

Objektorientierte Programmierung

Klassen, Objekte...

1. Methode entspricht Funktion in C, aber auf Objekt bezogen
 - a. Methodenkopf – Methodenkörper
 - b. Modifizier *Modifizierer*
 - c. Datentypen
 - d. Parameter *int, Double, String, Byte, Boolean*
 - e. Rückgabe *return*
 - f. Aufruf *x = funktion();*
2. Attribut (Eigenschaft/Variable)
 - a. Deklarieren – Initialisieren *→ automatisch 0/0.0*
 - b. Modifizier *→ #, ~, +, -*
 - c. Datentypen
 - d. Klassenvariable – Objektvariable – lokale Variable *{}_n*
3. Klasse ist Bauplan für Objekt
 - a. Instanziierung ist Erstellen von Objekten von Klassen durch Aufrufen des Konstruktors
 - b. Operator new
 - c. Defaultkonstruktor
 - d. Objekt wird durch Attribute und Methoden definiert
 - e. „.-Operator für Zugriff von „außen“
4. realitätsbezogen Programmieren: Dinge, Personen, Tiere ähnlich als Objekte abbildbar, Softwarekrise in 90ern aufgrund von fehlerhafter Programmierung
5. Merkmale der Objektorientierung
 - a. Abstraktion z.B. Socket, komplexe Vorgänge für Entwickler einfach realisierbar
 - b. Datenkapselung einschließen von Daten durch private
 - c. Polymorphie Eine Referenz eines Typs kann aufgrund von Vererbungen auch alle abgeleiteten Formen enthalten: List x = new LinkedList();

private -
public +
protected #
package ~
scope

static
final
abstract
synchronized
oder Wrapper

static

komplizierte
Internet-Verbind
ist socket

für den
Programmieren
den Socket
verwendet
einfach

Vererbung ermöglicht es,
dass eine Referenz auch alle
abgeleiteten Formen eines anderen
Typs enthalten können

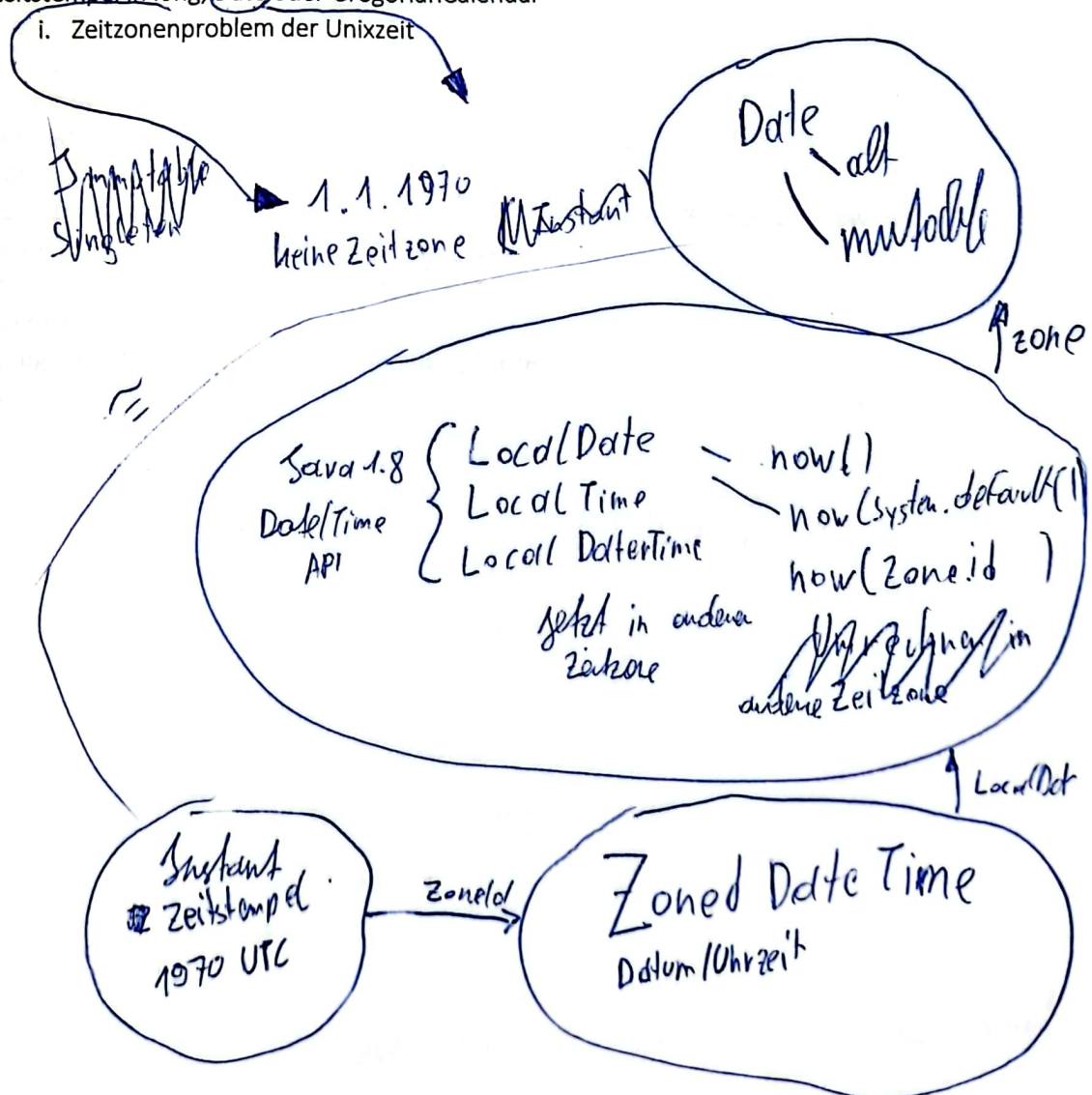
für Programmstellung bewährt hat

Designpattern

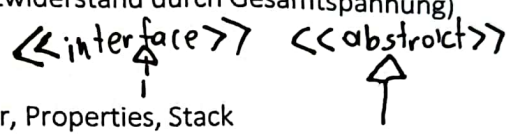
1. Standardlösungen zur allgemeinen Verständlichkeit von Quellcode
2. immutable: Alle Attribute final, Veränderungen nur mit getter und setter
a. Java mit Objektreferenzen (Zeigern); wenn Objektreferenzen übergeben werden, könnte anderer in der Zwischenzeit Elemente ändern;
b. String ist immutable, schlecht ausgeführt bei Collections; diese können in einer Klasse final gesetzt werden, der Inhalt aber nicht; wenn jemand anders auf Collection zugreifen kann, kann der Daten ändern
3. Singleton
a. Immer maximal eine Instanz von einem Objekt verfügbar
4. Rechnerklasse: Parameter bzw. Daten werden übergeben, dann werden Ergebnisse zur Verfügung gestellt
5. Es soll eine Klasse zur Verwaltung von Messwerten nach dem Entwurfsmuster immutable erfolgen.
a. Alle Attribute final setzen, über den Konstruktor initialisieren und getter-Methoden zur Verfügung stellen
b. Zeitstempel in long, Date oder GregorianCalendar
i. Zeitzoneproblem der Unixzeit

Datenhaltungsklasse

→ Datenbankklasse



Interface

1. nur Methodenköpfe
2. mehrere Interfaces können implementiert werden
3. Ausnahme: unterschiedliche Interfaces enthalten Methode mit selbem Namen und selber Parameterliste
4. Beispiel: Liste von Bauteilen
 - a. Widerstand und Kondensator implementieren Bauteil
 - b. Bauteil: Interface getResistor(), getActiveVoltage()
 - c. Gesamtstrom berechnen (alle Widerstandswerte und alle Spannungswerte der Liste addieren, dann Gesamtwiderstand durch Gesamtspannung)
5. UML-Diagramm
6. Interface Collection
 - a. Vorgänger Dictionary, Vector, Properties, Stack
 - b. Performance collections
 - c. verschiedene ~~Listen~~ sollen sich ähnlich Verhalten
 - d. einfaches Erweitern und Bearbeiten
 - e. AbstractList, LinkedList, ArrayList
 - f. Methoden der einzelnen Collections
 - g. Iterator
7. Wrapper-Klassen
 - a. null verfügbar
 - b. Generische Klassen
 - c. Collections arbeiten intern noch immer mit Wrapper-Klassen z.B.: .add(5): int wird intern in Integer umgewandelt)
8. lokalisierte vs nicht lokalisierte Ausgabe
 - a. Scanner vs parse
 - b. String.format vs toString; internationalisation i18n

↳ (LOCALE.DE

Fehlerbehandlung in Java

1. Exception
 - a. trycatch, multicatch, weiterwerfen
2. Ableitungsbaum: Object – Throwable – Exception und Error unchecked
 - a. weitere Ableitungen von Exception u.a. RuntimeException mit weiteren Ableitungen
 - b. Errors nicht fangen, da Programm dann schon am Arsch ist
 - c. unchecked vs checked Exceptions:
 - i. Error und RuntimeExceptions unchecked, alle anderen Ableitungen Throwables sind checked Exceptions
 - ii. Wenn RuntimeExceptions weitergeworfen werden führen diese in der obersten Schicht zum Programmende mit Ausgabe der Exception
 - iii. checked bedeutet, dass diese Exceptions bereits beim Compilieren auftreten
 - d. Fehlerbehandlung in C:
 - i. Rückgabewert einer Funktion: return 0 oder Fehlercode
 - ii. Zeiger auf NULL bei z.B. file.open kein File gefunden heißt NULL
 - iii. Globale Variablen, errno bei Files
 - iv. unsystematisch, jeder macht anders, kein Konzept; C++ als erstes mit Exceptions (nur unchecked Exceptions)