Grundlagen der Informatik für Ingenieure I

19.1 Rechnerarchitektur

- Rechnerarchitektur
  - Funktionseinheiten der “von Neumann-Architektur” - bg 3 - Teil1.4ff -
    - *Rechenwerk*
    - *Leitwerk*
    - *Speicher*
    - *E/A-Werk*
  - Wichtige CPU-Register und ihre Funktion - bg 3.16ff ; Tafel -
  - Welche “Attribute” der Rechnerarchitektur spielen eine Rolle im Zusammenhang mit - bg 3 - Teil1.23ff -
    - *Wertebereichen* - Registerbreiten -
    - *Genauigkeit* - Registerbreiten -
    - *Adressierbarkeit* - Registerbreiten; Auswirkung auf ASP, HintergrundSP, Plattensystem, Dateilängen
    - *Typzugehörigkeit primitiver Datentypen* - Functional Units -
  - Leistungsmaße für Prozessoren - bg 3 - Teil 2 .7ff -
  - Speicherhierachien
    - *“Geschwindigkeitsproblem”* - bg 3 - Teil 2 .9ff -
    - *“Adressraumproblem”* - bg 3 - Teil 2 .9ff -

19.2 Betriebssysteme

- Betriebssystemklassen
  - Batchsysteme, Realtime Systeme, Time Sharing Systeme
- Grobe Gliederung
  - Systemschnittstelle (API), System-Call-Handler - SVC -
  - Dateisystem -open(); close(); read(); write(); cntrl(); seek(); skip(); sync();
  - Prozessverwaltungssystem -fork(); exec(); wait(); exit(); -
  - Speicherverwaltungssystem - malloc(); free() -
  - E/A-System -open(); close(); read(); write(); cntrl(); -
- Ereignisse
  - asynchrone - zum lfd. Prozess -
  - synchrone - zum lfd. Prozess -
  - Interrupt - extern; EA - System; lfd. Prozess wird unterbrochen, falls Interrupt nicht gesperrt -
  - Trap; - ausgelöst durch den Prozess selbst, z. B. SVC -

19.2.1 Dateisysteme

- Darstellung
  - Kataloge
  - Dateien
  - Links
  - Zugriffsrechte
  - Dateihierachie
- Organisation (Datenstrukturen) und dynamischer Ablauf - bg 4.13 -
  - E/A-Systemcalls
  - Dateikontrollstrukturen (inodes, filetables)
  - Puffersysteme
- Plattenorganisation - nicht Teil der Prüfung -

19.2.2 Prozessverwaltungssystem

- Prozesse, Threads - bg 5 -
  - Schutzumgebungen, Prozess - bg 5.8ff -
    - Prozess = n x Aktivitätsträger (threads) + Schutzumgebung, häufiger Sonderfall n =1; dann spricht man allg. nur von einem Prozess - auch als Aktivitätsträger !
  - Zustände, Scheduler - bg 5.10ff -
  - Interupts; Traps; SVC - wie F3 -
  - Erzeugung/Ausführung eines Prozesses unter Unix (Prinzipskizze) - bg 5.16ff; Tafel
  - Prozesshierachie - bg 5.18, Tafel -
  - Wechselwirkungen
    - *Speicherverwaltung* -siehe dort -
    - *Dateiverwaltung/ E-A-Verwaltung* -siehe dort -
  - Zuteilungsstrategien - bg 5.26ff -

19.2.3 Speicherverwaltungssystem

- Strukturierung von Speichern
  - physikalischer Adressraum/ logischer Adressraum - Hardware/programm- bzw. prozessspezifisch; was sind symbolische Adressen? -
  - Relocation - bg 6.11ff; Verschiebbarkeit; log. Adressraum -
  - Bit, Bytes, Worte (prozessorspezifisch), Segmente (var. Länge), Kacheln (feste Länge)
  - Codesegment/Datensegment/Stacksegment -Tafel -
- Verfahren der Speicherverwaltung/Hardwareunterstützung
  - Vergabestrategien - bg 6.6 -
  - Segmentierung/Paging - bg 6.13ff -
  - Virtueller Speicher - bg 6.21ff -
  - Ein/Auslagerungsstrategien - bg 6.27ff -

19.2.4 Rechnernetze

- Netzwerk-Protokolle
  - was ist grundsätzlich ein Netzwerkprotokoll?
    - eine Strukturierung von Informationen, die sowohl dem Sender als auch dem Empfänger bekannt ist - Beispiel: Analogtelefonie; Protokoll?
  - sichere und unsichere Datenübertragung; Flusskontrolle
  - Client, Server - bg 7.11ff -
  - Adressierung und Routing auf IP-Ebene; Routingtabellen, DNS, ARP - bg 7.18ff -
  - Dienste, Services, Ports - bg 7.28ff -
  - NORMA-Architektur - nicht Teil der Prüfung, wie auch Kap 16 -

19.3 OO Programmentwicklung

19.3.1 Klassen - 5.ff -

- Was ist eine Klasse? -template; Bauplan; statisch; Definitionen -
- Klassenhierachien - in Java baumförmig wg, single Inheritance -
- Vererbung *(Inheritance)*
  - *Single ...* - Java -
  - *Multiple ...* - z. B. C++; Java-Interfaces?!? -
  - *(Wie und) was erbt man implizit?* - Object-class -
  - *Wie erbt man explizit?* - extends; 5.70ff -
- Klassenvariable - static - Gültigkeitsbereich? -
- Klassenmethoden - ohne Init. "nutzbar"!
  - *Modifier* - static -
  - *Zugriff auf Variable* - nur auf Klassenvariable -

19.3.1 Klassen - 5.ff -

- Klassendefinition
  - *extends* - Vererbung, ..erbt von ..; erweitert Oberklasse... -
  - *implements* - Aufzählung der Interfaces, die in "dieser" Klasse definiert sind -
  - *Modifier*
    - *public* - auch von ausserhalb des packages aufrufbar -
    - *abstract* - abstrakte Klasse; Definitionen; keine Instantierung wohl aber Bildung von Unterklassen -
    - *final* - keine Erweiterung möglich; kein extends... -
  - *Konstruktoren*
  - *default Constructor* - Aufruf ohne Parameter; default Vorbesetzung - Tafel -
  - *default finalizer* - implizit; Referenzen werden auf "null" gesetzt; Garbage Collector -

19.3.2 Objekte/Instanzen - 5.ff -

- Was ist ein Objekt, eine Instanz?
  - "dynamisches" Objekt erzeugt aus dem "Bauplan" der Klasse -
- Kreieren eines Objekts, Instantiieren
  -new; implizit: string literale, arrays mit Initialisierung -
  - *Template --> Objekt*
  - *Referenz auf das Objekt* - Objektvariable; == null heisst keine Referenz -
  - *Constructor* - init. Objekt -
  - *Variable* - Instanz -(Objekt-) Variable -
  - *Code vs. Daten* - Code sharable; Daten lokal -
  - *Speichermanagement*
    - automatisch: bei "new" --> malloc(); Garbadge Collector --> free() -
- Objektzustand - "Summe"; "Hülle" aller Instanzvariablen-Inhalte -
- Instanzvariable
  - *Gültigkeitsbereich* - default: innerhalb package -
  - *Modifier* - public, private, protected -
  - *Access Controll* - need to know prinziple; Kappselung -

19.3.3 Methoden und Konstruktoren - 5.ff -

- Methoden
  - Definition - [modifier] returntype methodName([Parameterliste]) Rumpf -
    - *Signatur* - methodName(Parameterliste) -
    - *Returntyp* - void, prim Typ, Objekte -
    - *Access Control* - public, private, protected, (default)-
  - Parameterübergabe
    - *Call by Reference* - Änderungen auf das Original; Objekte (in Java) -
    - *Call by Value* - Änderung auf eine Kopie; prim. Typen (in Java) -
  - *Modifier*
    - *abstract* - nur Signatur; keine Implementierung - wofür dann gut? -
    - *static* - Klassenmethode -
    - *(native)* - nicht behandelt -
    - *final* - kein Overriding -
    - *synchronized*
      - im Code des Rumpfes der Methode befindet sich immer nur ein Aktivitätsträger -

19.3.3 Methoden und Konstruktoren - 5.ff -

  - *Overriding* - Signatur gleich; neue Implementierung der (geerbten) Methode -
  - *Overloading* - alternative Signaturen -
- Konstruktoren - Init. von Objekten -
  - *Overriding* - nein! -
  - *Overloading* - ja, Objekte können beliebig viele Konstruktoren besitzen, min. einen "default" ohne Parameter -
  - *super()* - automatisches int. einer Super- (Ober-) klasse -
  - *super(p1, p2,...)* - explizites init. einer Super- (Ober-) klasse; 1. Statement im Konstruktor der Subklasse!! -
- Referenzen
  - *this* - "dieses" Objekt -
  - *this()*
    - Aufruf des Konstruktors dieses Objekts = Instantiierung eines weiteren Objekts vom gleichen Typ -
  - *this.name* - Instanzvar "dieses" Objekts -
  - *return[value]* - Rücksprung in die aufrufende Methode -
  - *super.methodename()*
    - Aufruf einer Methode der Superklasse; z. B. dann, wenn man sie in der Unterklasse überschrieben hat -

19.3.3 Methoden und Konstruktoren - 5.ff -

- Ausgezeichnete Methoden
  - *main()* - Applikationen -
  - *init()* - Applets -
  - *run()* -Threads -
- Referenzen auf Objekte
- Zugehörigkeit eines Objekts zu einer Klasse - instanceof -
- Relationale Operationen auf Objekte - "==", "!="; Identität wird festgestellt!!! -

19.3.4 Variable und Konstanten - 4ff -

- Namenskonventionen
- Variable, Konstante - 4.5ff -
  - *Definition* - [modifier] typ varName [= Wert]; -
  - *Datentypen* - 8 prim. ; Objekte; strings; arrays -
  - *Namenskonstante, Literale (Konstante)*
  - *Modifier*
  - *Initialisierung* - Namenskonstante muss; Varaiable kann -
  - *Gültigkeitsbereiche* - abhängig vom Ort der Definition und von Modifiern -
  - *Wertebereiche (Wortbreite)*
  - *Genauigkeit (Wortbreite)*
  - *Lebensdauer*
    - lokale Variable bis Verlassen der "Umgebung {}"; Objekte bis sie gleich "null" gesetzt werden (Garbage Collection); sonst: solange Prozess existiert -
  - *Casting* - 5.43ff; automatisch, wenn keine Info verloren geht, sonst nur explizit -
  - *Converting* - 5.43ff; Umwandlung prim. Typ <--> Objekt des entsprechenden Typs

19.3.5 Datentypen, Operatoren - 4ff -

- Primitive Datentypen
  - *Typgruppen* - ganzzahlig, reell, logisch, Zeichen -
  - *Typen* - byte, short, int, long, float, double, boolean, char -
  - *Wertebereiche*
  - *Operatoren*
    - arithmetische
    - bool'sche
    - relationale
    - Vergleichsoperator als spez. Statement
    - *Casting* - 5.43ff -
    - *Converting* - 5.43ff; ---> "new"; <--- TypklassennameValue
    - Objekte prim. Datentypen
    - *arrays* - new; init. -
    - *strings* - new; literal -

19.3.6 Java-Sprachkonstrukte - 4ff -

- Quellcodelayout
- Strukturierung
- Statement
- Block of Statements
- Flusskontrolle
  - *if ..*
  - *if...then...*
  - *if...then...else*
  - *while...*
  - *do ... while*
  - *for....*
  - *break*
  - *continue*
  - *switch*
  - *label*

19.3.6 Java-Sprachkonstrukte

- Regeln
  - *Precedenceregeln*
  - *Klammerungen*
  - *Aussprünge aus Schleifenkonstrukten*
  - *Schreibweisen von*
    - Klassennamen
    - Variablennamen
    - Methodennamen
    - Konstantennamen
    - Literalen

19.3.7 Modifier

- *default, public, private, final, static, protected, abstract, synchronized, (volatile), (native)* angewandt auf: - Aufgabe 10 -
  - *Variable*
  - *Instanzvarialble*
  - *Klassenvariable*
  - *Methoden*
  - *Klassen*
- Schutz-(Protection) Konzept
- Kapselung
- *“Need to Know” - principle*
- *Access Methods*

19.3.8 Applets und Applikationen - 6ff -

- Applets
  - *Eigenschaften* - nur im Browserumfeld, Appletviewer ablauffähig -
  - *Voraussetzungen* - java enabled Browser -
  - *Anwendungsumfeld* - Internet - Anwendungen -
  - *Restriktionen (Security)*
    - keine E/A; keine Third-Party-Komm. (nur mit Server); kein fork() - keine Prozessstart; Kein Laden von Libs; "sandboxing" -
- Applikationen - analog zu einem lokalen C-Programm -
  - *Eigenschaften*
  - *Voraussetzungen*
  - *Anwendungsumfeld*

19.3.9 Exceptionhandling

- Worum geht es beim *Exceptionhandling?* - 12ff, 13ff -
- Gruppierung von *Exceptions*
  - *synchron, asynchron*
  - *Error, Exceptions*
  - *Runtime-Exceptions*
  - *implizit, explizit*
  - *I/O-Exceptions*
- Sprachkonstrukte und ihre Bedeutung
  - *try {...}* - Codeteil, der eine Exception auslösen könnte -
  - *catch {...}* - Codeteil, der nach einer Exception diese "behandelt" -
  - *throw* - Auslösen einer Exception (nicht behandelt!) -
  - *...throws* - ...behandelt Exceptions -
  - *(“rethrowing”)* - nicht behandelt -
  - *finally* - nicht behandelt -

19.3.10 Interfaces - 9ff -

- Was ist ein Interface?(vs. abstracte Klassen)
  - *Hierachiebeziehungen*
  - *nur abstrakte Methoden*
  - *Namenskonstante sind möglich*
  - *implements*
    - vollständige Implementierung
  - *mehrfache Interfaces*
  - *Interfacedefinition*
  - *Beispiele*
    - *Runnable (threads)*
    - *xxxxListener*
    - *LayoutManager*
    - *DataInput*
    - *DataOutput*

19.3.11 Java-E/A-System -14ff -

- Streamskonzept
  - nicht durch das BS interpretierte Datenströme; Datenquellen, Datensenken; sämtliche E/A-Geräte sind konzeptionell Dateien! -
  - *Inputstreams*
  - *typische Methoden* - read, skip, available, close; kein open! -
    - Reader - char -
    - InputStream - byte -
  - *Outputstreams*
  - *typische Methoden* - write, flush, close, kein open! -
    - Writer - char -
    - OutputStream - char -
  - *Ein/Ausgabe auf Dateien* - File Class -
  - *Ein-/Ausgabe auf Sockets* - Socket Class -
- Standard-E/A - System.out; System.in; Kanäle sind autum. geöffnet -
- Formatierte E/A
  - Interfaces: DataInput, DataOutput ... maschineninterne Darstellung, nicht mit Editor lesbar; nicht portierbar -
  - Random Access Dateien - Simulation auf Dateisystem -

19.3.12 Threads

- Was sind *Threads*?
- Threadkontrolle
  - *run()*
  - *start()*
  - *yielt()*
  - *sleep()*
  - *...*
- Threadanwendungen

19.3.13 Abstract Window Toolkit

- Was ist das AWT?
  - *Basiskonstrukte*
  - *Layoutkonstrukte*
  - *Windows*
- ...in Applets
- ...in Applikationen
- *Eventhandling*
  - *Listener*
  - *Events*

19.4 Datenstrukturen -14ff -

- Datenstrukturen
  - Pointer (Zeiger) in C und Java - nicht relevant für Prüfung -
  - Felder(arrays)
  - Verkettete Listen - nicht relevant für Prüfung -
  - Warteschlangen, Kellerspeicher (wo einschlägig?)
    - WS - Betriebssystem, KS - Realisierung v. Methodenaufrufen, Gültigkeitsbereiche -
  - Suchen und Sortieren - nicht "direkt" relevant für Prüfung -
  - Gestreute Speicherung (Hashing) - Aufgabe 14 -

19.5 Parallele und Verteilte Systeme - nicht relevant für Prüfung -

- Kassifikation Paralleler und Verteilter Systeme
- Synchronisation und Koordinierung
- Multithreading
- Sockets

19.6 Allgemeines

- Was versteht man unter dem Begriff *JavaVM*?
- Was versteht man unter einer interpretativen Abarbeitung?
  - *Welche Vorteile*
  - *Welche Nachteile*
- Was ist eine Klassenbibliothek?
- Was ist ein Package?
- Was verspricht man sich vom Einsatz des OO-Programmiermodells?
- Wie erreicht man die Kompatibilität von Java-Programm-Systemen?
- *Byte Code*
- Was ist ein *“Just in Time Compiler”*?
  - Generieren von Maschinencode während des Ablaufs eines Programms; z. b. bei Schleifenkonstrukten sinnvoll! -
- Was versteht man unter “Garbage Collection”?
  - Hintergrundprozess im BS sucht nach nicht mehr verwendeten Objekten und gibt deren Speicher frei; "null"! -
- Java und Security - siehe F19 -

19.7 Notizen