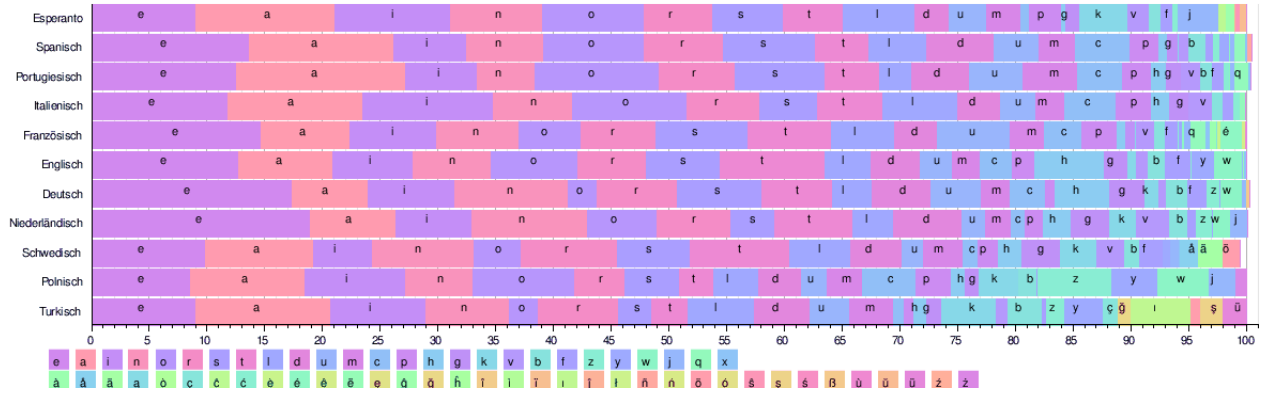
Entwicklungsprozess am Beispiel der „CharCounter“-Software

Ziel ist eine Software, die durch Einlesen eines längeren Textes automatisch erkennt,
um welche Sprache es sich bei diesem Text höchstwahrscheinlich handelt.
Dazu werden die Häufigkeiten eines jeden Zeichens in einem Text gezählt.
Durch das Umrechnen in relative Häufigkeiten (bzw. prozentuale Häufigkeiten) liegt das Profil der Häufigkeitsverteilung vor. Dieses Profil wird mit den Referenzprofilen aus verschiedenen Sprachen verglichen.



Buchstabe	Deutsch	Englisch	Französisch	Spanisch	Esperanto	Italienisch	Schwedisch
a	6,51%	8,167%	7,636%	12,53%	12,12%	11,74%	9,3%
b	1,89%	1,492%	0,901%	1,42%	0,98%	0,92%	1,3%
c	3,06%	2,782%	3,260%	4,68%	0,78%	4,5%	1,3%
d	5,08%	4,253%	3,669%	5,86%	3,04%	3,73%	4,5%
e	17,40%	12,702%	14,715%	13,68%	8,99%	11,79%	9,9%
⋮							

Quelle: *http://de.wikipedia.org/wiki/Buchstabenh%C3%A4ufigkeit*

Sie können lernen:

- Den Umgang mit Strings (Zeichenketten).
- Verwenden von fremden Klassen
- Ein- und Ausgabeoperationen
- Anlegen und Ansprechen von Arrays (Feldern)
- Evtl. Verwenden von Map-Datenstrukturen (TreeMap, HashMap)
- Verfahren zum Vergleich (Bewertung von Ähnlichkeiten) von hochdimensionalen Sachverhalten
- Kennenlernen von Softwareentwicklungsstrategien

1. Texteingabe, Textausgabe

Sie können lernen:
Einbinden und Verwenden von fremden Klassen(-Dateien); Kapseln der Dateioperationen
Ein- und Ausgabeoperationen (Tastatureingaben; Bildschirmausgaben, formatiert Ausgabe)

Auftrag für Einsteiger:
Binden Sie die Klasse IO bzw. die Datei IO.java in Ihr Projekt ein.
Bitte denken Sie daran, vor der Texteingabe eine Eingabeaufforderung zu platzieren.
Geben Sie den eingegebenen String wieder auf den Bildschirm aus.

Auftrag für Fortgeschrittene und Experten:
Verwenden Sie nach Möglichkeit zur Ausgabe **System.out.format (….)**
und nicht **System.out.println (...)**

2. Zugriff auf einzelne Zeichen eines Strings

Sie können lernen:
Methoden der JAVA-Klasse String: **length()**, **charAt()**
Indexmanipulation
evtl. foreach - Schleifenkonstruktion

Auftrag für Einsteiger:
Geben Sie die Zeichen einzeln aus. Dazu ist eine Schleifenkonstruktion notwendig.
Wählen Sie aus den zur Verfügung stehenden Schleifenkonstruktionen
for- und while- Schleife (kopfgesteuert) bzw. do-while (fußgesteuert) ein passendes Konstrukt aus.

a. Ausgabe der Zeichen des Strings in jeweils einer neuen Zeile aus.
b. Ausgabe der einzelnen Zeichen in umgekehrter Reihenfolge

Auftrag für Fortgeschrittene und Experten:
JAVA erlaubt eine foreach - Schleifenkonstruktion. Hierbei handelt es sich um eine abkürzende Schreibweise,bei dem einfach alle Elemente eines Feldes (Array, String, Liste usw.) angesprochen werden. Dazu muss allerdings die Daten in der Form eines Arrays ansprechbar sein.
Formulieren Sie die Aufgabe mit einer foreach-Schleife.

Hinweis:
```
for (char cZeichen : strString.toCharArray( ) )
{….}
```

3. Zählen und Ausgeben der Häufigkeiten eines (festen) Zeichens

Sie können lernen:
Fallabfrage innerhalb einer Schleife
Unterscheidung zwischen Klein- und Großbuchstaben (case-sensitive)

Aufgabe:
Notieren Sie eine Schleife, die alle Zeichen eines Strings genau einmal anspricht.
Das angesprochene Zeichen soll dann mit einem anderen Zeichen, z.B. das kleine 'e' verglichen werden. Bei einer Übereinstimmung soll eine Zählvariable hochgezählt (inkrementiert) werden.
Damit die Klein- und Großbuchstaben gezählt werden muss die Abfrage abgeändert werden.
Dazu gib es mehrere Möglichkeiten:
-Sie können eine zweite Abfrage einbauen, die den Großbuchstaben zählt, z.B. 'E'

-Sie können den String komplett in Klein- bzw. Großbuchstaben umwandeln.
Dazu können Sie die String-Methode **lowerCase()**, **upperCase()** aufrufen.

Aufgabe für Fortgeschrittene:
Das zu zählende Zeichen soll eingebbar sein.
Außerdem soll die Häufigkeit für Groß- und Kleinbuchstaben getrennt gezählt werden.
Ausgeben werden soll also für den Buchstaben e:
Anzahl e: 47 Anzahl E: 11 Gesamtanzahl (e und E): 58
Geben Sie das Ergebnis möglichst formatiert aus.

4. Zählen und Ausgeben der Häufigkeiten aller Zeichen

Sie können lernen:
Verwenden einer größeren Datenstruktur als hochdimensionalen Zählvariable.
Verwendung des ASCII-Codes.

Aufgabe:
Um alle auftretenden Zeichen eines Strings zu zählen sind mehrere Verfahren möglich.
Prinzipiell könnten Sie für jedes Zeichen eine Fallabfrage formulieren. Das ist sehr aufwendig.
Ein anderer Weg ist die Umwandlung in ASCII-Code bzw. die Interpretation des Zeichens als ASCII-Code. Der ASCII-Wert kann dann als Index für ein Integer-Array verwendet werden. Dieses Verfahren ist in der Ausführung äußerst schnell. Nachteilig ist jedoch, dass verschiedene Sonderzeichen (Nicht-ASCII-Zeichen) nicht dargestellt bzw. nicht gezählt werden können.
Eine weitere Möglichkeit ist die Verwendung einer Map. Hierbei handelt es sich um eine Tabelle, bei der (Nutz-) Wert durch einen Schlüssel und nicht durch einen Index angesprochen wird.
In JAVA sind die beiden Maps *HashMap* (schnell) und *TreeMap* (sortiert) vorbereitet.
Die Verwendung dieser Klassen ist etwas aufwendig. Die Ausführung ist nicht ganz so schnell wie bei der vorigen Möglichkeit. Trotzdem handelt es sich hier um das Vorgehen der Wahl.

Entscheiden Sie sich für eine der drei Möglichkeiten und implementieren Sie Funktionalität, alle auftretenden Zeichen zu zählen.

Aufgabe für Experten:
Erstellen Sie eine Klasse, die die Klasse *TreeMap* erweitert.
Hier erstellen Sie eine Methode, die die zu zählenden Zeichen ausfiltern kann.

5. Einlesen der Zeichenketten aus einer Datei

Sie können lernen:
Einlesen von Textdateien
Umrechnung der absoluten in relative Häufigkeiten
Evtl. Effizienzüberlegungen, Effizienzmessungen

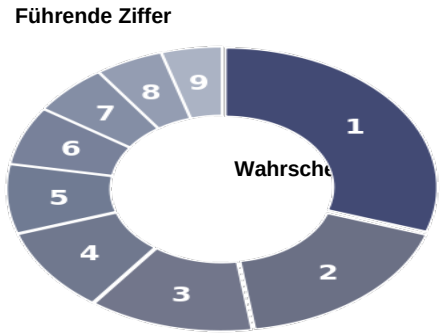
Aufgabe für Einsteiger:
Lesen Sie eine größere Textdatei in den Eingabestring ein. Mittels der IO-Klasse ist das problelos möglich. Laden Sie sich dazu ein größere Datei (mindestens 800kByte) aus dem Internet auf den Rechner. Bei großen Textdateien macht die Ausgabe von absoluten Häufigkeiten wenig Sinn (4711000mal das 'e' oder 'E'). Geeigneter erscheint hier die Ausgabe der relativen oder der prozentualen Häufigkeit (also zum Beispiel 'e': 0,174 bzw. 17,4%).

Aufgabe für Fortgeschrittene und Experten:
Stellen Sie Überlegungen an, wie verändert sich die Ausführungszeit bzw. Laufzeit wenn die zu untersuchende Textdatei doppelt, vierfach oder zehnfach so groß wird.
Stimmen Ihre Überlegungen mit den gemessenen Zeiten ungefähr überein?

6. Analyse von ziffernlastigen Textdateien (Textdateien, die sehr viele Zahlen bzw. Ziffern enthalten)

Sie können lernen: Benford'sche Gesetz
Extrahieren von Ziffern

Das Bendford'sche Gesetz besagt, dass die Häufigkeit der führenden Ziffer nicht gleichverteilt sind. Nach Bendford tritt die 1 im Vergleich zur 9 mehr als sechsmal häufiger als führende Ziffer auf.



1	30,1%
2	17,6%
3	12,5%
4	9,7%
5	7,9%
6	6,7%
7	5,8%
8	5,1%
9	4,6%

Quelle: Tabelle und Grafik,
http://de.wikipedia.org/w/index.php?title=Benfordsches_Gesetz&redirect=no

Mit Hilfe dieses Phänomens wurden Betrugsfälle im Rechnungswesen durch das Management bei Enron und Worldcom nachgewiesen. Die Manipulation von Wirtschaftsdaten in Griechenland ließ sich ebenfalls durch das Benford-Gesetz nachweisen.

Aufgabe für Einsteiger:
Untersuchen Sie die Häufigkeiten von unterschiedlichen ziffernlastigen Textdateien (z.B. 100.000 Stellen von Pi, Messdaten, Bilanzen). Die Eigenschaft, dass es sich um die erste Ziffer handeln muss, sollen/können sie vernachlässigen.

Aufgabe für Fortgeschrittene:
Stellen Sie Überlegungen an, wie bei dem Auszählen der verschiedenen Ziffern nur die jeweils erste Ziffer einer Zahl berücksichtigt wird.

Aufgabe für Experten:
Erstellen Sie eine Sonderversion, die beim Auszählen der Ziffern jeweils nur die erste Ziffer einer Zahl berücksichtigt. Überprüfen Sie an Hand einer zahlenlastigen Textdatei das Benford-Gesetz

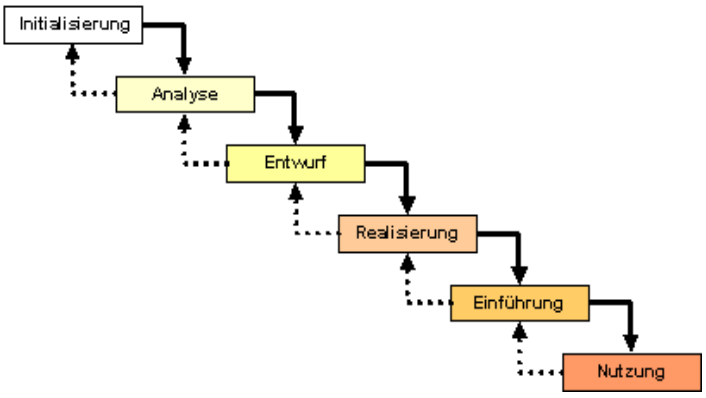
7. Softwareentwicklungsprozesse

Sie können lernen:
Kurzvorstellung von verbreiteten (etablierten) Entwicklungskonzepten
Konzeptionelle Überlegungen zu Testroutinen

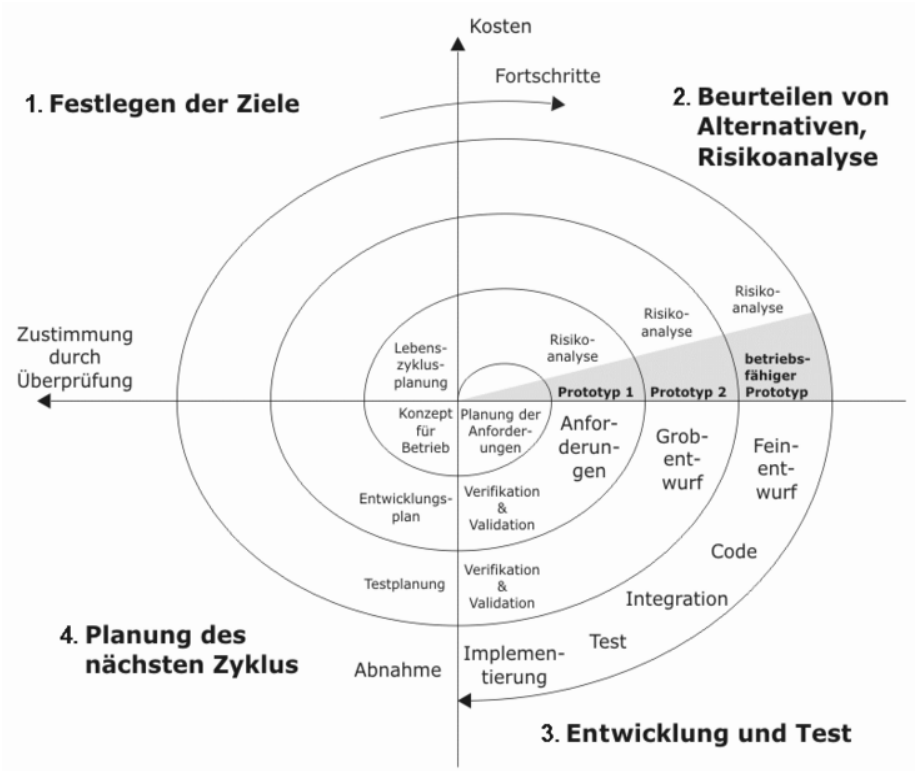
Bei jeder Software existiert das Problem der Beweisbarkeit der Korrektheit des Programms. Nachweisen lässt sich nur die Anwesenheit von Fehlern. Nicht die Abwesenheit von Fehlern! Die Aussage: „Das Programm ist korrekt“ gilt nur solange bis das Gegenteil bewiesen ist. Deshalb kommt den unterschiedlichen Verfahren zur effizienten Softwareentwicklung eine besondere Bedeutung zu. Vergleichen Sie dazu:
http://de.wikipedia.org/wiki/Liste_von_Softwareentwicklungsprozessen

Einige wichtige, etablierte Verfahren sind:

a. Das Wasserfallmodell

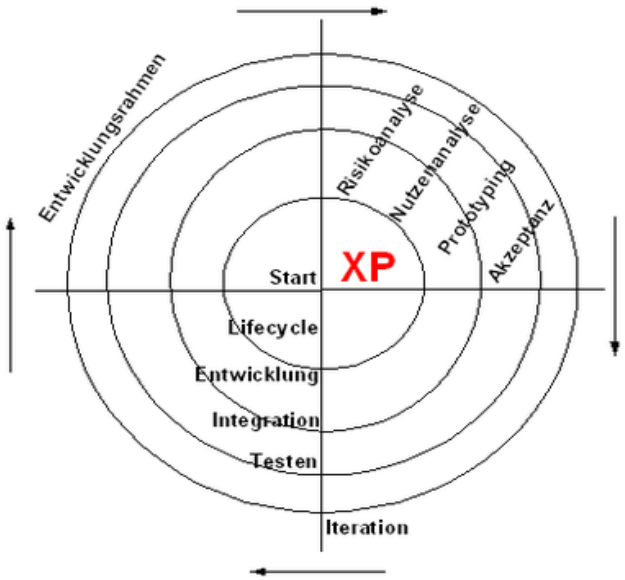


b. Das Spiralmodell



c. eXtreme Programming (XP)

Ein wichtiger Aspekt in der Vorgehensweise der XP ist die testgetriebenen Entwicklung. Das bedeutet, dass der Programmierer vor der Erstellung einer Routine zuerst die Testroutine schreibt. Das erzwingt zunächst eine genaue Überlegung bezüglich Aufgabe (Spezifikation) der neuen Routine. Der eigentliche Softwaretest läuft dann automatisch ab.



Aufgabe:
Stellen Sie Überlegungen an, wie sich Ihre erstellte Software (automatisch) testen lässt.
Formulieren Sie Bedingungen die eine zu erstellende Testroutine erfüllen muss.

Das Ziel des zweiten Entwicklungsschrittes ist es, einem analysierten (sinnvollen, längeren) Text die entsprechende Sprache zuzuordnen. 'Analysiert' soll hier bedeuten, dass die unterschiedlichen Zeichen des Textes ausgezählt wurden und dieses Ergebnis vorliegt.
Für den folgenden Entwicklungsschritt benötigen Sie also einen korrekt funktionierenden CharCounter, der Textdateien einlesen kann und die Anzahl der Zeichen ausgeben kann. (Aufgabe 5 bzw. 6).
Die Software zur Sprachzuordnung soll in mehreren Teilschritten entwickelt werden. Die Zuordnung soll mit Hilfe der Tabelle zur Buchstabenhäufigkeit für verschiedene Sprachen erfolgen.
Vergleichen Sie dazu

http://de.wikipedia.org/wiki/Buchstabenh%C3%A4ufigkeit

Ziel ist es, die Sprache auszuwählen, zu der das Auszählergebnis am Besten passt.

8. Aufbereiten der Analysedaten

Aufgabe für Einsteiger:
Das Auszählen der Zeichen einer Textdatei unterscheidet zwischen Groß- und Kleinbuchstaben.
Es bietet sich an, vor dem Auszählvorgang den eingelesen Text entweder in Groß- oder in Kleinbuchstaben zu wandeln.

Zusätzliche Aufgabe für Fortgeschrittene:
In den Tabellen der Buchstabenhäufigkeiten werden nur Angaben zu den Buchstaben gemacht (eigentlich klar!). Die CharCounter – Software zählt jedoch alle Zeichen, eben auch Leerzeichen, Satzzeichen (*Punkt, Komma, Semikolon, Doppelpunkt, Ausrufezeichen, Fragezeichen, Anführungsstriche, Bindestriche usw*.)
Um das Auszählergebnis mit den Tabelle vergleichen zu können, müssen alle Zeichen die keine Buchstaben sind, ausgefiltert werden.
Überlegen Sie sich eine Vorgehensweise, wie dieses Ausfiltern am Besten zu realisieren ist.

Zusätzliche Aufgabe für Experten:
Die Tabelle der Buchstabenhäufigkeiten macht Prozentangaben (relative Häufigkeit). Die Ergebnisse des CharCounter enthalten jedoch die Anzahl des jeweiligen Zeichens (absolute Häufigkeit). Legen Sie eine Datenstruktur an, die die jeweils ermittelte Anzahl in Prozentwerte umrechnet.

9. Vergleichen der Häufigkeitsverteilung

Die Daten der Buchstabenhäufigkeit aus den Wikipediatabellen müssen dem Softwaretool (irgendwie) zugänglich gemacht werden. Dazu gibt es verschiedene Vorgehensmöglichkeiten mit unterschiedlichem Schwierigkeitsgrad:

> Aufgabe für Einsteiger:
> Legen Sie für jede Sprache ein Array an. Diese Arrays sollen die Prozentwerte aufnehmen können. Die Zuweisung kann im Quelltext fest codiert werden.
>
> Aufgabe für Fortgeschrittene:
> Legen Sie für jede Sprache ein Array oder eine Hashmap an. Diese Datenstruktur soll dann durch das Einlesen von Dateien mit den Prozentwerten befüllt werden.
>
> Aufgabe für Experten:
> Für jede Sprache sollen Dateien mit den entsprechenden Prozentwerten vorliegen. Halten Sie Ihr Programm so flexibel, dass je nach vorhandenen Sprachprofilen (je nach vorhandenen Dateien, die die Prozentwerte enthalten) Ihre Software die entsprechenden Tests durchführt. Ihr Programm kann also automatisch darauf reagieren, wenn eine neues Sprachprofil (Prozentwerte für neue Sprache) zur Verfügung steht.

10. Vergleichen der Häufigkeitsverteilung

Sie können lernen:
Metrik zur Distanzberechnung von hochdimensionalen Daten.
Mit anderen Worten:
Es werden Verfahren vorgestellt, die eine Entscheidung ermöglichen, zu welchem Sprachprofil (Prozentuales Vorkommen der Zeichen einer Sprache) die ausgezählten Zeichen einer Textdatei (Testprofil) am ähnlichsten sind.

Wenn das Testprofil (prozentuales Vorkommen der Zeichen im zu untersuchenden Text) zu dem Sprachprofil (prozentuales Vorkommen der Zeichen einer Sprache) sehr unähnlich ist, dann spricht man von einer großen Distanz (großer Abstand) zwischen beiden Profilen.
Wenn Testprofil und Sprachprofil genau gleich sind, dann besitzen beide Profile den Abstand 0.

Um den Abstand zwischen zwei Profilen, dem Profil einer Sprache und dem Testprofil, zu ermitteln, gibt es mehrere Möglichkeiten:

T_i: Relative Häufigkeiten für den Buchstaben i des Testprofils
R_i: Relative Häufigkeiten für den Buchstaben i des Sprachprofils (Referenzprofils)

a. Betragsmetrik (auch: Manhattan-Metrik oder Mannheimer-Metrik)

Der Abstand wird berechnet, in dem man die positiven Differenzen des jeweiligen Einträge aufaddiert.

Formel zur Berechnung:

Abstand $d := |T_a - R_a| + |T_b - R_b| + |T_c - R_c| + ... + |T_z - R_z|$

b. Euklid'sche Metrik

Der Abstand wird berechnet, in dem man die Quadrate der Differenzen des jeweiligen Einträge aufaddiert.

Formel zur Berechnung:

Abstand $d := [(T_a - R_a)^2 + (T_b - R_b)^2 + (T_c - R_c)^2 + ... + (T_z - R_z)^2]^{0,5}$

Aufgabe für Einsteiger:
Ihre Software soll den Abstand zwischen einem Testprofil und dem Profil einer Sprache erstellen und ausgeben. Dazu genügt es, eine der beiden vorgestellten Metriken auszuwählen.

Aufgabe für Fortgeschrittene:
Berechnen Sie die Abstände des Testprofils zu den Ihnen vorliegenden Sprachprofilen. Finden Sie heraus zu welchem Sprachprofil das Testprofil den kleinsten Abstand besitzt (größte Ähnlichkeit!).

Aufgabe für Experten:
Zu welchem Sprachprofil besitzt das Testprofil die größte Ähnlichkeit (kleinste Distanz)?
Führen Sie Ihre Untersuchungen mit der zweiten Metrik durch. Erhalten Sie das gleiche Ergebnis?