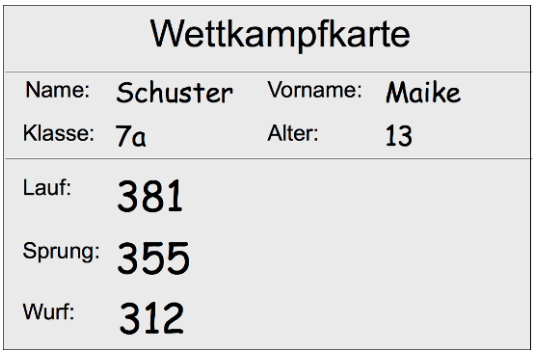
Projekt Bundesjugendspiele

1. Aufgabenstellung und erster Entwurf

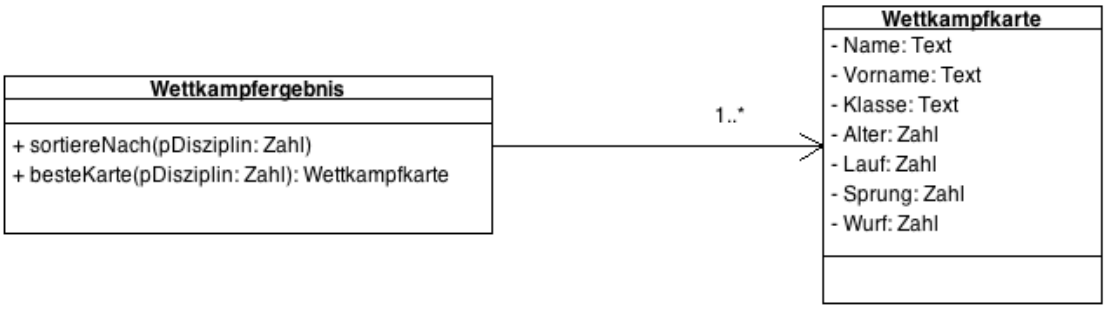
Jedes Jahr finden in Schulen bekanntlich die Bundesjugendspiele statt. Die Teilnehmer nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das Modell auf die drei Disziplinen „Lauf", „Sprung" und „Wurf" beschränken.



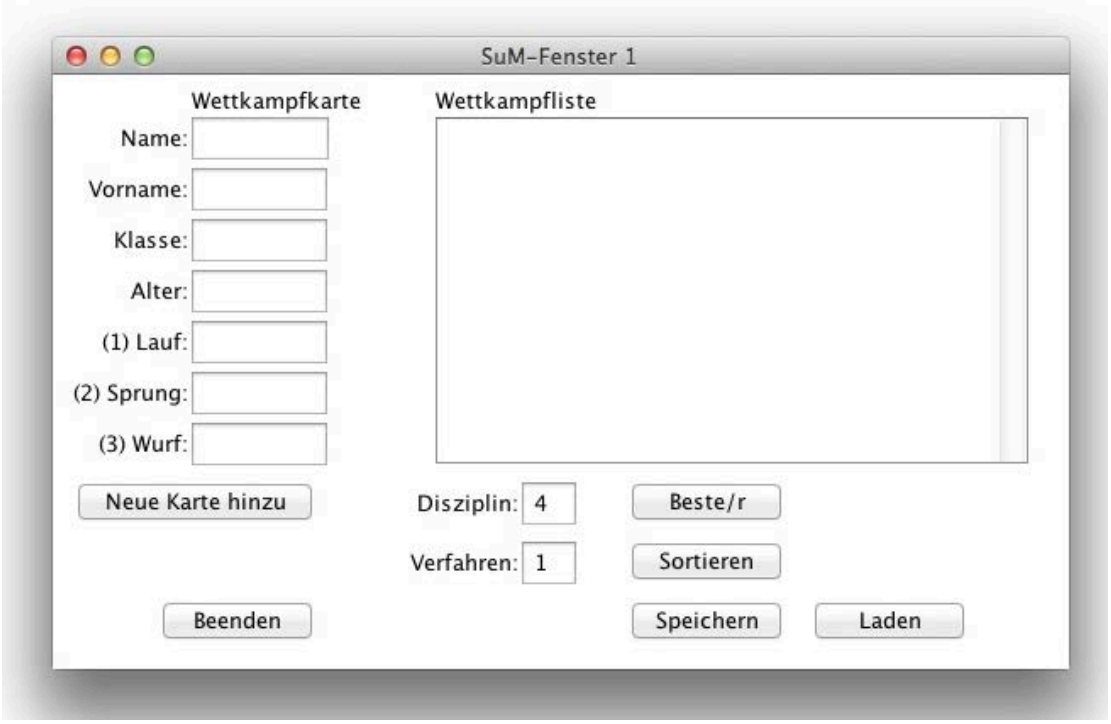
Im Wettkampfbüro wird das Ergebnis erstellt.

Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien sortieren können.

Der erste Entwurf spiegelt diese Festlegungen (Dienste des Ergebnisses und Attribute der Wettkampfkarte) in einem Entwurfsdiagramm wider:



Die Benutzungsoberfläche des Sortierprogramms kann vorgegeben werden. Auf jeden Fall benötigt das Fenster einen Bereich, indem die Information für eine Wettkampfkarte eingegeben oder dargestellt werden kann, einen Bereich, indem die sortierte Form aller Ergebnisse angezeigt werden kann und einen Bereich, der zur Steuerung dient. So kann die Oberfläche aussehen:



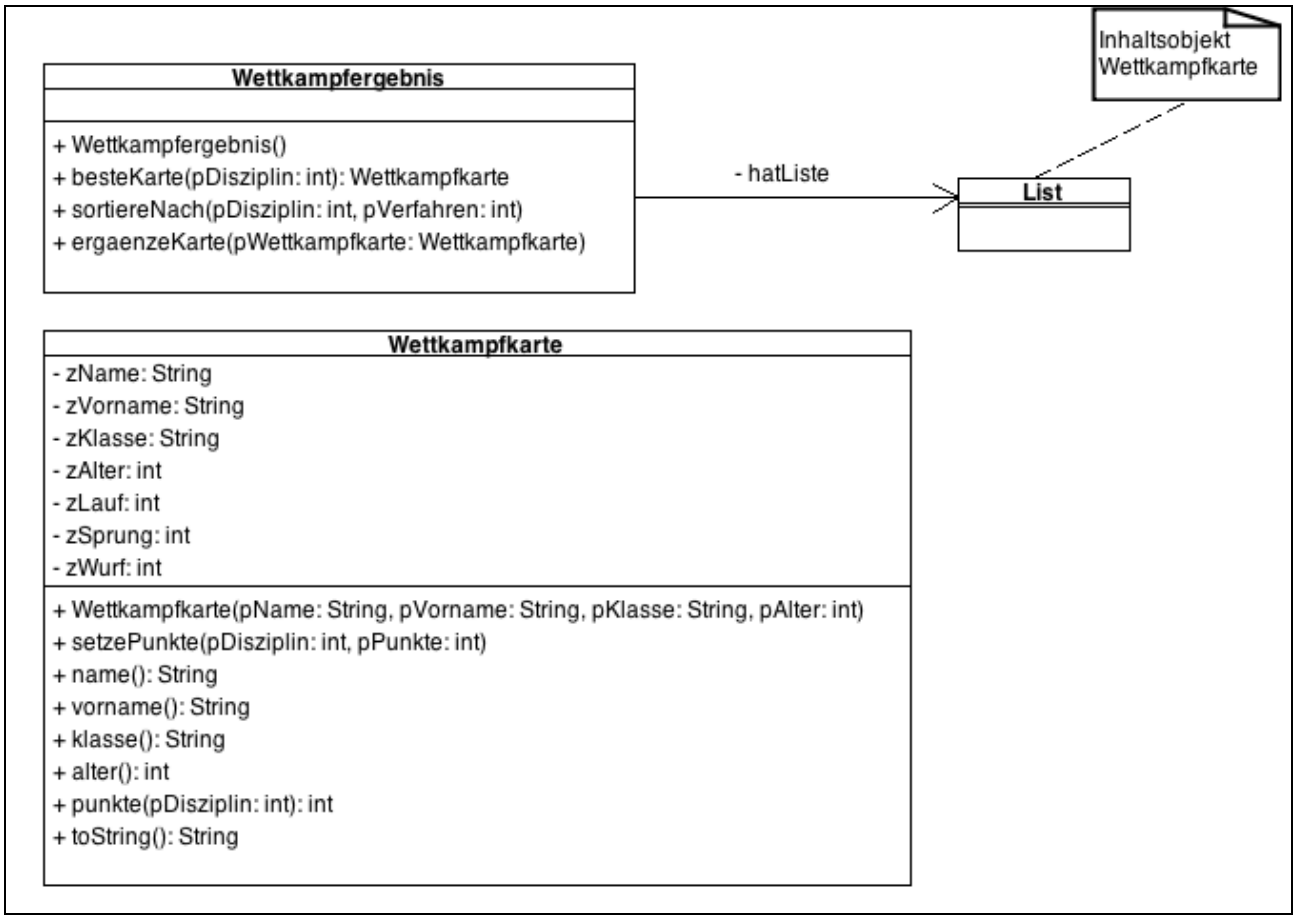
Hier finden sich als Ergänzung noch zwei Knöpfe für das Speichern und Laden der Karten, damit nicht bei jedem Testlauf alle Daten wieder eingegeben werden müssen. Diese Funktionen sollen aber in diesem Projekt als „Blackbox“ betrachtet und nicht thematisiert werden.

Die Disziplinen werden durchnummeriert, 4 steht für die Punktsumme. Ebenso werden gleich mehrere Sortierverfahren vorgesehen.

Das Implementationsdiagramm enthält gegenüber dem Entwurfsdiagramm weitere Festlegungen. Dabei wird auch die Benutzungsoberfläche mit einbezogen. Für das Wettkampfergebnis bedeutet das, dass die Methode sortiereNach das Verfahren als weiteren Parameter erhält und das eine Methode zum Ergänzen einer neuen Karte hinzugefügt wird.

Für die Assoziation zwischen dem Wettkampergebnis und den Karten wird jetzt als konkrete Struktur eine Liste verwendet.

Die Wettkampfkarte erhält nun Methoden. Dabei werden im Konstruktor nur die dauerhaften Daten erfasst, weil die Punkte ja im Laufe des Wettkampfs nach und nach dazukommen. Daher gibt es für die Punkte eine eigene Setze-Methode. Außerdem gibt es Anfragen für alle Attribute, wobei die Punkte wieder in einer Anfrage zusammengefasst werden. Schließlich liefert toString wie üblich eine textliche Darstellung des Objekts.



2. Entdecken von einfachen Sortieralgorithmen

Das Suchen der besten Karte ist so einfach, dass der Algorithmus nach einem kurzen Gespräch sofort klar ist. Dagegen sollte der Entwicklung der Sortieralgorithmen Zeit gegeben werden.

Mit dem folgenden Satz von Wettkampfkarten kann die Lerngruppe die Sortierverfahren selbstständig entdecken. Die Karten sind nummeriert, damit in einer arbeitsgleichen Gruppenarbeit alle Gruppen von derselben Kartenreihenfolge ausgehen.

1 Wettkampfkarte			
Name:	Schuster	Vorname:	Maike
Klasse:	7a	Alter:	13
Lauf:	381		
Sprung:	355		
Wurf:	312		

2 Wettkampfkarte			
Name:	Müller	Vorname:	Annika
Klasse:	7b	Alter:	13
Lauf:	313		
Sprung:	361		
Wurf:	289		

3 Wettkampfkarte			
Name:	Lange	Vorname:	Bärbel
Klasse:	7b	Alter:	13
Lauf:	271		
Sprung:	367		
Wurf:	255		

4 Wettkampfkarte			
Name:	Rotmann	Vorname:	Gerlinde
Klasse:	7a	Alter:	13
Lauf:	381		
Sprung:	395		
Wurf:	377		

5 Wettkampfkarte			
Name:	Müller	Vorname:	Tanja
Klasse:	7b	Alter:	14
Lauf:	330		
Sprung:	362		
Wurf:	285		

6 Wettkampfkarte			
Name:	Wafzinek	Vorname:	Olga
Klasse:	7b	Alter:	14
Lauf:	312		
Sprung:	320		
Wurf:	381		

7 Wettkampfkarte			
Name:	Bunse	Vorname:	Maria
Klasse:	7a	Alter:	13
Lauf:	377		
Sprung:	256		
Wurf:	325		

8 Wettkampfkarte			
Name:	Puhl	Vorname:	Cornelia
Klasse:	7a	Alter:	14
Lauf:	385		
Sprung:	336		
Wurf:	330		

Die Lernenden neigen oft dazu, die Karten intuitiv in einer Form zu ordnen, die ein Algorithmus für den Computer nicht nachvollziehen kann. Das lässt sich leicht verhindern, indem die Menschen wie der Computer nicht alle Karten gleichzeitig betrachten dürfen. Es gibt daher beim Entdecken der Algorithmen die Vorschrift, dass während des gesamten Verfahrens immer nur höchstens zwei Karten offen liegen dürfen, während alle anderen umgedreht sein müssen.

Ein weiteres Hilfsmittel ist eine Folie, auf der die Änderung der Sortierreihenfolge dargestellt werden kann und die Grundidee mit Hilfe farbiger Markierungen verdeutlicht werden kann. Die Folie findet sich wie die Wettkampfkarten in kleiner und großer Form in den Materialien.

Nach langjährigen Erfahrungen werden in allen Oberstufen- und Fortbildungskursen in der Gruppenarbeit von allen Gruppen mindestens zwei der drei Standardverfahren entdeckt,

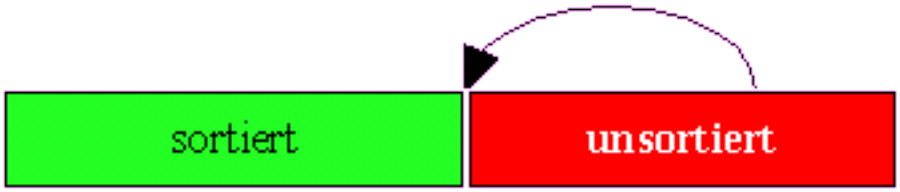
zusammen lassen sich immer alle Verfahren abdecken. Auch Quicksort-ähnliche Verfahren werden manchmal schon hier gefunden.

So lassen sich dann die drei Standardverfahren durch Gruppenvorträge darstellen. Als Materialien gibt es hierfür die Wettkampfkarten in großer Form für eine Magnettafel oder eine Pinnwand. Dazu kann außerdem die Folie projiziert werden.

In einer Ergebnissicherung kann die Moderation die Grundideen in grafischer Form darstellen. Dabei wird jeweils ein sortierter und ein unsortierter Bereich dargestellt:

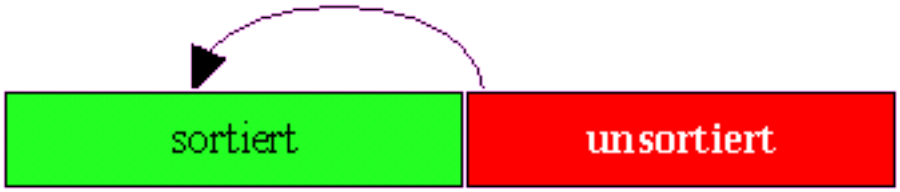
Sortieren durch Auswahl

Es wird die beste Karte aus dem unsortierten Bereich herausgenommen und an das Ende des sortieren Bereichs angehängt.



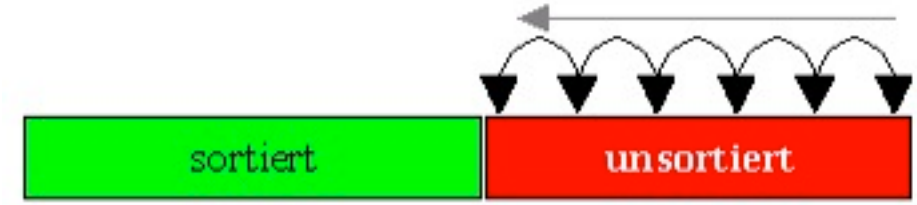
Sortieren durch Einfügen

Es wird die erste Karte aus dem unsortierten Bereich herausgenommen und an die richtige Stelle des sortieren Bereichs eingefügt.



Sortieren durch Austausch benachbarter Objekte (Bubblesort)

Im unsortierten Bereich werden von z. B. von hinten her benachbarte Karten verglichen und bei einer Fehlstellung vertauscht. Dadurch wandert die beste Karte an den Anfang des unsortierten Bereichs und kann dann in den sortierten Bereich verschoben werden.



Bei diesem Verfahren gibt es viele Modifikationen. Da die aktuelle Liste für das Zentralabitur ein Rückwärtsgehen nicht ermöglicht, kann man natürlich auch von vorne aus vergleichen, wobei dann jeweils am Ende die schlechteste Karte in den sortierten Bereich verschoben werden kann. Oder aber man teilt hier gar nicht in einen sortierten und unsortierten Bereich auf, sondern führt einfach solange Durchgänge durch die Karten durch, bis kein Tausch erfolgt ist. Dieses Verfahren ist natürlich für bereits sortierte Karten besonders effektiv.

3. Realisierung der Standardverfahren in Java

(Nicht alle im Folgenden beschriebenen Verfahren müssen berücksichtigt werden. Aus Zeitgründen empfiehlt es sich, sich auf eines der von den Schülerinnen und Schülern entwickelten Verfahren zu konzentrieren.)

Schon wegen des Speicherns und Ladens der Daten kann das Projekt von den Lernenden nicht komplett selbst erstellt werden. Aber auch aus anderen Gründen empfiehlt sich hier die Verwendung eines Prototyps. So können sich die Lernenden auf das Wesentliche beschränken und implementieren nur die Methoden besteKarte und sortiereNach der Klasse Wettkampfergebnis.

Dazu kann die Moderation nach dem Erstellen des Entwurfs den Prototyp mit vielleicht minimal anders genannten Methoden herausgeben oder den Prototyp auf den Entwurf anpassen.

Der Prototyp enthält eine eigene Variante der Klasse List, da die vom Ministerium gelieferte nicht benutzt werden kann, weil sie nicht abgespeichert werden kann. Außerdem wurde die Klasse List um die Methode length erweitert, die bei den Sortierverfahren sehr nützlich ist. In der Modelllösung wird sie (später) in den Verfahren Quicksort und Mergesort benutzt. Puristen, die sie nicht verwenden möchten, können auch ohne arbeiten.

Die Abfrage, ob die Länge > 1 ist, kann in beiden Fällen durch die Abfrage, ob die Liste nicht leer ist ersetzt werden. Dann ist die Rekursion nur etwas ineffektiver, weil sie jeweils einen Schritt länger läuft. Das Aufteilen der Listen in zwei Hälften bei Mergesort kann in anderer Form erfolgen, indem abwechselnd in die erste und die zweite Liste verteilt wird.

Hinweise zu den drei quadratischen Standardverfahren:

Das Sortieren durch Auswahl lässt sich besonders leicht realisieren, weil die beste Karte gesucht werden kann und so eine bereits realisierte Methode verwendet wird.

Sowohl das Sortieren durch Auswahl wie durch Einfügen lässt sich am einfachsten so realisieren, dass eine Hilfsliste benutzt wird. Es wird jeweils eine leere Hilfsliste erzeugt, dann in einer Schleife solange ein Element aus der Originalliste (unsortierter Bereich) in die Hilfsliste (sortierter Bereich) transportiert, bis die Originalliste leer ist. Danach wird der Originalliste die Hilfsliste zugewiesen.

Das Realisieren von Bubblesort gestaltet sich schwierig, seit die Liste für das Zentralabitur nicht mehr rückverkettet ist. Da ist das Vertauschen zweier benachbarter Elemente nur noch extrem ineffektiv möglich, indem das vorige Element durch einen erneuten Durchlauf bis zur entsprechenden Stelle bestimmt wird. Mit einem kleinen Trick kommt man ohne das vorige Element aus:

Das erste Element wird als „Blase“, die in der Liste aufsteigen kann, gemerkt und aus der Liste gelöscht. Nun wird es mit dem neuen aktuellen Element verglichen. Ist es „besser“ oder gleich, hat es die richtige Position und wird wieder vor dem aktuellen Element eingefügt. Ist es „schlechter“, hat es die falsche Position und ein Tausch muss erfolgen. Dazu rückt das einfach das aktuelle Element weiter und die Blase ist dadurch ein Element hochgerückt. Zum Schluss wird die Blase an die Liste angehängt. Hier arbeitet man dann die ganze Zeit auf der Originalliste.

4. Vergleich der Sortierverfahren

(Nicht alle im Folgenden beschriebenen Verfahren müssen berücksichtigt werden. Aus Zeitgründen empfiehlt es sich, sich auf Sortieren durch Einfügen, Quicksort und evtl. ein weiteres Verfahren zu konzentrieren, das von den Schülerinnen und Schülern entdeckt wurde.)

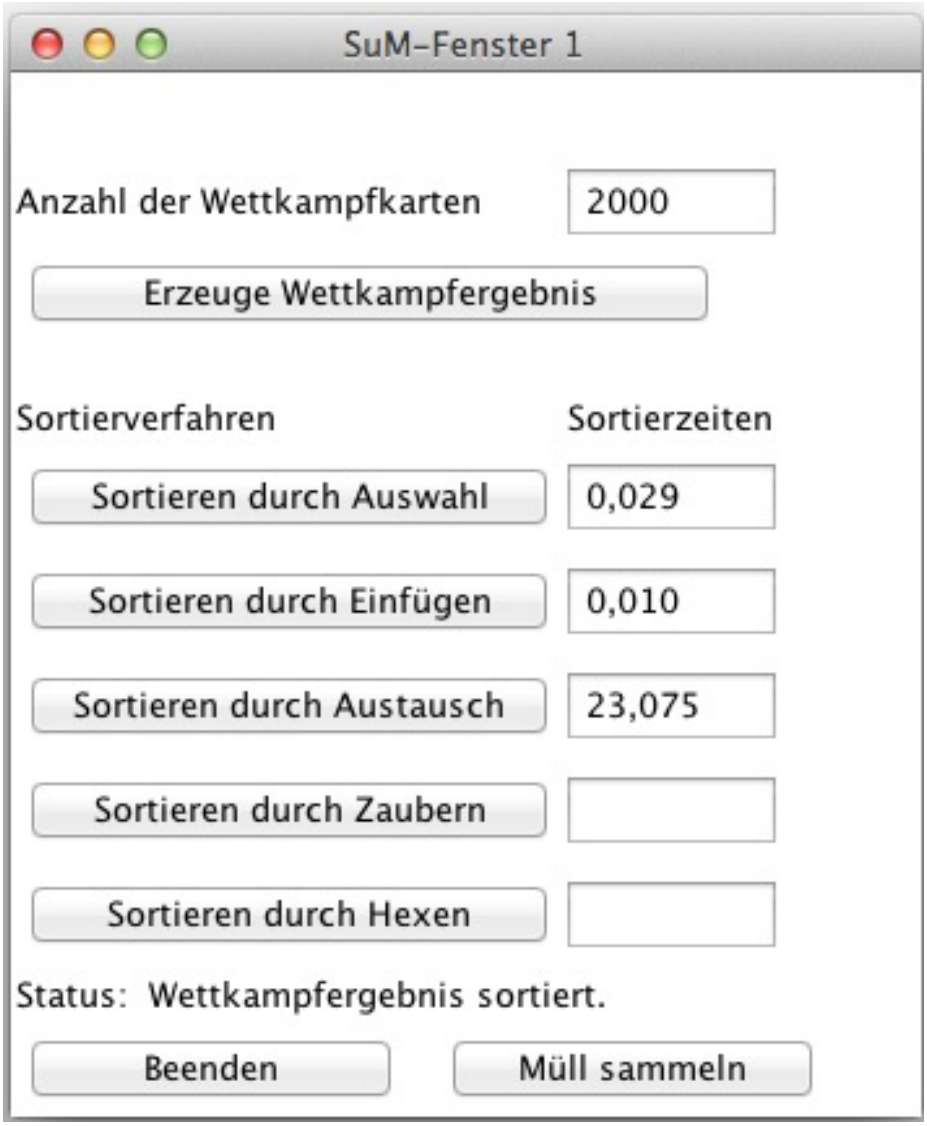
Hier werden sowohl die Verfahren sowohl praktisch verglichen wie auch theoretisch analysiert.

Verschiedene Sortierverfahren sind nun implementiert. Aber welches davon ist das beste? Welche Möglichkeiten gibt des so etwas herauszufinden?

Innerhalb einer Gruppenarbeit können die Lernenden vor einem praktischen Test über die Geschwindigkeit der drei Verfahren diskutieren. Neben ersten Vermutungen sollten dabei auch Begründungsversuche entwickelt werden.

Die Vermutungen können abgesichert werden, indem die Verfahren mit unterschiedlich vielen Wettkampfkarten praktisch erprobt werden. Natürlich ist es nicht sinnvoll diese Karten alle von Hand einzugeben. Vielmehr muss hier eine neue Anwendung erstellt werden, die diese Wettkampfkarten zufällig erzeugt. Weiterhin sollte dieses Programm auch die Sortierzeiten stoppen können. Ein solches Programm kann wahlweise von allen Lernenden oder von einer Teilgruppe entwickelt werden oder von der Moderation geliefert werden.

Die Tests sehen dann etwa so aus:



Man erkennt, dass Einfügen schneller als Auswahl arbeitet und Bubblesort deutlich langsamer ist. Wenn man genug Zeit hat, kann man auch parallel auch verschiedene Anzahlen stoppen, die Ergebnisse in eine Tabelle eintragen und dazu Graphen zeichnen lassen, die andeuten, dass es sich um quadratische Funktionen handeln.

Bevor man zur theoretischen Erörterung gelangt, können die Lernenden versuchen, die unterschiedlichen Zeiten anschaulich zu begründen, z. B.:

- Sortieren durch Austausch benachbarter Objekte ist so schlecht, da in jedem Durchgang viele Vertauschungen stattfinden. Bei den anderen beiden Verfahren ist dies nicht der Fall, da ja nur ein Objekt vom der unsortierten Liste in die sortierte Liste gebracht wird.

- Sortieren durch Auswahl ist schlechter als Sortieren durch Einfügen, da bei der Auswahl des besten Objekts der gesamte unsortierte Bereich durchsucht werden muss. Beim Sortieren durch Einfügen hingegen wird nur solange gesucht, bis die richtige Einfügeposition gefunden ist.

Grundsätzlich werden bei der theoretischen Analyse drei Fälle unterschieden:

- Verhalten im besten Fall (best case)
- Verhalten im schlechtesten Fall (worst case)
- Verhalten im mittleren Fall (average case)

Sei im Folgenden n die Anzahl der zu sortierenden Objekte. Es werden nun die Vergleiche gezählt.

Sortieren durch Auswahl

In jedem Durchgang muss die unsortierte Liste komplett durchlaufen werden um die beste Wettkampfkarte zu ermitteln. Allerdings wird die unsortierte Liste in jedem Schritt kleiner. Es ergibt sich folgende Reihe: $(n-1)+(n-2)+...+2+1=\frac{1}{2}(n-1)n=\frac{1}{2}n^2-\frac{1}{2}n$

Das Verfahren hat also für alle Fälle quadratischen Aufwand. Die übliche Schreibweise ist $O(n^2)$. Das bedeutet, dass es eine Konstante c und eine natürliche Zahl n_0 gibt, so dass für alle Zahlen $n > n_0$ gilt, dass der Aufwand $f(n) \leq c \cdot n^2$.

Sortieren durch Einfügen

In jedem Durchgang wird ein Objekt in die sortierte Liste eingefügt. Es ergibt sich folgende Reihe:

- Verhalten im besten Fall (best case)
  In diesem Fall kann das Objekt immer vor dem ersten Element der sortierten Liste eingefügt werden. Das ist der Fall, wenn die Originalliste umgekehrt sortiert ist. Es ergibt sich ein linearer Aufwand: 1 + 1 + ... + 1 = n

- Verhalten im schlechtesten Fall (worst case)
  In diesem Fall muss das Objekt immer an das Ende der sortierten Liste angehängt werden. Das ist der Fall, wenn die Originalliste bereits richtig sortiert ist. Es ergibt sich ein quadratischer Aufwand:
  $0+1+2+...+(n-2)+(n-1)=\frac{1}{2}(n-1)n=\frac{1}{2}n^2-\frac{1}{2}n$

- Verhalten im mittleren Fall (average case)
  Jedes Objekt wird im Durchschnitt etwa in die Mitte der sortierten Liste eingefügt. Hier ergibt sich ebenfalls wieder ein quadratischer Aufwand:
  $\frac{1}{2}\left(0+1+2+...+(n-2)+(n-1)\right)=\frac{1}{2}\cdot\frac{1}{2}(n-1)n=\frac{1}{4}n^2-\frac{1}{4}n$

Das Sortieren durch Einfügen hat also im besten Fall linearen Aufwand, im schlechtesten Fall denselben quadratischen Aufwand wie das Sortieren durch Auswahl. Es ist auch im mittleren Fall quadratisch, aber mit dem Faktor 0,5 gegenüber dem Sortieren durch Auswahl.

Sortieren durch Austausch

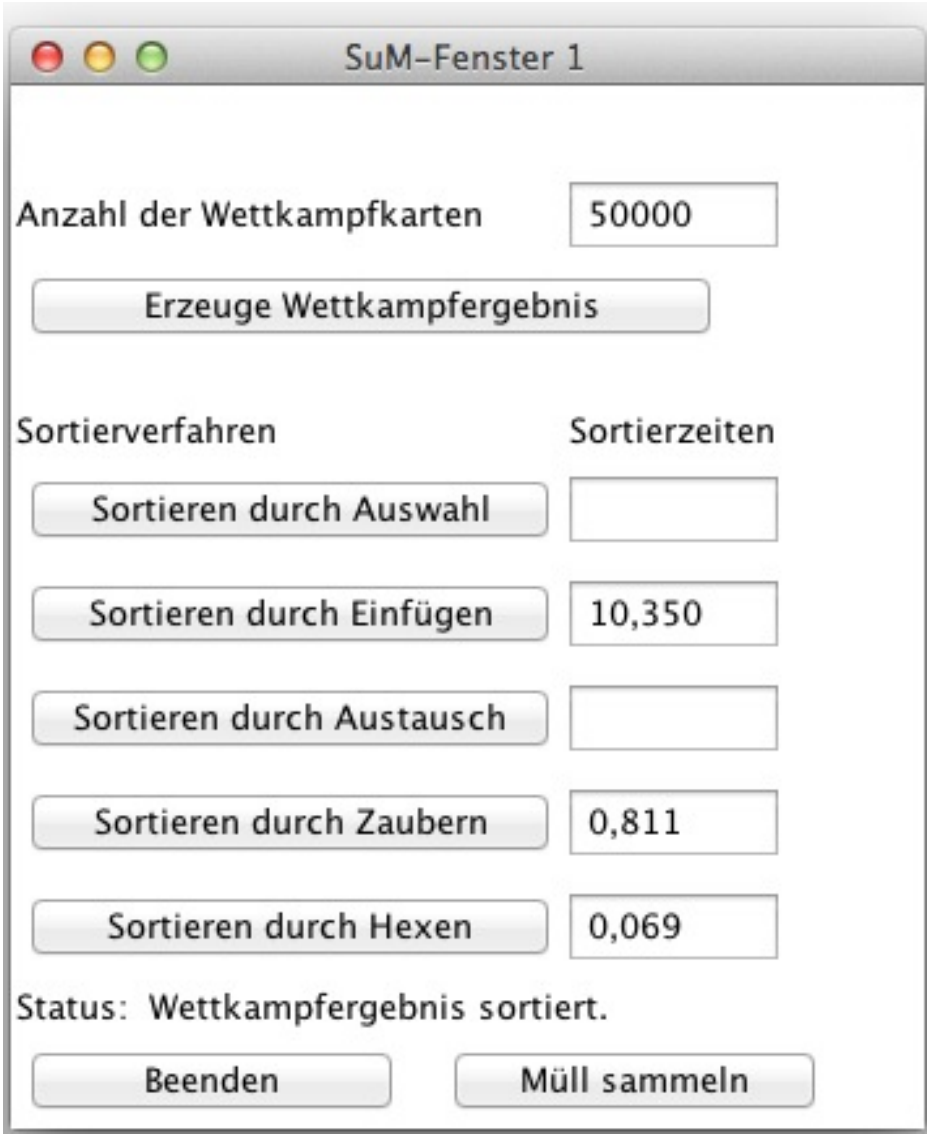
In jedem Durchlauf muss die unsortierte Liste komplett durchlaufen werden und jedes Objekt mit seinem Vorgänger verglichen werden. Die unsortierte Liste wird in jedem Schritt kleiner. Es ergibt sich wieder die Reihe wie beim Sortieren durch Auswahl. Allerdings muss hier noch die Zahl der Vertauschungen gerechnet werden, die auch mehr

Rechenzeit kosten als ein Vergleich. Für die Vertauschungen gibt es die drei Fälle wie beim Sortieren durch Einfügen, d. h. im besten Fall linear, im schlechtesten Fall dieselbe Reihe wie beim Vergleich und im mittleren Fall wieder die Hälfte dieser Reihe.

Insgesamt bleibt das Verfahren in der Summe der Vergleiche quadratisch, allerdings mit einem deutlich höheren Faktor als die beiden anderen Verfahren.

Das Fenster oben zeigt Bubblesort aber noch langsamer als nach dieser Berechnung zu erwarten wäre. Der Grund dafür ist die interne Realisierung des Einfügens in der Zentralabitur-Version der Liste. Dazu wird jedes Mal die Liste bis zum Auffinden des Vorgängers noch einmal durchlaufen. Damit wird eine dritte Durchlauf-Ebene geschaffen und durch diese Realisierung hat Bubblesort damit sogar kubischen Aufwand $O(n^3)$.

Da die aktuelle Realisierung der Liste im Falle Bubblesort nicht nur zu einem höheren Aufwand führt, sondern auch das Vertauschen nur noch durch „Tricks“ sinnvoll zu implementieren ist, sollte Bubblesort bei der Realisierung nur noch zur inneren Differenzierung dienen, um Lernenden, die sehr schnell programmieren, eine weitere Aufgabe zu geben.



Prognose

Da Informatik eine Frage der Größenordnung ist, sollten noch deutlich größere Datenmengen betrachtet werden. Dazu lässt sich aus dem obigen Beispiel eine Prognose berechnen.

Die Verfahren der selbst gefundenen Verfahren verhalten sich quadratisch, der Scheitelpunkt der Funktion liegt im Ursprung. Vernachlässigt man den linearen Anteil (was für große n sinnvoll ist), lässt sich die Laufzeit L durch die Funktion $L(n) = a\ n^2$ prognostizieren. Im obigen Beispiel ergibt sich dann a als $\frac{10,35}{50000^2}$. Mit einer Tabellenkalkulation lassen sich dann die zu erwartenden Laufzeiten für größere Datenmengen berechnen, die hier als Beispiel für Bevölkerungszahlen genommen werden:

Rechenzeiten beim Sortieren durch Einfügen						
Bevölkerung	**Anzahl**	**Sekunden**	**Minuten**	**Stunden**	**Tage**	**Jahre**
Schwerte	50000	10,35	0,1725	0,002875	0,000119792	3,2820E-007
Dortmund	600000	1490,4	24,84	0,414	0,01725	4,7260E-005
Deutschland	80000000	26496000	441600	7360	306,6666667	0,840182648
Welt	7000000000	2,029E+011	3381000000	56350000	2347916,667	6432,648402

Die Rechenzeiten gehen bei großen Zahlen deutlich in die Höhe. Für die Einwohnerzahl von Deutschland erhält man fast ein Jahr, für die Weltbevölkerung etwa 6500 Jahre. Diese Rechenzeiten sind nicht mehr tragbar, auch wenn man 10-mal so schnelle Rechner benutzt. Statt leistungsfähiger Prozessoren muss also Gehirnschmalz aktiv werden. Zur Motivation für die Lernenden dienen die Zeiten, die das Sortieren durch Zaubern (Quicksort) oder durch Hexen (Radixsort) bietet.

7. Schnelle Sortierverfahren

Warum sind die drei Standardverfahren so schlecht? Der Grund liegt in ihrer prinzipiellen Vorgehensweise. In jedem Durchlauf kümmern sich die Verfahren nur um ein Objekt. Dadurch werden auf jeden Fall n Sortierschritte benötigt. Ein besseres Verfahren muss sich also in jedem Schritt um möglichst viele Objekte kümmern.

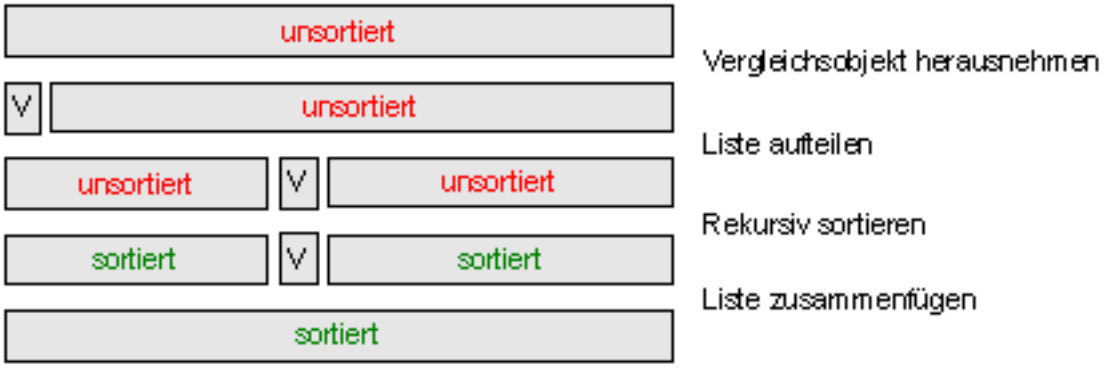
Als schnellere Verfahren findet man in der Modelllösung Quicksort, Mergesort und Radixsort. Hier wird zunächst Quicksort behandelt.

Die Grundidee des schnellen Verfahrens ist recht einfach. Es funktioniert folgendermaßen:

- Nimm irgendeine (möglichst mittelmäßige Karte) aus der Wettkampfliste.
- Lege alle Karten mit besserer Leistung in eine Liste und alle Karten, die schlechter sind, in eine andere.
- „Irgendwer“ soll sich nun um diese beiden Listen kümmern.

- Anschließend wird die Liste wieder zusammengefügt. Zunächst die (mittlerweile sortierten) besseren Karten, dann die Vergleichskarte und dann die (mittlerweile ebenfalls sortierten) schlechteren Karten.

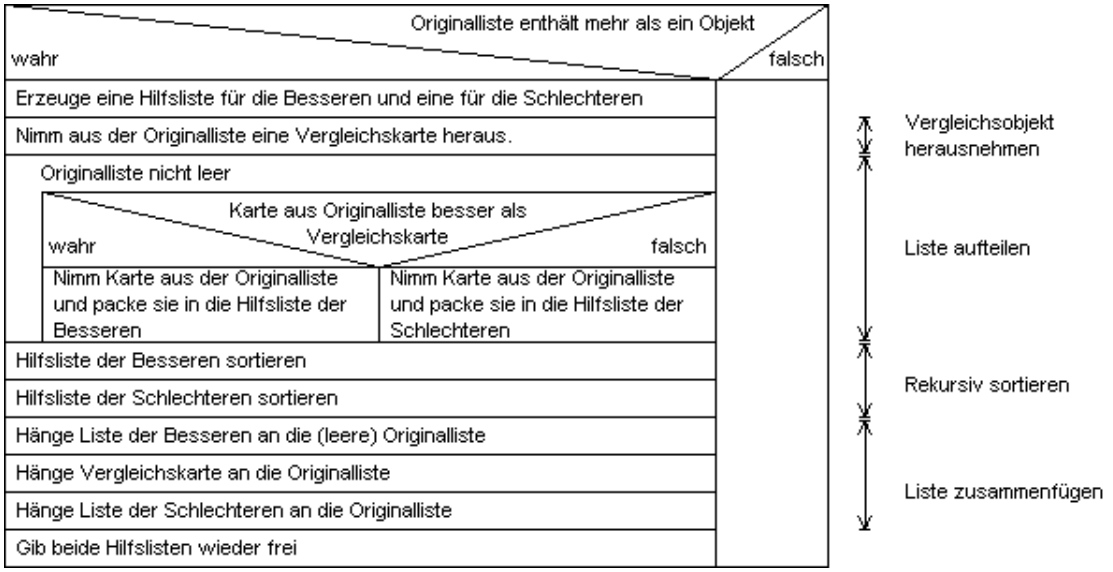
Damit ist die gesamte Liste sortiert. An der Beschreibung wird deutlich, dass dieses Verfahren rekursiv arbeitet, da sich ja „irgendwer“ um die Besseren und Schlechteren kümmern muss.



Die Liste wird also zerlegt und nach Sortierung der Teile wieder zusammengefügt. Das Verfahren ist auch in der Wissenschaft aufgrund seiner Geschwindigkeit als Quicksort bekannt. Algorithmen dieser Form werden auch "divide and conquer" (teile und herrsche) Verfahren genannt.

Man kann das Verfahren nun gut mit großen Blättern mit passenden Zahlen (etwa das Beispiel weiter unten in den theoretischen Überlegungen) an einer Magnettafel durchspielt, wobei man gut die „Irgendwer“-Rollen an die Lernenden verteilen kann.

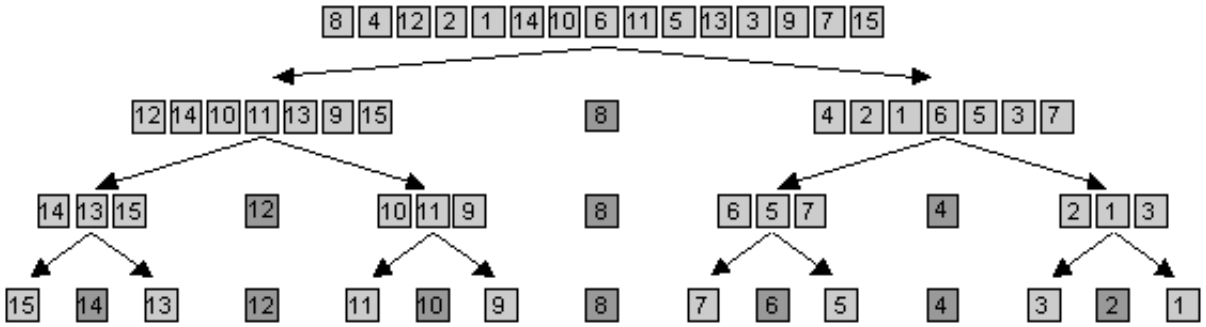
In einem Struktogramm sieht das Verfahren so aus:



Es sollte nun problemlos implementiert werden können.

Auch hier kann man theoretische Überlegungen zum Rechenaufwand dieses Verfahrens anstellen:

- Verhalten im besten Fall (best case)
  Im besten Fall wird die Liste in eine Vergleichskarte und zwei gleich große Listen aufgeteilt. Für 15 Karten sieht die Aufteilung dann so aus.



  In jedem Schritt halbiert sich also in etwa die Anzahl der Karten in einer Wettkampfliste. Daher benötigt man in etwa ld(n) Sortierebenen. Dabei bezeichnet ld den Logarithmus zur Basis 2 (Logarithmus dualis). In jeder Ebene müssen alle Karten in die jeweiligen Listen aufgeteilt werden. Damit hat das Verfahren etwa den Aufwand $O(n \cdot ld(n))$.
  Da der Logarithmus ein sehr schwaches Wachstum hat, ist $n \cdot ld(n)$ deutlich kleiner im Wachstum als n^2. Das Verfahren ist also „um eine Klasse“ besser.

- Verhalten im schlechtesten Fall (worst case)
  Im schlechtesten Fall allerdings erwischt man als Vergleichskarte immer die beste (oder die schlechteste). Dann wird die Liste in eine nur um eins kleinere Liste und eine leere Liste aufgeteilt. Man bekommt also n Ebenen. Das Verfahren hat im schlechtesten Fall also quadratischen Aufwand.

- Verhalten im mittleren Fall (average case)
  Praktische Untersuchungen zeigen, dass das Verfahren immer noch den Aufwand $O(n \cdot ld(n))$ hat. Dies theoretisch zu ermitteln ist ausgesprochen schwierig.

Warum ist Quicksort nun so schnell? Anders als die drei langsamen Verfahren arbeitet Quicksort innerhalb einer Stufe mit allen Objekten. Diese werden dabei über große Distanzen bewegt. Dadurch kommt das Verfahren mit deutlich weniger Stufen aus.

In zwei weiteren schnellen Verfahren ist der Beweis des Aufwandes viel leichter und dabei ist auch keine Fallunterscheidung nötig. Diese Verfahren, Mergesort mit demselben Aufwand wie Quicksort $O(n \cdot ld(n))$ und Radixsort mit sogar nur linearem Aufwand $O(n)$, eignen sich daher sehr gut für Klausuraufgaben im Anschluss an diese Reihe.