Point
Zeichnen der Punkte	GUI
Öffnen eines Fensters	
Zeichnen eines Koordinatensystems	
Einzeichnen der Punkte	
Starten des Programms	Launcher
Instanziieren des PointReaders und der GUI	

Ein großer Vorteil des Seperation of Concerns Paradigma ist, dass die damit entwickelten Programme automatisch modular sind, da man die Verschiedenen Klassen auch jeweils als ein Modul auffassen kann. Man kann eine Klasse, wenn die Klassen gut dokumentiert sind, getrennt von den anderen entwickeln und muss nicht die genaue Funktionsweise der anderen Klassen kennen oder verstehen, sondern nur wissen welche Aufgabe diese Klassen erfüllen, was ihre Verantwortlichkeit ist.

Aufbauend auf dem Seperation of Concerns Paradigma, ist das Model-View-Controller Software Entwurfsmuster (engl. design-pattern) ein weit verbreiteter Ansatz um die Aufteilung in Verantwortlichkeiten zu beginnen. Das MVC Entwurfsmuster gibt vor die Software in mindestens drei getrennte Module, das Model, die View, und die Controller, aufzuteilen. Die Verantwortlichkeit der View ist es den Nutzern Informationen darzustellen, die Verantwortlichkeit der Controller ist es die Eingaben der Nutzer entgegenzunehmen und zu interpretieren, und das Model ist für alles weitere verantwortlich. Die genaue Definition der Verantwortlichkeiten im MVC Entwurfsmuster unterscheidet sich in verschiedenen Quellen, ist allerdings immer ähnlich.

Nachdem diese Begriffe nun mehrfach genutzt wurden, müssen wohl noch abschließend ein paar Worte über den Unterschied zwischen einem Software Design Paradigma und einem Entwurfsmuster fallen. Ein Paradigma beschreibt einen Programmierstil, welcher von der genutzten Programmiersprache unterstützt werden muss, der es dem Programmierer einfacher machen soll ‚guten Code' zu schreiben. Da es gar nicht so klar ist, was ‚guten Code' überhaupt ausmacht und die Anforderungen an Software sehr vielseitig sein können, gibt es verschiedenste Paradigma. Neben der Speration of Concerns zählen beispielsweise auch die objektorientierte Programmierung und die funktionale Programmierung dazu.

Daneben ist ein Software Entwurfsmuster eher eine Schablonenlösung für häufig auftauchende Probleme in der Software-Architektur. Entwurfsmuster sind normalerweise für eines oder mehrere spezielle Paradigmen entwickelt und nicht direkt auf Programmiersprachen anwendbar, die diese Paradigmen nicht unterstützen, das MVC-Entwurfsmuster wurde beispielsweise für die objektorientierte Programmierung entworfen.

01.1 Aufgabe:

Entwickle ein kleines Programm, das dem Nutzer erlaubt auf der Konsole zwei Ganzzahlen einzutippen und ihm die Summe beider Zahlen, ebenfalls auf der Konsole, anzeigt. Identifiziere dabei zuerst die Verantwortlichkeiten des Programms und leite daraus die Klassen und Objekte ab, aus denen das Programm bestehen soll.

Tipp 1:

[redacted]

Tipp 2:

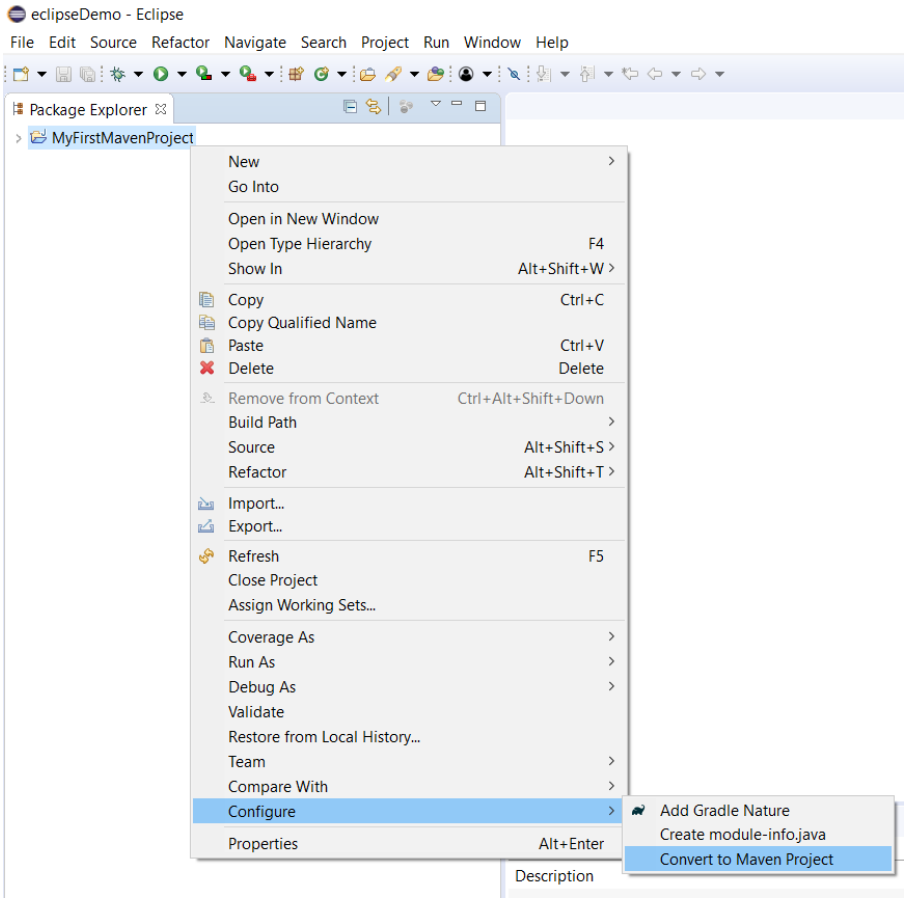
[redacted]

02 Benutzen einer Bibliothek

Eine Bibliothek ist eine Sammlung von Klassen, die Lösungen für bestimmte Probleme liefern. Die Java-Standartbibliothek bietet beispielsweise Klassen zum Einlesen von Dateien, zum Öffnen von Fenstern, zur Realisierung von Netzwerkkommunikation, und vieles mehr. Java-Bibliotheken haben meist das Dateiformat .jar. Diese Dateien sind in der Regel nicht ausführbar, die Klassen, aus denen sie bestehen, enthalten keine main-Methode. Die Java-Standartbibliothek ist automatisch in jedes Java-Projekt eingebunden, die .jar Dateien aller anderen Bibliotheken müssen erst dem Projekt hinzugefügt werden, damit diese benutzt werde können.

Das Einbinden einer Bibliothek in ein Projekt kann auf mehrere Arten geschehen, die einfachste ist die Benutzung eines Build-Management-Tools wie Maven oder Gradle. Die meisten IDEs unterstützen diese beiden Tools direkt. Des Weiteren bieten viele IDEs auch die Möglichkeit Bibliotheken einem Projekt manuell hinzuzufügen.

Um in Eclipse mit Maven eine Bibliothek einem Projekt hinzuzufügen, kann man im Package Explorer einen Rechtsklick auf das Projekt machen und im Untermenü „Configure“ den Menüpunkt „Convert to Maven Project“ auswählen.



Nachdem man den entstehenden Dialog bestätigt hat, wird dann eine pom.xml Datei in dem Projekt erstellt. Diese Datei definiert wie das Projekt strukturiert ist, und damit auch welche Bibliotheken das Projekt nutzt. Um eine Bibliothek dem Projekt hinzuzufügen, muss nur die pom.xml um den Abschnitt „dependencies“ erweitert werden und die Bibliothek dort als „dependency“ definiert werden. In der Abbildung ist ein Beispiel für die Bibliothek Gson in der Version 2.8.5 zu sehen.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:>
  <modelVersion>4.0.0</modelVersion>
  <groupId>MyFirstMavenProject</groupId>
  <artifactId>MyFirstMavenProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.8.5</version>
    </dependency>
  </dependencies>
</project>
```

Man kann auf der Seite *mvnrepository.com* nach allen Bibliotheken suchen, die man mit Maven ohne weiteres auf diese Weise einbinden kann.

Informationen über die Benutzung jeglicher Java-Bibliotheken findet man vorrangig in der Javadoc der Bibliothek, eine Dokumentation, in der alle enthaltenen Klassen im Detail beschrieben werden. Weil Javadocs sehr groß sind und keinen klaren Einstieg haben, kann es nützlich sein zuerst auf Tutorials und Guides zurückzugreifen um einen ersten Anhaltspunkt dafür zu haben, wie eine Bibliothek zu nutzen ist.

02.1 Aufgabe:

Modifiziere dein Programm aus der vorherigen Aufgabe so, dass es nicht mehr die Summe der beiden eingegebenen Zahlen berechnet, sondern die Zahlen als erste und zweite Koordinate eines Punktes in einem kartesischen Koordinatensystem interpretiert. Das Programm soll dann den Winkel ausgeben, den die Gerade, die durch den eingegebenen Punkt und den Koordinatenursprung verläuft, und die Rechtsachse einschließen. Benutze hierfür die Apache Commons Math Bibliothek.

Tipp 1:

Tipp 2:

03 Debugger

Chuck Norris starrt seinen Code so lange an, bis dieser ihm die Fehler freiwillig zeigt. Für alle anderen gibt es Debugger. Die meisten IDEs enthalten solche Debugger Tools, mit denen man Fehler im Code leicht finden kann. Die gängigsten Funktionen von Debuggern sind die Steuerung des Programmablaufs durch das Unterbrechen der Ausführung an Haltepunkten und die weitere schrittweise Ausführung, sowie das Auslesen von Variablen zur Laufzeit.

Es gibt kaum Statistiken dazu wie hoch der Anteil an Arbeitszeit ist, der in einem Software-Projekt in Tests und Debugging fließt, die Zahlen, die man dazu findet, liegen aber immer irgendwo von 50% bis 90%. Bei so viel Zeit, die man ins Debugging steckt, ist es wichtig relativ effizient und schnell zu sein. Debugger sollen einem genau das erlauben. Obwohl man anfangs ein wenig Zeit investieren muss, um zu lernen wie Debugger funktionieren, sind wir vom SEP-Team, als erfahrene Entwickler, der Meinung, dass sich dieses Investment schon bei kleinen Projekten bezahlt macht.

03.1 Aufgabe:

Benutze den Debugger einer IDE wie Eclipse an dem Beispiel des Programms der vorherigen Aufgabe. Setze einen Haltepunkt so, dass es möglich ist das Ergebnis der Berechnung des Winkels im Debugger abzulesen, bevor es auf der Konsole ausgegeben wird.

Tipp 1:

Tipp 2:

04 JavaFX

JavaFX ist ein Teil der Java-Standartbibliothek, der es erlaubt graphische Oberflächen darzustellen und Nutzereingaben auf diesen Oberflächen entgegenzunehmen.

04.1 Aufgabe:

Erstelle ein Programm, das es dem Nutzer erlaubt in ein Textfeld in einem Fenster einen Text einzugeben und auf Knopfdruck den gesamten Text zu löschen.

Tipp:

00 Pause

Behalte im Hinterkopf, dass das Ziel dieser Aufgaben es nicht ist, Dein Wissen abzufragen oder sie möglichst schnell zu erledigen. Sie sollen lediglich als Kontext dienen, in dem Du Dich mit den angesprochenen Themen und Programmierung befassen und darüber lernen kannst.

00.1 Aufgabe:

Mach eine Pause! Geh an die frische Luft, spiel ein Videospiel, schau einen Film. Komm dann entspannt zurück und bearbeite die restlichen Aufgaben.

05 Threads

Was genau ist ein Thread? Das ist eine recht schwierige Frage, um sie zur Genüge zu beantworten müsste man wohl eine gesamte Vorlesung halten. Concurrency ist der Name dieser Vorlesung an unserer Universität. An dieser Stelle sei nur so viel gesagt, dass Threads erlauben Anweisung eines Programms scheinbar parallel auszuführen. Ebenso wie es möglich ist an einem Computer mehrere Prozesse zu starten, um beispielsweise gleichzeitig Musik zu hören und im Internet zu surfen, kann man beides auch mit mehreren Threads in nur einem Programm erledigen. Wenn man möchte, kann man sich jedes Programm als ein Team von Arbeitern vorstellen, wobei jeder Thread des Programms einen Arbeiter repräsentiert, der gleichzeitig mit seinen Kollegen eine Aufgabe erfüllen kann.

Das ganze Konzept scheint nun auf den ersten Blick unschuldig und unkompliziert, ebenso wie es das Starten von mehreren Prozessen ist. Bei genauerer Betrachtung ergeben sich aber einige kuriose Probleme. Mehrere Prozesse können nicht ohne weiteres auf Basis der selben Daten arbeiten, jeder Prozess hat einen Bereich im Arbeitsspeicher des Computers für sich reserviert und kein anderer Prozess darf ohne weiteres auf diesen Bereich zugreifen. Mit Threads ist dies (gewollt) anders. Grundsätzlich hält nichts mehrere Threads eines Programms davon ab auf dieselben Daten, auf den selben Bereich des Arbeitsspeichers, zuzugreifen. In Java würde dies bedeuten, dass diese Threads auf dieselben Objekte und Variablen zugreifen können. Das kann, wenn man als Entwickler nicht aufpasst, unangenehme Konsequenzen haben.

Zum Beispiel, um bei der Analogie mit den Arbeitern zu bleiben, man stelle sich vor vier Arbeiter würden in einem Lagerhaus an einem Stapel von ursprünglich fünf Kisten arbeiten. Ein Arbeiter fügt dem Stapel zehn Kisten hinzu, während der zweite die Kisten des Stapels zählt, der dritte sieben Kisten entfernt und der vierte in jede Kiste eine Rose legt. Alle Arbeiter arbeiten gleichzeitig und kommunizieren nicht miteinander. Was das Ergebnis dieser Arbeit wäre ist geradezu nicht vorstellbar. Wie viele Kisten zählt der zweite Arbeiter? Welche Kisten entfernt der dritte, entfernt er auch Kisten, die der erste dem Stapel hinzugefügt hat, wenn ja wie viele davon? Welche Kisten enthalten am Ende Rosen, auch die neu hinzugefügten und die entfernten?

Um diese Art von Problemen für Threads zu lösen, gibt es mehrere Möglichkeiten. Die einfachste ist wohl das Benutzen eines Lock-Objekts. Ein Lock funktioniert im Prinzip wie ein Wartebereich für beispielsweise einen Geldautomaten. Nur eine Person kann den Geldautomaten benutzen, alle anderen müssen warten bis diese Person fertig ist. Ebenso ist es auch mit dem Lock, man sagt nur nicht benutzen, sondern halten. Threads können, einfach durch den Aufruf einer Methode auf dem Objekt, versuchen ein Lock aufzuheben oder abzusetzen, aber immer nur ein Thread darf das Lock halten. Wenn ein Thread versucht ein Lock aufzuheben, welches schon von einem anderen Thread gehalten wird, so muss er warten bis das Lock von dem anderen Thread wieder abgesetzt wird.

Wie hilft das unseren Arbeitern? Angenommen es gibt neben dem Stapel an Kisten einen Hut auf dem „Lock“ in neon-roter Farbe geschrieben steht, und nur der Arbeiter, welcher diesen Hut trägt, darf an dem Stapel arbeiten. Zu Beginn der Arbeit versucht jeder Arbeiter den Hut aufzuheben, aber nur einer schafft es als erster. Dieser kann dann freudig mit seiner Arbeit beginnen, während die anderen Arbeiter warten müssen bis der erste den Hut wieder absetzt. Nun kann ein zweiter den Hut aufsetzen und Arbeiten, es geht so weiter bis alle Arbeiter einmal den Hut aufgesetzt und gearbeitet haben. Am Ende ist noch immer nicht eindeutig, wie der Stapel Kisten aussieht, dies hängt davon ab in welcher Reihenfolge die Arbeiter den Hut aufgesetzt haben, aber es gibt schon deutlich weniger Möglichkeiten. Beispielsweise, der zweite Arbeiter kann nur entweder 5, 0, 15 oder 8 Kisten zählen, je nach der Reihenfolge der Arbeiter. Jedoch, spätestens an dieser Stelle, sollte man sich fragen, ob es überhaupt sinnvoll ist die Arbeit auf vier Arbeite aufzuteilen. Mit dem Hut in der Analogie kann so oder so nur ein Arbeiter wirklich arbeiten, währenddessen die anderen drei warten. Vielleicht wäre es geschickter die

vier Aufgaben von vornherein nur einem Arbeiter zu geben, vielleicht auch in klarer Reihenfolge? Damit sei nicht gesagt, dass Parallelisierung keinen Nutzen hat, in manchen Fällen ist sie sogar notwendig. Beispielsweise muss vielleicht ein Arbeiter ständig nach einer neuen Lieferung Kisten Ausschau halten und kann nicht selbst gleichzeitig an dem Stapel arbeiten.

05.1 Aufgabe:

Erstelle ein Programm mit zwei Threads, die „gleichzeitig“ (der Fachausdruck wäre Nebenläufig) jeweils alle Ganzzahlen zwischen 0 und 1000 auf die Konsole schreiben.

Tipp:

05.2 Aufgabe:

Entwickle ein Programm mit zwei Threads die „gleichzeitig“ mit einem Queue-Objekt arbeiten. Dabei soll der erste dieser Threads der Queue in einem zufällig gewählten Abstand zwischen 500 und 1500 Millisekunden ein Element hinzufügen und dann die neue Länge der Queue auf der Konsole ausgibt. Der zweite Thread soll in einem zufällig gewählten Abstand zwischen 500 und 1500 Millisekunden das erste Element der Queue entfernen und ebenfalls die neue Länge der Queue auf der Konsole ausgeben. Die genaue Länge der Zeitabstände soll in jedem Durchlauf neu zufällig bestimmt werden.

Tipp:

05.3 Aufgabe:

Erweitere das Programm aus der vorherigen Aufgabe mit JavaFX durch eine graphische Benutzeroberfläche, die es dem Nutzer erlaubt einen String einzugeben, welcher dann von dem ersten Thread in die Queue gespeichert wird. Die Länge der Queue soll nicht mehr auf der Konsole, sondern auf der Nutzeroberfläche angezeigt werden, ebenso wie alle Inhalte der Queue.

Tipp:

06 Sockets

Sockets sind der Grundstein der meisten Netzwerkverbindungen, auch LANs und dem Internet. Sie funktionieren auf eine einfache Art und Weise. Ein Socket hat einen InputStream über den er Bytes empfangen kann und einen OutputStream über den Bytes senden kann. Am häufigsten benötigt man Sockets um TCP/IP Verbindungen zu realisieren, welche immer aus zwei Sockets bestehen. Dabei können die Bytes, die der erste Socket über seinen OutputStream sendet, über den InputStream des zweiten Sockets gelesen werden, und vice versa.

Von sich aus können Output- und InputStreams nur mit einfachen Bytes arbeiten. In vielen Anwendungsfällen möchte man allerdings komplexere Daten über eine Netzwerkverbindung versenden, beispielsweise einen String. Dafür gibt es in der Java-Standartbibliothek Klassen, die es erlauben komplexere Datenstrukturen in Bytes umzuwandeln um sie über einen OutputStream zu senden, und Klassen, die es erlauben diese Bytes wieder zurück in die komplexen Datenstrukturen zu übersetzen. Zwei dieser Klassen sind der PrintWriter, welcher Strings über einen OutputStream versenden kann, und der BufferedReader, welcher Strings von einem InputStream lesen kann.

06.1 Aufgabe:

Entwickle ein kleines Chat Programm, mit dem zwei Nutzer über eine Netzwerkverbindung mit einander chatten können, indem sie ihre Nachrichten auf der Konsole eingeben und die Nachrichten des anderen auf der Konsole lesen können.

Tipp 1:

Tipp 2:

Tipp 3:

06.2 Aufgabe:

Erweitere das Programm aus der letzten Aufgabe durch eine graphische Benutzeroberfläche, in der die Nutzer ihre Nachrichten eingeben und die Nachrichten des anderen lesen können. Außerdem sollen die Nutzer die IP-Adresse des andern eingeben können um sich mit ihm zu verbinden.

07 Dateien

Dateien sind Bündel von persistent auf einem Datenträger gespeicherte Daten, die über die Laufzeit eines Programms hinaus bestehen. Die momentanen gängigsten Betriebssysteme sortieren Dateien auf einem Datenträger in einem Dateibaum. In diesem Baum sind die Knoten entweder Ordner, welche auf weitere Knoten verweisen können, oder Dateien selbst. Durch den Dateibaum kann man, ähnlich wie bei anderen Graphen, mit Pfaden navigieren. Diese werden in absolute und relative Pfade unterschieden.

Absolute Pfade starten immer in der Wurzel des Dateibaums, auf einem Windows-Betriebssystem könnte ein solcher Pfad als String ausgedrückt beispielhaft so aussehen: „C:\TollerOrdner\TollereUnterordner\TollsteDatei.toll“
Wie ein Pfad als String ausdrückt werden kann unterscheidet sich von Betriebssystem zu Betriebssystem. Der selbe Pfad wäre auf einem Linux-Betriebssystem: „/mnt/c/TollerOrdner/TollereUnterordner/TollsteDatei.toll“

Relative Pfade sind einfach Pfade, die nicht in der Wurzel des Dateibaums starten. In vielen System gilt als Konvention, dass absolute Pfade mit einem Slash oder Backslash beginnen, relative Pfade nicht. Von daher wäre „TollererUnterordner/TollsteDatei.toll“ ein relativer Linux-Dateipfad, der aus dem Ordner „TollerOrdner“ heraus auf die Datei „TollsteDatei“ zeigt.

Weil Dateien vom Betriebssystem gehandhabt werden, ist der Zugriff auf sie in allen Programmiersprachen ähnlich. Über einen Pfad kann man eine Datei identifizieren und dann mit InputStreams aus der Datei lesen oder mit OutputStreams in die Datei Schreiben.

07.1 Aufgabe:

Entwickle ein kleines Programm, dass es dem Nutzer erlaubt einen Text auf einer Benutzeroberfläche einzugeben und dann in einer Datei zu speichern.

Tipp 1:

Tipp 2:

Tipp 3:

07.2 Aufgabe:

Erweitere das Programm aus der vorherigen Aufgabe so, dass es dem Nutzer auch erlaubt den Text aus einer Datei wieder zu laden.

Tipp:

08 libGDX

libGDX ist eine Open Source Java-Spiel-Engine. Anders als gewöhnliche Bibliotheken fügt man libGDX nicht einem bestehenden Projekt hinzu, sondern kann die libGDX eine Setup-Applikation benutzen, welche ein neues Projekt automatisch anlegt. Dieses neue Projekt kann man dann in die IDE seiner Wahl importieren.

08.1 Aufgabe:

Entwickle mit libGDX ein kleines Spiel, in dem der Spieler einfach nur ein Bild mit den Pfeiltasten auf dem Bildschirm verschieben kann.

Tipp:

08.2 Aufgabe:

Entwickle mit libGDX eine kleine Physik-Simulation, die den freien Fall einer Kugel im zweidimensionalen Raum simuliert.

Tipp:

08.3 Aufgabe:

Erstelle mit libGDX ein kleines Spiel, in dem der Spieler mit einer durch die Tastatur gesteuerten Abrisskugel ein Kartenhaus zerstören kann.