Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Prof. Dr. R. Dumke, 23. Februar 2010

Prüfungsklausur Programmierung[1]

Aufgabe 1 (6 Punkte): *Welche drei Komponenten bestimmen eine Programmierumgebung. Nennen Sie jeweils ein Komponentenbeispiel für eine Java-Programmierumgebung.*

Lösung:

Softwareprozessoren, Dokumentationen, Bibliotheken **3 P**
(*aus U=({Proz}, Dok, Appl./Lib.)*)

Eclpise (Java), JavaDoc, Java-Pakete **3 P**

Aufgabe 2 (6 Punkte): *Gegeben sei die folgende Syntaxbeschreibung in der BNF:*

```
<fallanweisung> ::= if (<b1>) a1 [else <a2>] |
                    switch ( <b2> ) {case <b3>: <a3>} [default: <a4>]
```

mit den speziellen Ausdrücken b1, b2 und b3 und den Anweisungen a1, a2, a3 und a4, die jeweils mit einem Semikolon abgeschlossen werden und mittels geschweifter Klammer zusammengefasst werden können. Welche (Java-) Codes entsprechen dieser Beschreibung?

```
a) if a > 3 b = 7;                  b) switch (b) {case 3: ; case 4: b = 5;}
c) if (b = 0) then a = 5;           d) if (a < 7 && b > 0) b = 7; else a == 7;
e) switch (c) default: b = 7;       f) if (a > b) else b = 7;
```

Lösung:

a) `if a > 3 b = 7;` fehlende Bedingungsklammern (falsch) **1 P**

b) `switch (b) {case 3: ; case 4: b = 5;}` korrekt **1 P**

c) `if (b = 0) then a = 5;` Vergleich nur mit ‘==’ und ‘then’ nicht zulässig (falsch) **1 P**

d) `if (a < 7 && b > 0) b = 7; else a == 7;` ‘==’ nicht zulässig (falsch) **1 P**

e) `switch (c) default: b = 7;` korrekt nach obiger Regel, gilt aber nicht in Java! **1 P**

f) `if (a > b) else b = 7;` fehlende (leere) Anweisung (falsch) **1 P**

Aufgabe 3 (6 Punkte): *Welche drei Grundkomponenten gehören zum Java-Ereignismodell? Welche Ereignisse erzeugen jeweils die Klasse MenuItem, Checkbox und Window?*

Lösung:

Model, Listener, (Action) Event **1 P**

MenuItem: ActionEvent **1 P**

Checkbox: ItemEvent **1 P**

Window: WindowEvent **1 P**

Aufgabe 4 (4 Punkte): *Was ist der Unterschied zwischen Algorithmenkorrektheit und Programmkorrektheit? Beschreiben Sie kurz zwei dynamische Programmtestmethoden.*

Lösung:

Algorithmenkorrektheit: (formal) beweisbar **1 P**

Programmkorrektheit: nicht beweisbar; folgt: (systematisches) Testen **1 P**

[1] **Allgemeine Hinweise:** Schreiben Sie auf jedes Blatt Ihren Namen, Ihre Matrikelnummer und eine Seitennummer. Als Hilfsmittel sind nur Schreibmaterialien zugelassen!

White-Box-Test: Programmabarbeitung unter Einsicht in die strukturellen bzw. internen Programmeigenschaften **1 P**

Black-Box- oder Funktionstest: Programmabarbeitung mit vorgegebenen Testdaten und (einfacher) Ergebniskontrolle **1 P**

(aber auch: Performance-, Stresstest, Tracing usw.)

Aufgabe 5 (6 Punkte): *Was ist der Unterschied zwischen Überladen und Überschreiben von Methoden in Java? Ordnen Sie die Java-Elemente Interface, abstrakte Klasse, finale Klasse und innere Klasse der folgenden, jeweils typischen Eigenschaft zu.*

a) keine Instanz bildbar, b) kann sich auf mehrere Klassen beziehen,
c) kann anonyme Klasse sein, d) kann nicht überschrieben werden.

Lösung:

Überschreiben: „Wirkung der Vererbung“ **1 P**

Überladen: unterschiedliche Signaturen für dieselbe Methode **1 P**

a) abstrakte Klasse, **1 P**

b) Interface, **1 P**

c) innere Klasse , **1 P**

d) finale Klasse. **1 P**

Aufgabe 6 (4 Punkte): *Erklären Sie kurz zwei vordefinierte Annotationen. Erläutern Sie zwei JUnit-Annotationen, die im Java-Quelltext den OO-Programmtest spezifizieren.*

Lösung:

@Retention: definiert Annotationslebensdauer **1 P**

@Deprecated: kennzeichnet (u.U.) veraltetes Java **1 P**

(*@Documented usw.*)

@Test: in den JUnit-Test einbezogen **1 P**

@Ignore: aus dem Test „herausgenommen“ **1 P**
(*usw.*)

Aufgabe 7 (4 Punkte): *Was versteht man unter einer Beweisskizze? Charakterisieren Sie kurz die Begriffe Testfall, Testabdeckung und Error Propagation.*

Lösung:

Beweisskizze: Angabe von Zusicherungen (Assertions) als Kommentare **1 P**

Testfall: definiert zu testenden Programmabschnitt, die Testdaten und die zu erwartenden Ergebnisse **1 P**

Testabdeckung: Maß im White-Box-Test für die Menge bzw. Art der „abgedeckten“ Programmpfade **1 P**

Error Propagation: „Fortpflanzung“ eines Fehler z. B. durch Vererbung und/oder Nachnutzung **1 P**

Aufgabe 8 (12 Punkte): *Ordnen Sie die folgenden Methoden den jeweiligen Java-Klassen zu:* `getAdler(), setText(), usePattern(), getPort(), getLanguage(), setID(), nextElement(), isAlive(), logout(), drawOval(), isEmpty(), nextInt().`

```
a) URL              b) LoginContext     c) Matcher              d) Thread
e) Hashtable        f) Inflater         g) TimeZone             h) Scanner
i) Graphics         j) TextField        k) StringTokenizer      l) Locale
```

Lösung:

a) `getPort()`	**1 P**	g) `setID()`	**1 P**
b) `logout()`	**1 P**	h) `nextInt()`	**1 P**
c) `usePattern()`	**1 P**	i) `drawOval()`	**1 P**
d) `isAlive()`	**1 P**	j) `setText()`	**1 P**
e) `isEmpty()`	**1 P**	k) `nextElement()`	**1 P**
f) `getAdler()`	**1 P**	l) `getLanguage()`	**1 P**

Aufgabe 9 (2 Punkte): *Erläutern Sie kurz die OO-Qualitätsindikatoren „ausgeschlagenes Erbe“ und „Importchaos“. Beschreiben Sie kurz zwei Formen der Java-Programmoptimierung.*

Lösung:

ausgeschlagenes Erbe: *Methoden der Oberklasse werden zumeist nichtleer redefiniert*

Importchaos: *die Imports sind redundant und nicht explizit angegeben*

1 P

*das **Systemverhalten** des möglicherweise vorhandenen Systems, dessen Teil das Programm darstellt,*

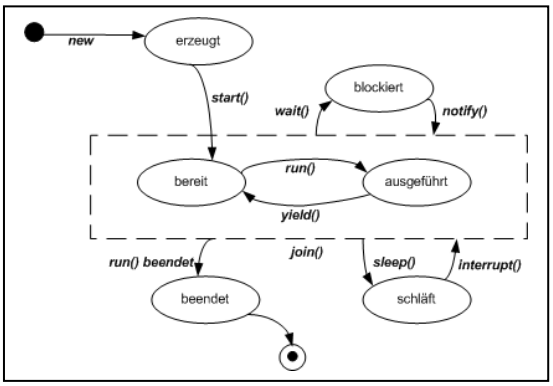
*das **Leistungsverhalten** des Programms selbst und*

*das **Ressourcenverhalten** des jeweiligen Rechners bzw. des Betriebssystems*

1 P

Aufgabe 10 (6 Punkte): *Erklären Sie drei unterschiedliche Zustände eines Java-Thread. Erläutern Sie kurz die drei Interfaces als komplexe Datengrundstrukturen des Java Collection Framework.*

Lösung aus:



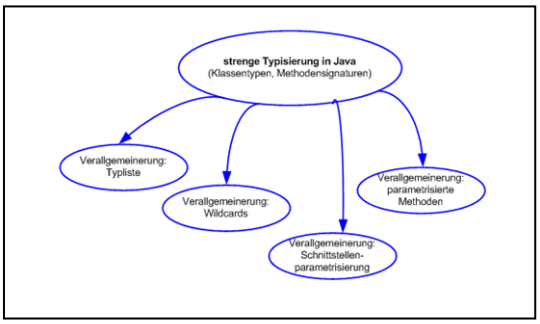
mit ganz kurzen Erläuterungen **3 P**

Collecion Framework Interfaces:

Set	: Daten-/Wertemenge	**1 P**
Map	: indexierte Wertemenge	**1 P**
List	. Werteliste	**1 P**

Aufgabe 11 (4 Punkte): *Charakterisieren Sie kurz drei Formen generischer Typbildung und -anwendung. Welche Aufgabe hat der Garbage Collector in Java?*

Lösung:



3 P

Die 6 formalen Darstellungsformen sind Anwendungen dieser Grundformen und werden natürlich auch angerechnet!

Garbage Collector: bereinigt Objektsituation im laufenden Java-Programm **1 P**