Optische Darstellung von Sortieralgorithmen

Ein Projekt für den Informatikunterricht am Gymnasium in Jahrgangsstufe 10

Inhaltsverzeichnis

Diese Arbeit wurde (in abgewandelter Form) als Zulassungsarbeit für das 2. Staatsexamen im Fach Informatik (Bayern) im Jahr 2009 eingereicht.

1 Projektvorhaben und Projektbeschreibung

1.1 Projektvorhaben

Sortieren ist für Computer eine Hauptaufgabe. Schätzungen zufolge verbringen im Mittel etwa 25% der Computer weltweit ihre Zeit mit Sortieren. Wozu werden Daten im Computer denn überhaupt sortiert?

Stellen Sie sich vor, Sie müssten eine Telefonnummer suchen und das Telefonbuch wäre nicht sortiert! Das wäre sehr zeitintensives Unterfangen. Genauso geht es auch einem Computer, der eine Vielzahl von Datensätzen in Bruchteilen von Sekunden finden soll. Das funktioniert nur mit sortierten Daten [ABInf, Kapitel 2]. Oder stellen Sie sich vor, Sie suchen in ihrem Dateimanager nach einer bestimmten Datei oder in Ihrem Mailprogramm nach einer bestimmten Mail (Auf- / Absteigende Sortierung nach Betreff, Absender, Datum, Größe). Sortieren ist also ein wichtiges Thema der Informatik.

Bereits in der Jahrgangsstufe 7 bietet es sich unter NT 7.2.3 „Beschreibung von Abläufen durch Algorithmen“ an, auf einen Sortieralgorithmus wie z.B. Bubblesort einzugehen. Dabei kann man den Sortieralgorithmus anhand der für Siebtklässler bekannten Programmierumgebung Robot Karol optisch darstellen. Dies ist technisch allerdings nicht mit dem Programmpaket Robot Karol möglich, sondern nur mit der java-basierten Programmierumgebung Javakarol. Optisch ergibt sich für den Schüler allerdings kein Unterschied.

Der Lehrplan für die Jahrgangsstufe 10 sieht unter INF 10.3. ein komplexeres Anwendungsbeispiel vor, das den Jugendlichen die Möglichkeit bietet zu erfahren, dass erst das Zusammenspiel ihrer bisher erworbenen Kenntnisse und konstruktives Arbeiten im Team es erlauben, schwierige Aufgabenstellungen zu bearbeiten.

Da in der Jgst 10 Java bereits als Programmiersprache eingesetzt wird, besteht hier also die Möglichkeit, Javakarol einzusetzen und damit weitere Sortieralgorithmen optisch darzustellen. Die Schüler können in Gruppenarbeit verschiedene Sortieralgorithmen übernehmen und ihre bisher erworbenen Kenntnisse praktisch einsetzen (Lehrplan INF 10.1.2 „Zustände von Objekten und algorithmische Beschreibung von Abläufen“). Im Rahmen eines größeren Projekts mit vorhandener Projektstruktur und Organisation werden die von den Gruppen erstellten optischen Darstellungen der Algorithmen zu einem Gesamtprogramm zusammengeführt.

1.2 Projektbeschreibung

Ziel des Projektes ist es, mehrere Sortieralgorithmen optisch darzustellen. Dazu wird eine Mauer mit mehreren Mauerelementen der Größe nach von einem Roboter sortiert.

Der Benutzer des Programms soll alleine durch Beobachtung des Roboters und der Mauerelemente die Funktionsweise der Algorithmen verstehen können.

Die Klasse wird dazu in mehrere Gruppen (zu 4 Personen) aufgeteilt. Jede Gruppe übernimmt einen Sortieralgorithmus und kümmert sich um die Implementierung, die Bewegungen des Roboters und der Mauerelemente sowie die Sprachmeldungen des Roboters.

Der von den Gruppen erzeugte Quellcode wird anschließend in einem Programm zusammengefügt. Über eine GUI kann der Benutzer die implementierten Algorithmen auswählen, die Geschwindigkeit des Roboters einstellen und weitere Aktionen durchführen, wie z.B. den Sortiervorgang zurücksetzen, eine andere Mauer zufällig erstellen lassen oder selbst eine Mauer vorgeben.

Die GUI wird (unter anderem) im Rahmen eines BlueJ-Projektes vom Lehrer zur Verfügung gestellt. Die Erstellung der GUI sowie das zugehörige Event-Management wird im Kapitel 2 „Programmtechnische Vorarbeiten“ erläutert. Der Schüler bekommt dies zu keinem Zeitpunkt zu sehen, da es für ihn nicht relevant ist.

Die Schüler sollen den Sortieralgorithmus ihrer Gruppe soweit verstehen, dass sie in der Lage sind anderen Personen (ohne IT-Vorkenntnisse) die Funktionsweise des Algorithmus zu erklären.

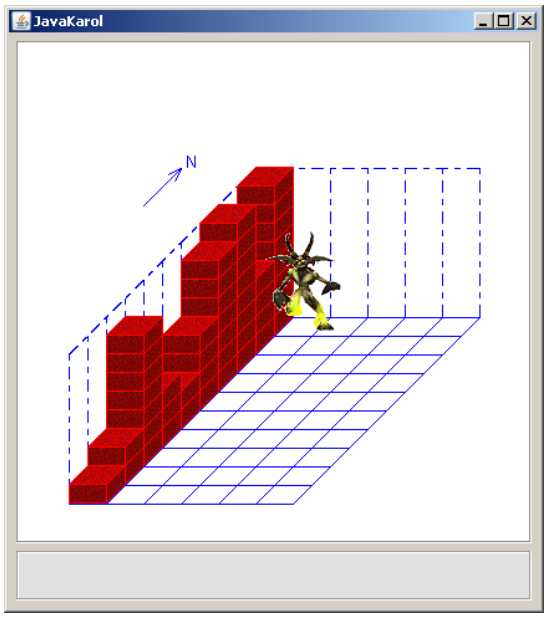
Der Java-Quellcode der Algorithmen soll dabei nicht selbst geschrieben werden. Die Schüler erhalten die Anweisung den Java-Quellcode aus dem Internet oder aus Büchern zu übernehmen, da dies den geplanten zeitlichen Rahmen (Lehrplan 10 Stunden) sprengen würde und es auch wenig sinnvoll erscheint, die Schüler in der Jgst 10 einen Sortieralgorithmus programmieren zu lassen.

2 Programmtechnische Vorarbeiten

Dieses Kapitel beschreibt die Vorarbeiten, die nötig sind, um dem Schüler zu Beginn der Unterrichtssequenzen ein funktionsfähiges Programmpaket zur Verfügung zu stellen. Dies beinhaltet vor allem die Erstellung einer GUI (mit Netbeans) sowie die Verlagerung der Applikationslogik (hier der Sortiervorgang und dessen optische Darstellung) in einen eigenen Thread. Es richtet sich primär an Lehrkräfte, die ebenfalls mit der Entwicklung einer GUI konfrontiert werden. Es kann bei Bedarf auch übersprungen werden.

2.1 Javakarol als Programmpaket zur optischen Darstellung

Bereits in der Jgst 7 hat sich das Programmpaket Javakarol zur optischen Darstellung des Sortieralgorithmus Bubblesort bewährt. Deshalb liegt es nahe auch hier Javakarol zur optischen Darstellung zu verwenden (vgl. Javakarol URL [JK1], Javakarol-Handbuch [JK2]).



2.2 Das Programmkonzept von Javakarol

Javakarol besteht aus der Java-Archiv-Datei javakarol.jar, die über Tools-Preferences-Libraries in BlueJ eingebunden werden kann.

Die Archivdatei enthält zwei Verzeichnisse javakarol und imgs.

Im Verzeichnis imgs befinden sich für jeden der Roboter 1-5 jeweils vier Bilder (Ansicht Westen, Osten, Norden, Süden).

Im Verzeichnis javakarol befinden sich die beiden Klassen Roboter.class und Welt.class (ohne Quellcode, man beachte die Kleinbuchstaben).

Das Beispielprojekt BlueKarol weist die beiden neuen Klassen ROBOTER und WELT (mit Quellcode, man beachte die Großbuchstaben) auf, die von Roboter und Welt erben.

```
public class WELT extends Welt {...}
public class ROBOTER extends Roboter {...}
```

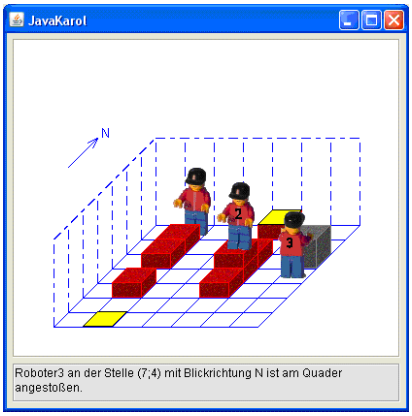
Der Programmierer arbeitet nun im folgenden ausschließlich mit den Klassen ROBOTER und WELT und kann dort (vgl. [JK2, S18ff])

1. alle öffentlichen Methoden der Klasse Roboter verwenden,
2. die Klasse ROBOTER um neue Methoden ergänzen.

2.3 Ersetzen der Roboter-Figur

Als Standardfigur wird in Javakarol ein Legomännchen verwendet. Dies erscheint mir einer zehnten Klasse altersmäßig nicht mehr angemessen. Darum wurde die Figur ersetzt durch einen Wichtel aus dem Massive-Multiplayer-Online-Rollenspiel World of Warcraft.

Die Figur wurde bei den Schülern sofort mit großer Begeisterung zur Kenntnis genommen.



2.4 Modifikation der Klasse Roboter

2.4.1 Die Methoden Aufheben(int n) und Hinlegen(int n)

Problematisch zeigte sich zunächst das Fehlen der Methoden `Aufheben(int n)` und `Hinlegen(int n)` im ursprünglichen Programmpaket, die den Roboter jeweils n Ziegelsteine gleichzeitig aufheben/hinlegen lassen. Im Programmpaket sind zwar die Methoden `Aufheben()` und `Hinlegen()` vorhanden; die beiden o.g. Methoden fehlen aber.

Die beiden fehlenden Methoden könnten nun durch Ableiten und anschließendes Erweitern einer Klasse ROBOTER, die alle Methoden von Roboter erbt, erstellt werden. Dies würde aber dazu führen, dass ein Objekt der abgeleiteten Klasse beim Aufruf von z.B. `Aufheben(5)` nun zwar 5 Bauklötze aufhebt, dies aber nicht gleichzeitig, sondern nacheinander tut. Dies ist für die Darstellung von Sortieralgorithmen allerdings nicht hilfreich, da ein Vertauschen von zwei Mauerelementen (unabhängig von der Höhe des Mauerelements) immer gleich lange dauern soll. Es wird also der Quellcode der ursprünglichen Klasse Roboter benötigt um dort die beiden Methoden hinzuzufügen. Die Klasse wurde dementsprechend modifiziert, neu kompiliert und anschließend die modifizierte Version ins Programmpaket eingefügt. Die Schüler erhalten die Javadokumentation zur modifizierten Klasse.

2.4.2 Die Methode Schritt(int n)

Die Klasse Roboter wurde auch um die fehlenden Methode `Schritt(int n)` ergänzt.

2.4.3 Die Methode MeldungAusgeben(String was)

Gibt ein Roboter über die Methode `MeldungAusgeben(String was)` eine Meldung aus, so wird in Javakarol zunächst ausgegeben, welcher Roboter die Meldung ausgibt und wo dieser Roboter gerade steht. Dies ist für unser Vorhaben eher störend, da wir sowieso nur mit einem Roboter arbeiten und es für die Meldung irrelevant ist, wo der Roboter gerade steht. Diese unnötigen Meldungen wurden ebenfalls durch Modifikation der Klasse Roboter entfernt.

2.4.4 Methoden zum Drehen des Roboters

Hinzugefügt wurden außerdem die Methoden `nachOstenDrehen()`, `nachWestenDrehen()`, `nachNordenDrehen()`, `nachSuedenDrehen()`.

Der Quellcode und die Dokumentation der orginalen und modifizierten Versionen der Klasse Roboter sind in folgenden Anhängen verfügbar:

[JK3] Roboter-orginal.html

[JK4] Roboter-orginal.java

[JK5] Roboter-modifiziert.html

[JK6] Roboter-modifiziert.java

2.5 Das Graphische User Interface (GUI) zum Steuern des Sortiervorganges

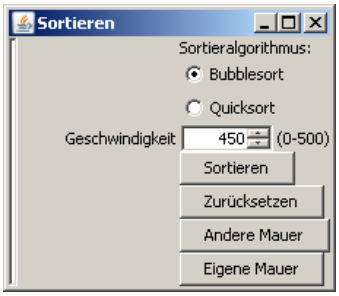
Da das Programm von Benutzern ohne IT-Vorkenntnisse bedient werden soll, liegt es nahe, eine GUI zu erstellen, die es dem Benutzer ermöglicht, die Programmsteuerung per Mausklick vorzunehmen. Da das Programm im Internet zur Verfügung gestellt werden soll, erscheint eine GUI hier unumgänglich.

Das Package javax.swing stellt alle Klassen bereit, die nötig sind um eine GUI zu erzeugen. Innerhalb des Java-Tutorials gibt es einen Abschnitt über das Thema „Creating a GUI with JFC/Swing“ [Swing] („Das Swing Tutorial“).

Folgende Anforderungen werden an die GUI gestellt:

1. Auswahl des Sortieralgorithmus durch Anklicken eines jRadioButtons
2. Auswahl der Sortiergeschwindigkeit durch einen jSpinner
3. Start des Sortiervorganges durch Anklicken eines jButtons
4. Unterbrechen und Wiederaufnehmen des Sortiervorganges
5. Zurücksetzen der Welt und der Mauer in den Ursprungszustand um evtl. einen anderen Sortieralgorithmus auf die gleiche Mauer anzuwenden.
6. Zufälliges Erstellen einer anderen Mauer.
7. Erstellen einer eigenen Mauer bzw. Festlegung der Höhe der Mauerelemente (Zusatzaufgabe)

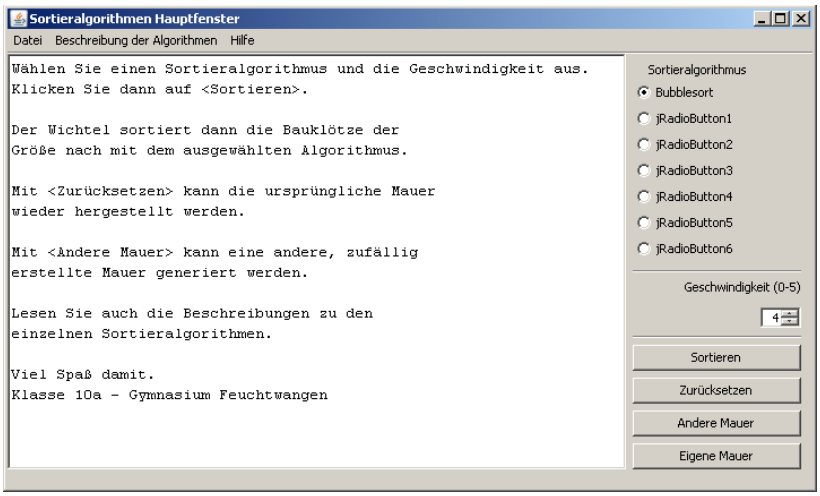
Zunächst wurde versucht, eine entsprechende GUI aus den einzelnen Swing-Komponenten mittels der verfügbaren Layout-Manager (per Hand) zu bauen. Dies funktionierte zwar, war aber trotz enormen Zeitaufwand optisch nicht ansprechend.



In [JT4] findet man folgenden Hinweis:

> **Note:** This lesson covers writing layout code by hand, which can be challenging. If you are not interested in learning all the details of layout management, you might prefer to use the `GroupLayout` layout manager combined with a builder tool to lay out your GUI. One such builder tool is the NetBeans IDE. Otherwise, if you want to code by hand and do not want to use `GroupLayout`, then `GridBagLayout` is recommended as the next most flexible and powerful layout manager.

Die Verwendung von Netbeans [NB1] als GUI-Builder führt bei erheblich geringerem Zeitaufwand zu einem qualitativ hochwertigeren Ergebnis.



2.6 Event-Management - der Event-Dispatch Thread und SwingWorker

Da das Hauptprogramm durch eine GUI gesteuert wird, kommen automatisch Event-Listener ([JT3]) wie z.B. ActionListener für die jButtons oder ChangeListener für den jSpinner ins Spiel.

Wir setzen als bekannt voraus, dass einer GUI-Komponente ein Event-Listener zugewiesen werden kann, der dann bei Betätigen der GUI-Komponente eine entsprechende Methode aufruft.

Beispielsweise erhält der Sortieren-Button durch den Befehl `sortierenButton.addActionListener(guiListeners)` einen ActionListener, der bei Betätigung des Sortieren-Buttons die Methode `public void actionPerformed(ActionEvent e) {...}` aufruft.

Wichtig zu wissen ist hier, dass

1. Event-Listener-Code (z.B. Code in actionPerformed()) wird in einem speziellen Thread ausgeführt (Event-Dispatch-Thread).
2. Code, der im Event-Dispatch-Thread abläuft, muss sehr schnell ablaufen und beendet werden, da während der Codeausführung die GUI blockiert wird.

> Swing event handling code runs on a special thread known as the event dispatch thread. Most code that invokes Swing methods also runs on this thread. This is necessary because most Swing object methods are not "thread safe": invoking them from multiple threads risks thread interference or memory consistency errors. [JT1]
>
> Tasks on the event dispatch thread must finish quickly; if they don't, unhandled events back up and the user interface becomes unresponsive. [JT1]

Würde man also (was zunächst nahe liegt) den Code für den Sortieren-Button so schreiben, dass in der actionPerformed-Methode des Sortieren-Buttons eine Methode zum Sortieren der Mauer aufgerufen wird, so würde die Sortieren-Methode im Event-Dispatch-Thread ablaufen und die GUI wäre bis zum Abschluss des Sortiervorgangs (der hier abhängig von der gewählten Geschwindigkeit sehr lange dauern kann) blockiert. Der Benutzer hätte somit während des Sortiervorgangs auch nicht die Möglichkeit z.B. die Geschwindigkeit zu ändern oder den Sortiervorgang kurzzeitig zu unterbrechen.

Folgender Code ist deshalb zwar nahe liegend, aber nicht zu empfehlen.

```
public void actionPerformed(ActionEvent e) {...
   Mauer.sortierenMitBubblesort();
}
```

Dazu kommt, dass neben der Blockierung der GUI auch andere Fenster mit graphischen Elementen blockiert werden und in der Darstellung nicht aktualisiert werden. D.h. auch das Javakarol-Fenster wird, während die Methode `actionPerformed()` abläuft, nicht aktualisiert. Das Ergebnis ist demnach folgendes:

1. Der Benutzer klickt auf den Sortieren-Button
2. Die GUI wird blockiert (der Sortieren-Button bleibt gedrückt)
3. Das Javakarol-Fenster wird blockiert
4. Der Roboter steht an der Anfangsposition

5. Der Benutzer wartet bis zum Abschluss des Sortiervorganges (ohne dass er eine einzige Bewegung des Roboters oder der Mauerelemente sieht)
6. Plötzlich steht der Roboter an der Endposition und die Mauer ist sortiert
7. Die GUI wird freigegeben und kann wieder benutzt werden

Die Lösung des Problems ist hier die Verlegung des Codes zum Sortieren der Mauer heraus aus dem Event-Dispatch-Thread in einen zweiten Thread. Dies geschieht in der Klasse GUIListeners. Dort wird ein SortierenThread erzeugt. Damit bleibt die GUI auch während des Sortiervorgangs ansprechbar und der Benutzer sieht alle Bewegungen.

Zu empfehlen ist also grundsätzlich folgende Implementierung:

```
public void actionPerformed(ActionEvent e) {...
   SortierenThread = new Thread(new Runnable() {
     public void run() {
       Mauer.sortierenMitBubblesort();
     }
   });
   SortierenThread.start();
}
```

Das Java-Tutorial geht auch auf das o.g. Problem ein und empfiehlt für den Fall, dass ein Swing-Programm eine länger andauernde Aufgabe ausführen muss, die Erzeugung eines im Hintergrund laufenden Worker-Threads. Eine vorgefertigte Klasse, die u.a. auch die geregelte Kommunikation zwischen Worker-Thread und Event-Dispatch-Thread zur Verfügung stellt, ist die Klasse SwingWorker.

> When a Swing program needs to execute a long-running task, it usually uses one of the *worker threads*, also known as the *background threads*. Each task running on a worker thread is represented by an instance of `javax.swing.SwingWorker`. [JT2]

Für unser Vorhaben reicht die Erzeugung eines neuen Threads völlig aus. Die Verwendung der Klasse SwingWorker wäre hier aber auch möglich.

Treten im Event-Listener-Code länger andauernden Aufgaben auf, so müssen diese aus dem Event-Dispatch-Thread heraus in einen eigenen Worker-Thread verlagert werden.

Die hier beschriebene Problematik wird auch in allgemeinerem Kontext und mit verschiedenen Programmcode-Alternativen in [JCiP, 9.3. „Long-running GUI tasks] diskutiert.

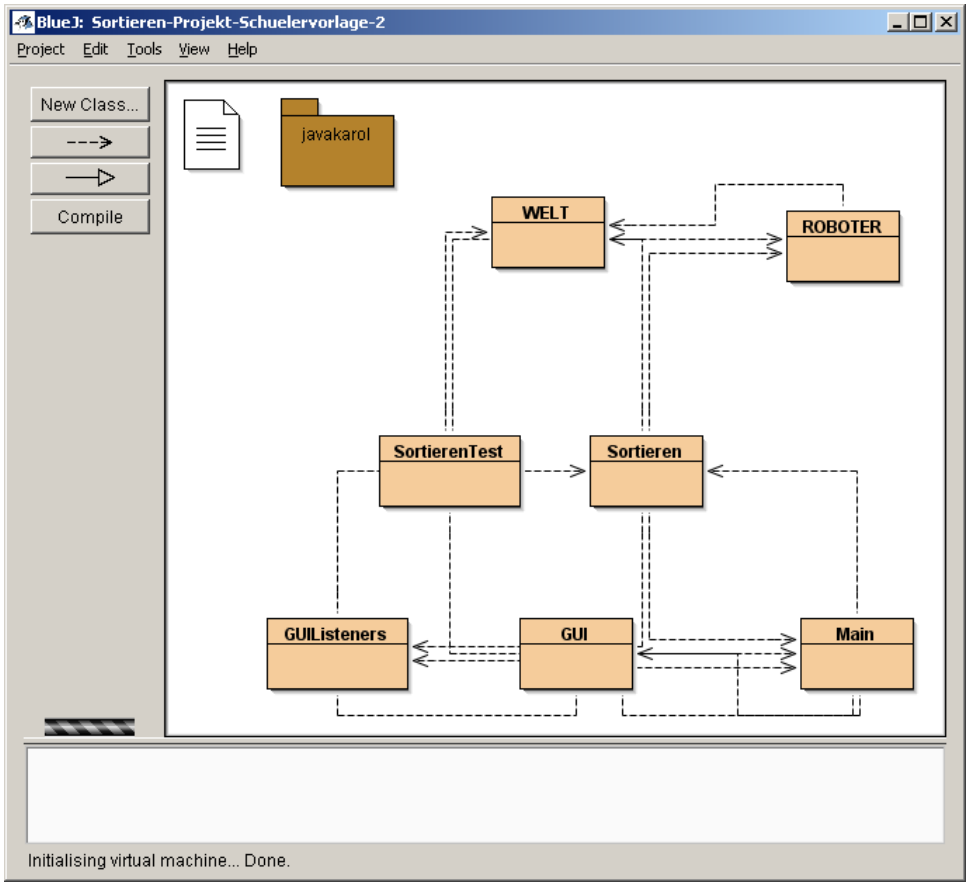
3 Die Schülervorlage als BlueJ-Projekt

Die hier aufgeführten Klassen des Programmpakets werden nur in Ausschnitten beschrieben. Es empfiehlt sich für die vollständige Dokumentation aus BlueJ heraus die Javadokumentation zu den Klassen über Tools - Project Documentation zu erstellen.

3.1 Dateistruktur der Datei Sortieren-Projekt-Schuelervorlage.zip

Sortieren-Projekt-Schuelervorlage\	GUI.java GUIListeners.java Main.java ROBOTER.java Sortieren.java SortierenTest.java WELT.java
doc\Roboter.html	Javadokumentation zur modifizierten Klasse Roboter.class
javakarol\	alle Dateien aus dem Programmpaket Javakarol inklusive der modifizierten Datei Roboter.class
imgs\	alle Bilder aus dem Programmpaket Javakarol inklusive der Bilder für den Wichtel

3.2 Ansicht in BlueJ



3.3 Arbeitsbereich der Schüler

Die Schüler arbeiten ausschließlich mit den Klassen SortierenTest (für den Test der Korrektheit des ausgewählten Algorithmus) und Sortieren (für die graphische Darstellung) sowie mit der Klasse ROBOTER.

3.4 Die Klasse javakarol/Roboter

Die modifizierte Klasse Roboter (vgl. [JK3-6]).

3.5 Die Klasse WELT

Eine Karolwelt ist ein rechteckiger, dreidimensionaler Raum, dessen Boden mit einem Quadratmuster ausgelegt ist und der von vier Wänden umgeben ist. In dieser Welt können sich mehrere Objekte der Klasse Roboter bewegen. Die Klasse WELT ist eine Unterklasse der Klasse Welt aus dem Paket javakarol.

Der Konstruktor erzeugt eine neue, leere Welt in der angegebenen Größe. Andere Konstruktoren wurden entfernt, da sie im Sortieren-Projekt nicht benötigt werden.

```
public class WELT extends Welt{
  public WELT(int breite, int laenge, int hoehe){
    super(breite,laenge,hoehe);
  }
```

3.6 Die Klasse ROBOTER

Die Klasse ROBOTER ist eine Unterklasse der modifizierten Klasse Roboter aus dem Paket javakarol.

Der erste Konstruktor erzeugt einen Roboter in einem Objekt der Klasse WELT (mit vorgegebener Startposition und Blickrichtung). Der zweite Konstruktor erzeugt einen Roboter in einer WELT an Position 1,1 und Blickrichtung 'S'.

Da die Figur ersetzt wurde, wird hier kein Legomännchen, sondern ein Wichtel erzeugt.

Die Schüler erhalten mit dieser Klasse die beiden Methoden

1. public void zumAnfangLaufen() : lässt den Roboter zum südlichen Ende der Mauer laufen und sich nach rechts drehen. Der Roboter steht dann vor dem ersten Mauerelement. Von dieser Position aus kann der Sortiervorgang gestartet werden.
2. public void nachLinksLaufen() : lässt den Roboter an das linke Ende der Welt laufen. Nach dem Sortiervorgang soll der Roboter an diese Position laufen.

3.7 Die Klasse GUI

Ein Objekt der Klasse GUI erzeugt ein jFrame mit den graphischen Komponenten zum Steuern des Sortiervorganges. Die Klasse wurde mit Netbeans erstellt und ist unkommentiert, da die Schüler diese Klasse nicht verändern müssen.

3.8 Die Klasse SortierenTest

Die Klasse ist zum Testen des von den einzelnen Gruppen ausgewählten Sortieralgorithmus da. Sie besitzt keine graphischen Elemente (wie z.B. Welt, Roboter) und kann dazu verwendet werden, den Algorithmus mit unterschiedlichen Arrays auf Korrektheit zu prüfen.

```
//Attribute
    int[] Mauer;
    int[] Mauerkopie;
    int Vertauschungen;
    int Vergleiche;
```

Gleichzeitig lernen die Schüler bereits hier, dass der Algorithmus das orginale Array Mauer nicht verändern darf, sondern dass ausschließlich die Kopie des Arrays Mauerkopie verändert werden darf.

Die Anzahl der Vergleiche und Vertauschungen werden jeweils als Attribut gespeichert und von den Schülern an dieser Stelle mitbehandelt, wenn der Algorithmus vergleichsbasiert ist. Eine Gruppe wählte beispielsweise den nicht vergleichsbasierten Algorithmus Bucketsort aus.

3.9 Die Klasse Sortieren

Die Klasse Sortieren weist neben den Attributen der Klasse SortierenTest ein Objekt der Klasse WELT und ein Objekt der Klasse ROBOTER auf.

```
    /** Attribute */
    ...
    public WELT meineWelt;
    public ROBOTER wichtel;
```

Nachdem die Schüler ihren Algorithmus in der Klasse SortierenTest auf Korrektheit überprüft haben, können sie den Quellcode des Algorithmus in die Klasse Sortieren übernehmen und anschließend zur optischen Darstellung des Algorithmus die Bewegungen des Roboters programmieren.

Die Schüler erfahren, dass, nachdem die Funktionsfähigkeit des Algorithmus hergestellt ist, sich weder der Roboter noch ein Mauerelement bewegt. Dazu sind geeignete Methoden zu programmieren, die in der Unterrichtssequenz „Wie der Wichtel laufen lernt“ gemeinsam entwickelt werden können.

3.10 Die Klasse GUIListeners

Die Klasse implementiert einen Action-Listener für sämtliche jButtons und einen ChangeListener für den jSpinner Geschwindigkeit. Die Klasse bildet die Schnittstelle zwischen GUI-Quellcode und Applikationslogik.

Sie übernimmt die Verlagerung des Sortiervorganges in einen eigenen Thread und speichert Status über den SortierenThread hinsichtlich darüber, ob der SortierenThread bereits vorher schon gestartet oder unterbrochen wurde in zwei boolschen Variablen.

```
//Attribute
    Thread SortierenThread;
    boolean threadWasAlreadyStarted = false;
    boolean threadSuspended = false;
```

Darüber hinaus übernimmt die Klasse die Umbenennung der jButtons und die Aktiv-Passiv-Schaltung der jButtons in Abhängigkeit vom jeweiligen Status des SortierenThreads. Beispielsweise darf der Button „andere Mauer“ nur dann aktiv (anklickbar) sein, wenn gerade kein Sortiervorgang läuft. Nach dem Sortiervorgang darf außer dem Zurücksetzen-Button kein jButton aktiv sein.

Die Klasse übernimmt auch den Aufruf des entsprechenden Algorithmus in Abhängigkeit vom ausgewählten jRadioButton sowie die Ausgabe von Textmeldungen auf die jTextArea.

In der finalen Version des Programms wurden die boolschen Variablen `threadWasAlreadyStarted` und `threadSuspended` entfernt, da der Status des Threads bereits implizit in den Beschriftungen der jButtons enthalten ist. Der Status wird stattdessen aus der Beschriftung ausgelesen.

3.11 Die Klasse Main

Die Klasse Main erzeugt beim Aufruf der main-Methode die GUI, fügt den GUI-Komponenten die Event-Listener aus GUI-Listener hinzu und erzeugt ein Sortieren-Objekt inklusive Javakarol-Fenster.

```
//Attribute
public final static String VERSION = "0.17.S";
public final static String DATE = "26.04.2009";
protected static GUI guiFrame;
public static GUIListeners guiListeners;
public static Sortieren sortieren;
```

Zum Starten des Programms muss die main-Methode in der Klasse Main aufgerufen werden.

Die Klasse stellt auch die Methode outMessage(String message) zur Verfügung, mit der die Schüler Meldungen auf die jTextArea in der GUI ausgeben können.

4 Praktische Durchführung in der Klasse

4.1 Zeitplanung

Thema	Anzahl Schulstunden
Unterrichtssequenz Arrays	(3)
Unterrichtssequenz: Einführung in Sortieralgorithmen mit Javakarol und Bubblesort	2
Festlegung der Gruppen auf einen Algorithmus, Klärung von Zuständigkeiten [Zust]	1
Unterrichtssequenz: Wie der Wichtel laufen lernt	2
Projektarbeit (Implementierung der Algorithmen in SortierenTest)	2
Projektarbeit (Implementierung in Sortieren)	1
Unterrichtssequenz: Wie der Wichtel nach Rechts laufen lernt	1
Projektarbeit (Wichtelmeldungen, Beschreibung des Algorithmus)	2
Präsentation der Ergebnisse Optimierungsarbeiten des zusammengefügten Programms	2
	Summe: 13 (16)

4.2 Unterrichtseinheit 0: Arrays

Es bietet sich an, dieses Projekt direkt nach der Unterrichtseinheit Arrays zu beginnen oder das Thema Arrays vor Beginn der Projektes noch einmal kurz zu wiederholen, da der sichere Umgang des Schülers mit Arrays das zügige Vorankommen im Projekt erheblich erleichtert.

Dazu besteht die Möglichkeit, das Thema Arrays innerhalb der Unterrichtseinheiten der ISB-Handreichung [ISBHR, S. 69 (Konzept 1)] durchzunehmen oder die selbst erstellte Unterrichtseinheit [UE-0] zu verwenden, die aus dem Kapitel über Arrays im Java-Tutorial entstanden ist (BlueJ-Projekt zu Arrays in [UE-0-1]).

Sowohl die ISB-Handreichung als auch die praktische Erfahrung in diesem Schuljahr zeigt, dass für die Einführung von Arrays (inklusive Übung und Verbesserung) mindestens 3 Schulstunden benötigt werden. Dabei bietet es sich an vor allem die Methode `vertausche(int i, int j, String[] anArray_start)` (Speichern eines Arrayelementes in einem Zwischenspeicher) ausführlich zu besprechen.

Das Thema Arrays wird hier also zur Vorbereitung angesprochen, ist aber nicht Thema und Inhalt des Projektes.

4.3 Unterrichtseinheit 1: Einführung in Sortieralgorithmen mit Javakarol und Bubblesort

4.3.1 Vorstellung von Bubblesort

Zu Beginn der Stunde wird der Algorithmus Bubblesort anhand des erzeugten Programmpaketes (GUI und Javakarol-Fenster) demonstriert. Des Wichtel sortiert die Mauerelemente und gibt entsprechende „Sprachmeldungen“ aus, die beschreiben, was er gerade macht.

Dabei wurden hier folgende Sprachmeldungen gewählt:

Zeitpunkt	**Meldung**
Start Algorithmus	"Das stand zwar nicht in meinem Vertrag, aber ok, dann sortiere ich Euch das Ding mal mit Bubblesort!"
Wichtel steht vor dem Mauerelement und überlegt:	"Muss das vertauscht werden? Hm!"
Wichtel beendet die Überlegung mit:	"Ja, das muss vertauscht werden!" oder "Nein, das muss nicht vertauscht werden!"
Wichtel hat einen Durchgang beendet und läuft wieder zum Anfang der Mauer:	"Jaja, immer die Kleinen vorschicken!"
Ende Algorithmus:	"Genug mit der Sklavenarbeit!"

Für die Demonstration ist eine Lehrerversion des Programmpaketes nötig, in der die Methoden für die Wichtelbewegungen [UE-2] bereits implementiert sind. Bei den Überlegungen des Wichtels werden zusätzlich jeweils die betroffenen Mauerelemente (Index und Höhe) ausgegeben.

Es empfiehlt sich, dabei die Geschwindigkeit sehr niedrig einzustellen und den Algorithmus mindestens 2 mal laufen zu lassen, damit die Schüler genau beobachten können, was der Wichtel tut und was er sagt.

An dieser Stelle bietet es sich an, die Frage an die Klasse zu stellen, wer der Meinung ist, die Funktionsweise des Algorithmus (nicht die Implementierung) verstanden zu haben. Dabei ist zu erwarten, dass sich ein großer Teil der Klasse meldet. Um das Verständnis zu überprüfen, soll ein Schüler noch einmal kurz zusammenfassen, wie der Algorithmus funktioniert. Die Schüler sollen erkennen, dass sie die Funktionsweise des Algorithmus alleine durch die Beobachtung der optischen Elemente im Programm verstanden haben.

Anschließend kann auf die jRadioButtons 1-7 eingegangen werden. Davon ist nur einer mit dem Namen des Algorithmus (Bubblesort) beschriftet. Die jRadioButtons 1-6 besitzen noch die Standardbeschriftung und weisen bei Auswahl keine Funktion auf. Die Schüler können hier schon ahnen, dass ihre Aufgabe darin besteht, die jRadioButtons 1-6 mit Leben zu füllen, d.h. mit entsprechenden Algorithmen, Wichtelbewegungen und Wichtelmeldungen zu versehen.

Anhand des Merkblattes [UE-1, Seite 1-2] können nun die Projektziele und die Hinweise zur Projektdurchführung mitgeteilt werden. An dieser Stelle kann die Projekthomepage mit allen nötigen Dokumenten und Links vorgestellt werden (vgl. Kapitel Projekthomepage).

Die Schüler sollen den Algorithmus nicht selbst implementieren, sondern dessen Funktionsweise anhand von Beispielen und durch Betrachten des Quellcodes nachvollziehen.

Dabei gilt es, das wichtigste Ziel hier entsprechend hervorzuheben: „Personen ohne IT-Vorkenntnisse sollen alleine durch Bedienung des Programms die Funktionsweise der einzelnen Algorithmen verstehen können."

Die Schüler sollen verstehen, dass hier nicht nur die Implementierung des Algorithmus eine Rolle spielt, sondern dass das Zusammenspiel von Algorithmus, Wichtelbewegungen und Wichtelmeldungen dazu führen soll, dass der Benutzer die Funktionsweise des Algorithmus nachvollziehen kann.

4.3.2 Gruppeneinteilung

Deshalb bietet es sich an, innerhalb der Gruppen verantwortliche Personen für diese Themen (Algorithmus, Wichtelbewegungen, Wichtelmeldungen) festzulegen. Eine Person ist mit der Leitung der Gruppe beschäftigt, sodass idealerweise Gruppen zu vier Personen gebildet werden sollten.

Im Falle der Klasse 10a konnte dies umgesetzt werden und 6 Gruppen a 4 Personen gebildet werden. Die Gruppen wurden nach bestimmten Kriterien zusammengesetzt und von der Lehrkraft vorgegeben. Jede Gruppe enthält dabei genau einen Schüler mit folgenden Fähigkeiten:

- sprachbegabt, kompromissbereit, konfliktfähig (Gruppenleiter mit social skills)
- fähig Aufgaben, die Java-Quellcode betreffen, schnell und sicher umzusetzen (Programmierprofi)
- mittelmäßige Begabung im Bereich Informatik (3er Schüler)
- unterdurchschnittliche Begabung (4-6er Schüler)

So können die Schüler ihre individuellen Stärken einsetzen und schwächere Schüler von ihren Teamkollegen profitieren. Die Verteilung der Zuständigkeiten stellt auch sicher, dass nicht einem Schüler alle Arbeit überlassen wird.

4.3.3 Wozu Sortierverfahren?

Zu diesem oder einem vorhergehendem Zeitpunkt kann eventuell die Frage aufkommen, warum und wozu man unterschiedliche Sortieralgorithmen benötigt und warum nicht einer alleine ausreicht.

Das Kapitel 2 in [AbInf] liefert dazu sehr schöne Beschreibungen und verständliche Beispiele. Es kann an dieser Stelle direkt im Unterricht zum Einsatz kommen.

Zusätzlich können folgende Begründung/Demonstrationen angebracht werden.

- Beispiel 1: Windows-Explorer (Auf- / Absteigende Sortierung nach Name, Größe, Typ, Geändert am)

- Beispiel 2: Mailclient Mozilla Thunderbird (Auf- / Absteigende Sortierung nach Betreff, Absender, Datum, Größe)
- Dabei ist darauf hinzuweisen, dass die Zeichenketten zunächst in Zahlen umgewandelt werden (z.B. ANSI-Codierung [Kle2, S. 32/3]), dann sortiert wird und anschließend die Umwandlung der Zahlen in Zeichenketten passiert.
- Bubblesort funktioniert zwar immer, ist aber einer der langsamsten Sortieralgorithmen. Hier können die Algorithmen Bubblesort und Quicksort, die beide auf ein sehr einfach gehaltenes Array (z.B. {4,7,5,8,3}) angewendet werden, verglichen werden (Demonstration an rechter/linker Tafel) und die Anzahl der Vergleiche und Vertauschungen notiert werden.

Quicksort

					C	M
4	7	5	8	3		
	⇧			⇧	3	1
4	3	5	8	7	2	2
⇧			⇧			
Summe					**5**	**3**

Bubblesort

					C	M
4	7	5	8	3	2	1
4	5	7	8	3	2	1
4	5	7	3	8	3	1
4	5	3	7	8	2	1
4	3	5	7	8	1	1
3	4	5	7	8		
Summe					**10**	**5**

(C = Comparisons/Vergleiche, M = Movements/Bewegungen)

- Hier ist es auch möglich, die Schüler selbst (auch fachfremde) Anwendungsgebiete von Sortieralgorithmen finden zu lassen. Allerdings würde dies etwas vom ursprünglichen Thema wegführen.

Mit jedem der genannten Beispiele wird den Schülern die Relevanz des Themas im realen Leben deutlich, was die Motivation, die Algorithmen zu durchschauen deutlich fördert.

4.3.4 Arbeitsauftrag: Vertrautmachen mit Javakarol

Die Schüler erhalten folgenden Arbeitsauftrag (Beamer):

- Entpacke das Programmpaket (Projektordner)
- Erstelle eine Karol-Welt (Rechtsklick auf Klasse WELT, Dimensionen der Welt müssen angegeben werden)
- Erstelle einen Roboter in dieser Welt (Rechtsklick auf die Klasse ROBOTER, Name des Welt-Objektes - meist wELT1 -muss angegeben werden)
- Steuere den Roboter durch die Welt durch Aufruf der entsprechenden Methoden (Rechtsklick auf das Roboter-Objekt)
- Beachte dabei auch die Methoden, die im Menü unter „inherited from Roboter“ zu finden sind.
- Lies dir die Methodenübersicht des Roboters (Roboter.html) durch, die auf der Projekthomepage und in deinem Projektordner im Unterverzeichnis doc/ zu finden sind.
- Lass den Roboter 5 Ziegel hinlegen, mache den Roboter unsichtbar, finde den Unterschied zwischen den Methoden NachOstenDrehen() und istBlickOsten() heraus.

4.3.5 Auswahl der Sortieralgorithmen durch die Gruppen

Nachdem sich die Schüler mit Javakarol vertraut gemacht haben, erhalten sie den Auftrag, sich innerhalb der Gruppe auf einen Sortieralgorithmus zu einigen [UE-1, Seite 3]. Dadurch schulen die Schüler zugleich Qualitäten wie Kompromissbereitschaft und Teamgeist und arbeiten sich eigenständig in die verschiedenen Algorithmen ein.

Die URL der gelungenen Wikipedia-Seite zu Sortieralgorithmen [WiSo] wird an die Schüler weitergegeben. Gleichzeitig erhalten die Schüler Kopien aus [OtWi] mit Beispielen und Beschreibungen zu den Algorithmen: Auswahlsort (Seite 66-69), Einfügesort (Seite 69-71), Shellsort (Seite 71-73), Bubblesort (Seite 73-76), Quicksort (Seite 76-89), Heapsort (Seite 89-96) und Mergesort (Seite 96-104) als Entscheidungshilfe.

4.3.6 Der Java-Quellcode zu Bubblesort

Zum Abschluss der Stunde kann der Java-Quellcode von Bubblesort durchgegangen werden [UE-1, Seite 4]. Dabei kann ggf. die Kurzversion am Beamer präsentiert werden und die Schüler sollen erklären, wie der Quellcode mit den Wichtelbewegungen aus der Demonstration zum Stundenbeginn in Einklang zu bringen ist. Anschließend wird die Seite 4 inklusive der kommentierten Version zur Ergebnissicherung ausgeteilt.

4.3.7 Folgestunden

Es bietet sich an, nach dieser Doppelstunde „Einführung in Sortieralgorithmen mit Javakarol und Bubblesort“ zunächst keinen neuen Stoff durchzunehmen, sondern stattdessen die Schüler in der Folgestunde selbständig arbeiten zu lassen. Dabei kann der bisher behandelten Stoff vertieft werden, die Festlegung der Gruppen auf einen Algorithmus und die Klärung von Zuständigkeiten [Zust] angegangen oder abgeschlossen werden, falls die Zeit in der Einführungsstunde hierfür nicht gereicht hat.

In den darauf folgenden Stunden sind zwei Ziele zu verfolgen:

1. Beschaffen des Quellcodes der Sortieralgorithmen (z.B. aus [OtWi] oder aus dem Internet) und Implementierung in der Klasse SortierenTest (Doppelstunde)
2. Unterrichtseinheit 2: Wie der Wichtel laufen lernt (Doppelstunde)

Die Reihenfolge kann hier grundsätzlich frei gewählt werden, da die beiden Einheiten unabhängig voneinander sind. Es ist hier Geschmackssache und liegt im Ermessen der Lehrkraft, ob zuerst 1. (Gruppenarbeit) oder 2. (Unterrichtsgespräch im Klassenverband) durchgeführt werden sollte. Eine sinnvolle Möglichkeit könnte es auch sein, die Schüler 45 Minuten lang mit 1. beginnen zu lassen und dann einen Methodenwechsel durchzuführen um 2. zu behandeln.

4.4 Unterrichtseinheit 2: Wie der Wichtel laufen lernt

Die Schüler erkennen, dass der Wichtel durch die Implementierung des Sortieralgorithmus selbst noch keinen Schritt läuft. Das Programmpaket Sortieren-Projekt-Schuelervorlage.zip [SPSV-1] enthält bisher keine Methoden, die den Wichtel zum Laufen bringen.

Innerhalb der Sortieralgorithmen wird es unabhängig vom verwendeten Algorithmus immer wieder nötig sein, dass der Wichtel zu bestimmten Mauerelementen läuft oder zwei Mauerelemente vertauscht. Ziel ist es daher, die entsprechende Methoden zur Verfügung

zu stellen:

Klasse Sortieren	
`private void zuMauerElementLaufen(int i)`	Der Roboter läuft zur Position i des Arrays / der Mauer.
`private void vertauscheMauerElemente (int i, int j)`	Der Roboter vertauscht die Elemente i und j im Array / der Mauer.

Betrachtet man die Javakarol-Welt und die Mauer (die intern in Form eines Arrays gespeichert wird) genauer, so stellt man u.a. fest, dass sich die Nummerierungsrichtungen der Welt und des Arrays unterscheiden (vgl. [UE-2]).

Daher ist es zweckmäßig zunächst die folgenden Methoden in der Klasse ROBOTER bereitzustellen:

Klasse ROBOTER	
`public void zuWeltPositionLaufen(int i)`	Der Roboter läuft zur Position i der Welt (nicht des Arrays).
`public void vertauscheWeltPositionen(int i, int j)`	Der Roboter vertauscht die Welt-Positionen i und j (nicht Array).

Den Schülern wird die hier sehr kurz beschriebene Problematik anhand der Grafiken in [UE-2, Seite 1] erklärt.

Anschließend können die Schüler im Unterrichtsgespräch Beiträge zur Implementierung der Methoden liefern. Dabei reicht es sicher aus, wenn die Schüler verbal beschreiben was der Roboter tun soll. Gemeinsam können dann mit Hilfe der Javadokumentation der Klasse Roboter die entsprechenden Methoden gefunden und in Quellcode umgesetzt werden.

Zur Entwicklung der Methode 1 `zuWeltPositionLaufen(int i)` im Unterrichtsgespräch werden die beiden nächsten Zeilen, sowie die Fallunterscheidung (1. Fall: `posRobo > i`, 2. Fall: `posRobo == i`, 3. Fall: `posRobo < i`) mit den entsprechenden Grafiken in [UE-2, Seite2] vorgegeben:

```
int i                                   Zielposition des Roboters
int posRobo = PositionYGeben();         Aktuelle Position des Roboters
```

Im dafür vorgesehenen freien Platz in [UE-2, Seite2] können nun die Vorschläge der Schüler am Beamer eingearbeitet werden und anschließend mit dem Quellcode am Ende der Seite (den die Schüler natürlich erst zum jetzigen Zeitpunkt sehen können, Scrollvorgang am Beamer) verglichen werden.

Die Schüler testen die eben entwickelte Methode direkt am Rechner auf Korrektheit.

In gleicher Weise können anschließend die noch ausstehenden Methoden 2-4 entwickelt und getestet werden. Alternativ dazu können nach dem Test der Methode 1 die Seiten 1-2 der ausgefüllten Version der Unterrichtseinheit verteilt werden [UE-2a] und der Auftrag

gegeben werden, einen Implementierungsversuch für die Methode 2 zu starten.

Die Schüler erhalten [UE-2a] vollständig zur Ergebnissicherung.

Zu diesem Zeitpunkt wird auf der Projekthomepage das um die vier Methoden erweiterte Programmpaket Sortieren-Projekt-Schuelervorlage-2.zip [SPSV-2] zur Verfügung gestellt.

4.5 Unterrichtseinheit 3: Wie der Wichtel nach Rechts laufen lernt

Für Sortieralgorithmen wie Bucketsort, und evtl. andere Verfahren wie Einfügesort, Mergesort oder Auswahlsort hat man die Anforderung, dass der Wichtel nach rechts an eine bestimmte Position läuft.

Gruppe 1 hat sich für Bucketsort entschieden und sich deshalb bereits im Unterricht mit dem Thema beschäftigt. Die Gruppe hat dabei intern diskutiert und einen Programmierversuch direkt am Rechner durchgeführt, der nur teilweise zum Ziel führte.

Diese Situation wurde genutzt, um zu verdeutlichen, dass ein Programmierversuch direkt am Rechner auch nur dann zielführend ist, wenn klar ist welche Funktion eigentlich programmiert werden soll. Stattdessen ist es vielleicht manchmal sinnvoller, wenn jedes Gruppenmitglied erst einmal für sich und in Ruhe (eventuell zu Hause) versucht, die Problemstellung zu konkretisieren. Dazu können Papier und Bleistift und ein 2-dimensionales Koordinatensystem (Karol-Welt) hilfreicher sein als die Entwicklungsumgebung am Rechner.

Dazu wurde (im Klassenzimmer, nicht im Computerraum) die Aufgabe gestellt, die Methode `zuMauerElementRechtsLaufen(int i)` mit Papier und Bleistift in zehnminütiger Stillarbeit zu formulieren. In [UE-3] findet sich auf Seite 1 eine konkretisierte Frage-/Aufgabenstellung, die nach Entwicklung im Unterrichtsgespräch verteilt werden kann, damit jeder Schüler die Problemstellung vor sich hat.

Zum Ende der Stillarbeit konnten 80% der Schüler die Lösung entweder direkt in Java-Quellcode oder in Pseudoquellcode angeben. Einige Schüler entwickelten sogar eine allgemeinere Lösung, die in Karol-Welten allgemeiner Größe funktioniert.

Die Verallgemeinerung der Methode `zuMauerElementLaufen(int i)` kann nach gleichem Muster im Unterricht behandelt werden.

Eine mögliche Lösung kann auf Seite 2 der Unterrichtseinheit [UE-3] zur Ergebnissicherung ausgeteilt werden.

4.6 Unterrichtseinheit 4: Eigene Mauer (Zusatzaufgabe)

Diese Zusatzaufgabe ist für Gruppen, die bereits vor der geplanten Zeit mit der Implementierung ihres Sortieralgorithmus fertig sind. Diese Aufgabe wirkte für einzelne Schüler als Ansporn und verhinderte Leerläufe bei zügig arbeitenden Schülern/Gruppen.

In der GUI gibt es einen Button "Eigene Mauer", der bei Betätigung noch keine Aktion ausführt. Ziel ist es hier dem Benutzer die Möglichkeit zu bieten, eine eigene Mauer einzugeben: Dazu wird die Höhe der Bauklötze (und evtl. auch die Länge der Mauer) vom Benutzer über Popup-Fenster abgefragt und vom Benutzer ein int-Array entgegengenommen. Anschließend wird ein neues Sortieren-Objekt mit diesem Array erzeugt.

Den Schülern wurde zu Projektbeginn sowohl mündlich als auch schriftlich in [UE-1] mitgeteilt, dass es diese Zusatzaufgabe gibt und dass weitere Hinweise auf einem gesonderten Blatt mitgeteilt werden, wenn die Schüler dies bei der Lehrkraft anfordern.

Kurz vor Beginn der Pfingstferien hat die Gruppe 2 die Zusatzhinweise angefordert, da sie bereits mit dem Pflichtteil des Projektes fertig waren (der von Gruppe 2 gewählte Algorithmus Bogosort ist relativ einfach). Den Schülern wurde daraufhin am 2. Juni 2006 (zunächst) über die Projekthomepage das Dokument [UE-4] mit zahlreichen Hinweisen zur Verfügung gestellt.

Es ist nicht zu erwarten, dass diese Aufgabe vollständig von einer Gruppe oder einzelnen Schülern alleine gelöst wird. Die Schüler werden darüber informiert, dass diese Aufgabe nicht leicht und nicht kurzfristig zu lösen ist, dass aber auch Teilergebnisse an die Lehrkraft weitergegeben werden sollen, da diese positiv in die Bewertung mit eingehen.

4.7 Möglichkeiten zur Optimierung des Unterrichtsablaufes

Es ist darauf zu achten, dass in der technischen Infrastruktur der Schule nicht bereits vor Projektbeginn das Programmpaket Javakarol als javakarol.jar im lib-Verzeichnis installiert ist, da es ansonsten zu Konflikten kommt.

Bei der Beschaffung des Java-Quellcodes für die Sortieralgorithmen durch die Schüler ist zu erwarten, dass der Code nicht sofort in das bereitgestellte Softwareprojekt integriert werden kann. Das bereitgestellte Softwareprojekt sieht eine einzelne Methode pro Sortieralgorithmus vor und der Quellcode aus dem Internet bezieht sich meist auf eine oder mehrere Klassen mit mehreren zugehörigen Attributen als globale Variablen. Es ist darauf zu achten, dass einzelne Gruppen nicht zu viel Zeit in die Integration des Quellcodes in das Projekt investieren, da den betreffenden Gruppen diese Zeit zu Projektende fehlen wird. Die Lehrkraft sollte hier steuernd eingreifen und ggf. Hilfen bei Implementierung geben.

Bereitgestellte Methoden können bei wiederholter Durchführung des Projektes sofort aussagekräftige benannt werden. Auf eine Umbenennung sowie auf ein Methodenverzeichnis kann somit verzichtet werden.

Trotz klarer Anweisung in [UE1] gilt es, die bereits besprochenen und schriftlich fixierten Ziele nicht aus den Augen zu verlieren. Beispielsweise hatten zahlreiche Gruppen die Wichtelmeldungen zunächst als Forum für lustige aber nicht zielführenden / erklärende Meldungen verwendet. Die Lehrkraft muss in diesem Fall darauf bestehen, dass diese Meldungen wieder entfernt werden und durch sinnvolle Meldungen ersetzt werden, die den Algorithmus verständlicher machen. Es bietet sich also an, die Zielsetzung in [UE1] während der Projektphase ab und zu zu wiederholen und den Schülern Rückmeldung zu geben, ob der aktuelle Entwicklungsstand voraussichtlich verwendbar sein wird.

Es stellte sich auch im Laufe des Projektes heraus, dass einzelne Gruppen es nicht schafften, die im Klassenverband erarbeiteten „Codeschnipsel“ aus den Unterrichtseinheiten in das laufende Programm zu integrieren. Dies allerdings nicht aus Unwissen, sondern offensichtlich eher aus einer gewissen Bequemlichkeit heraus. Auch in diesem Fall musste von der Lehrkraft immer wieder Konsequenz eingefordert werden.

Es bietet sich ebenso an, den Gruppen während der Projektphase immer wieder mitzuteilen, ob sie derzeit im aktuellen Zeitplan liegen, damit sie abschätzen können, ob es eventuell nötig ist, die Arbeiten außerhalb des Unterrichts fortzuführen.

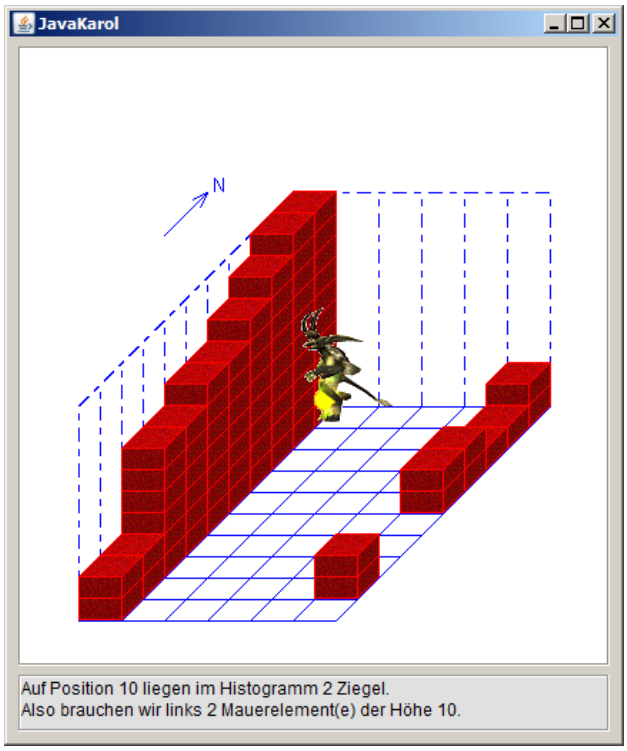
4.8 Fazit und Ausblick

Es hat sich gezeigt, dass dieses Projekt bei gewissenhafter Durchführung sowohl für den Lehrer als auch für den Schüler zwar zeitintensiv und anspruchsvoll, aber auch motivierend und Spaß bereitend ist.

Die Schüler hatten von vornherein die Anweisung, langfristig und kontinuierlich an dem Projekt zu arbeiten. Abgesehen von wenigen Motivationslöchern (bedingt durch die Temperatur im Computerraum in den Sommermonaten) war ein konzentriertes, produktives Arbeiten während der gesamten Projektphase möglich, sodass von der Klasse ein beachtliches Ergebnis erzielt werden konnte, das auf der Schulhomepage [HPpub] betrachtet werden kann.

Das Projekt forderte und förderte Ausdauer, Eigenständigkeit und Selbstverantwortung, Kompromissbereitschaft, phasenweise eine gewisse Frustrationstoleranz, Kontinuität bei der Arbeit und die Kommunikation sowohl in der Gruppe als auch mit der Lehrkraft. Die Aussicht auf die öffentliche Präsentation der Ergebnisse auf der Schulhomepage wirkte motivierend, da ein klar definiertes Ziel vor Augen stand.

Eine Fortführung des Projekts ist grundsätzlich sowohl im W-Seminar, als auch im P-Seminar denkbar. Im W-Seminar könnten die Laufzeiten der Algorithmen genauer untersucht und verglichen werden. Dabei könnte die Oh-Notation eingeführt und die untere Schranke für vergleichsbasiertes Sortieren diskutiert oder hergeleitet werden. Auch könnten Best-/Worst-/Average-Case Betrachtungen durchgeführt werden. Im P-Seminar könnte eventuell mit einem schulexternen Partner eine Anwendung als Weiterentwicklung des Programms erstellt werden wobei die exakten Anforderungen hier vom externen Partner formuliert werden sollten.



5 Anhang

5.1 Literaturverzeichnis

Hinweis: Die hier in eckigen Klammern angegebenen Literaturverweise können sich auf eine URL, eine Datei, ein klassisches Buch oder eine selbst erstellte Unterrichtseinheit beziehen. URLs und Dateien finden sich auf der beiliegenden CD-ROM im Verzeichnis Literatur. Sämtliche Unterrichtseinheiten finden sich sowohl gedruckt im Anhang dieser Arbeit als auch auf der beiliegenden CD-ROM im Verzeichnis Unterrichtseinheiten.

URLs und Dateien:

[HPpub] Öffentliche Homepage(s) zum fertigen Sortieren-Projekt
http://www.ulrichschneider.de/index.php?option=com_content&task=view&id=65&Itemid=92

http://www.schulweb-an.de/gymfw/index.php?option=com_content&view=article&id=361%3Aoptische-darstellung-von-sortieralgorithmen-projekt-10a&catid=148%3Ainformatik&Itemid=111&lang=de

[JK1] Javakarol Homepage
http://www.schule.bayern.de/karol/jkarol.htm

[JK2] Javakarol Handbuch

[JK3] Javadokumentation zur orginalen Klasse Roboter
Roboter-orginal.html

[JK4] Quellcode zur orginalen Klasse Roboter
Roboter-orginal.java

[JK5] Javadokumentation zur modizizierten Klasse Roboter
Roboter-modifiziert.html

[JK6] Quellcode zur modifizierten Klasse Roboter
Roboter-modifiziert.java

[JT1] Das Java Tutorial - The Event Dispatch Thread,
http://java.sun.com/docs/books/tutorial/uiswing/concurrency/dispatch.html

[JT2] Das Java Tutorial - Worker Threads and SwingWorker,
http://java.sun.com/docs/books/tutorial/uiswing/concurrency/worker.html

[JT3] Das Java Tutorial - Lesson: Writing Event Listeners,
http://java.sun.com/docs/books/tutorial/uiswing/events/index.html

[JT4] Das Java Tutorial - Lesson: Laying Out Components Within a Container,
http://java.sun.com/docs/books/tutorial/uiswing/layout/

[JTPD] Java Thread Primitive Deprecation, Sun, 1999,
http://java.sun.com/javase/6/docs/technotes/guides/concurrency/threadPrimitiveDeprecation.html

[JWS] Java SE Desktop Technologies,
http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp

[MeVz] Methodenverzeichnis
Methodenverzeichnis.pdf

[NB1] Netbeans Homepage
http://www.netbeans.org

[PHPS] Projekthomepage - Projekt 10 - Sortieren
http://www.sd-gymnasium.de/sortieren/ (passwortgeschützt)

[SPSV-1] Sortieren-Projekt-Schuelervorlage (ohne Wichtel-Methoden)
Programmpaket auf der CD-ROM im Ordner BlueJ-Sortieren-Projekt

[SPSV-2] Sortieren-Projekt-Schuelervorlage-2 (mit Wichtel-Methoden)
Programmpaket auf der CD-ROM im Ordner BlueJ-Sortieren-Projekt

[Swing] Trail: Creating a GUI with JFC/Swing - The Swing Tutorial
http://java.sun.com/docs/books/tutorial/uiswing/

[WiSo] Sortierverfahren - Wikipedia
http://de.wikipedia.org/wiki/Sortierverfahren

[Zust] Zuständigkeiten
Zuständigkeiten.pdf

Unterrichtseinheiten:

[UE0] Unterrichtseinheit: Arrays

[UE01] BlueJ-Projekt zu Arrays

[UE1] Unterrichtseinheit: Einführung in Sortieralgorithmen mit Javakarol und Bubblesort

[UE2] Unterrichtseinheit: Wie der Wichtel laufen lernt

[UE2a] Unterrichtseinheit: Wie der Wichtel laufen lernt - ausgefüllt

[UE3] Unterrichtseinheit: Wie der Wichtel nach Rechts laufen lernt

[UE4] Unterrichtseinheit: Zusatzaufgabe - Eigene Mauer

Bücher:

[AbInf] Gallenbacher, Abenteuer Informatik: IT zum Anfassen - von Routenplaner bis Online-Banking, Spektrum, 2. Auflage, 2008

[ISBHR] ISB-Handreichung „Informatik am Naturwissenschaftlich-technologischen Gymnasium Jahrgangsstufe 10“

[JCiP] Goetz, Bloch, Bowbeer, Lea, Holmes, Peierls, Java Concurrency in Practice, "Writing correct programs is hard; writing correct concurrent programs is harder ...", Addison-Wesley Longman, Amsterdam, 2006

[Kle2] Klett (Schülerbuch 9. Klasse), Informatik 2, Tabellenkalkulationssysteme, Datenbanken

[OtWi] Ottmann / Widmayer, Algorithmen und Datenstrukturen, Spektrum, 3. Auflage, 1996

5.2 Anlagen

Unterrichtseinheiten

1. Arrays [UE0]
2. Einführung in Sortieralgorithmen mit Javakarol und Bubblesort [UE1]
3. Wie der Wichtel laufen lernt [UE2]
4. Wie der Wichtel laufen lernt - ausgefüllt [UE2a]
5. Wie der Wichtel nach Rechts laufen lernt [UE3]
6. Zusatzaufgabe - Eigene Mauer [UE4]