1.Programmstruktur

(a)Erläutern Sie das Konzept des Java Bytecodes: Wie entsteht er, wie wird er ausgeführt und wieso werden Java-Programme durch ihn plattformunabhängig?

Compiler (für alle Plattformen) übersetzt Quellencode in Bytecode, der unabhängig von einem bestimmten Prozessor ist, jedoch nicht ausführbar ist. Java-Interpreter (JVM) analysiert Bytecode und führt diesen aus.

(b) In der Datei C.java haben Sie mittels

```
package p.q;
c].ass 0 {..}
```

eine Klasse definiert. Jetzt wollen Sie der Klasse den neuen Namen D **geben.Außerdem soll sie nach der Umbenennung zum Paket r. s gehören. Was müssen Sie tun? In welchem Verzeichnis muss die Datei stehen?**

- Eclipse ➔ Refactor in D Umbennenung des Packet in r.s
- r\ s

(b2003) Die Methode foo () sei in der Toplevel-Klasse C definiert; die zum Paket p. q. r gehört. Wie heißt die Byte-Code-Datei, in der sich der zu foo () gehörige Byte-Code befindet ? In welchem Verzeichnis muss sich diese Byte-Code-Datei im Datei-System befinden? Welchen Einfluss hat die Umgebungsvariable CLASSPATH?

C:Class
P\ q\ r
CLASSPATH gibt den Pfad zu den Bibliotheksklassen der Javacose API bzw. zu Benutzerdefinierten Klassen an. In diesem Fall vervollständigt die JVM durch sie den Pfad des Pakets.

(c) Erläutern Sie genau, was der Ausdruck

new ActionListener(void actionperformed(ActionEvent ae){...});
bewirkt!

Es wird ein neuen Listener-Interface Actionlistener erstellt mit der deklarierten Methode actionPerformed, welche ein Objekt ae zur Behandlung der Events übergeben bekommt

(c2003)Warum kann man in einer Klassenmethode (also einer Methode, die mit dem Schlüsselwort static vereinbart wurde) nicht die Objektreferenz this verwenden?

In Static – Methoden ist kein this vorhanden, da kein Objekt existiert.

(c2002) Was bewirkt eine import-Anweisung?

Importiert den Namensraum einer Klasse vergleichbar mit der Anweisung (using namespace in C++)

(d)Dürfen zwei Attribute einer Klasse den gleichen Namen besitzen? Dürfen zwei Methoden einer Klasse den gleichen Namen besitzen? Begründen Sie Ihre Antwort!

Zwei Attribute einer Klasse dürfen nicht der gleichen Namen besitzen, da sie nicht eindeutig zu unterschieden waren.
Unterscheiden sich Methoden in einer Signatur, dürfen sie den gleichen Namen besitzen.

(d2003)Was versteht man unter einer anonymen Klasse, einer lokalen Klasse und einer Elementklasse?

Anonyme Klassen sind lokale Klassen ohne Namen.
Elementklassen sind Elemente einer Klasse.
Lokale Klassen werden im Rahmeneiner Deklarationsanweisung definiert

(e)Das Datenfeld x sei innerhalb der Klasse 0 im Paket p ohne eines der Sichtbarkeitsattribute (public, private oder protected) definiert:

```
package p;
dass C{int x;}
```

Erläutern Sie, an welchen Stellen eines Programmes der Name x sichtbar ist und wo nicht!

Sichtbar für Klassen bzw. Schnittstellen desselben Pakets
Nicht sichtbar für Klassen bzw. Schnittstellen aus anderen Paketen und aus Unterpaketen

(e2003)Dürfen zwei Attribute einer Klasse den gleichen Namen besitzen? Dürfen zwei Methoden einer Klasse den gleichen Namen besitzen? Begründen Sie Ihre Antwort!

Quiz1 Frage 45
zwei Attritbute einer Klasse dürfen nicht den gleichen Namen besitzen, da sie nicht eindeutig wären.
Unterscheiden sich Methoden in ihrer Signatur, dürfen sie den gleichen Namen besitzen.

(f2003)Zählen Sie alle Arten von Bezeichnern auf, die in einem Java Programm vereinbart werden können!

Klasse, Methode, Konstruktor, Datenfeld, lokale Variable, Parameter, Schnittstelle, Paket.

(f2002)Geben Sie die kürzestmögliche Klassendefinition an!

```
class A{}
```

a) Warum kann ein Java Programm auf jeder Plattform ausgefuehrt werden?

Es kann auf jeder Plattform ausgefuehrt werden, da es von einer virtuellen Maschine aufgerufen wird.
Die gibt es auf jedem System bzw. jeder Plattform.

b) Wieso kann ein Applet mit fast jedem Browser ausgefuehrt werden?

Weil ein Applet von HTML-Seiten geladen wird und fast jeder Browser HTML-Seiten ausfuehren kann.

c) Warum braucht ein Applet keine Methode *void main(String[ ] args)* ?

Da es eine Unterklasse der Klasse Applet ist und es daher keine Methode main() hat.
Es muss mit eine der drei folgenden Methoden implementiert werden:
-init()
-start()
-oder paint()

d) Welche Methode eines neuen Applets ruft ein Browser zuerst, welche beim Verlassen der HTML-Seite?

Zuerst ruft er die Methoden init(), start() oder paint() auf und beim Verlassen ruft er die Methode destroy() auf.

e) Warum verwendet man eine *import*-Anweisung?

Man verwendet import-Anweisungen um Bibliotheken zu importieren wie Include-Anweisungen bei C++.

(e)Das Datenfeld x sei innerhalb der Klasse C im Paket p ohne eines der Sichtbarkcitsattribute (public, private oder protected) definiert:

```
package p;
 class C {int x;}
```

Erläutern Sie, an welchen Stellen eines Programmes dieser Name sichtbar ist und wo nicht!

- sichtbar für Klassen bzw. Schnittstellen desselben Pakets
- nicht sichtbar für Klassen bzw. Schnittstellen aus anderen Paketen und aus Unterpaketen

2. Typen

(a)Zählen Sie alle Arten der Initialisierung von Datenfeldern einer Klasse auf, in

der
Reihenfolge ihrer Ausführung!

Default-Werte von Datenfeldern werden durch eine manuelle Initialisierung überschrieben.
Initialisierungen, die im Konstruktor durchgeführt werden, überschreiben sowohl Default-Werte als auch die Werte einermanuellen Initialisierung und die Werte der Initialisierungen einer Initialisierungsblock.

(a2003)In Java werden Referenztypen und primitive Typen unterschieden. Erläutern Sie den Unterschied anhand der Zuweisung x = y und anhand des booleschen Ausdrucks x == y

Referenztypen zeigen auf Objekte

- primitive Typen werden mit Werten initialisiert, Zugrif erfolgt über Name
  X = Y
- Referenztyp: X zeigt auf dasselbe Objekt wie Y
  Primitiver Typ: X bekommt Wert von Y zugewiesen
  X == Y
  Referenztyp: Es wird verglichen, ob X und Y auf das gleiche Objekt zeigen
  Primitive Typen: Es werden die Werte von X und Y verglichen

(b)Ist es möglich, die genaue Ausführungsdauer einer Anweisung anzugeben? Begründen Sie!

(b2003)Warum ist ein Laufzeitsystem mit automatischer Speicherbereinigung für eine Echtzeitanwendung ungeeignet?

- Es ist nicht bekannt, wann der Garbage-Kollektor Speicherplatz freisetzt.
- Da es bei einem Echtzeitsystem auf Sekunden ankommt, könnte es fatal sein, wenn genau zu einem entscheidenden Zeitpunkt der Garbage-Kollektor beginnt, aufzuräumen.

(b2002)Welche Arten von Referenztypen werden in Java unterschieden?

Klassen – Typen, Array-Typen, Schnittstellen-Typen

(c)Ist für ein Array, wie z.B. mit int [1 la = new int [8]; eingeführt, der Ausdruck la. toString ~zulässig? Begründen Sie!

- Der Ausdruck ist zulässig.
- Begründung : Klasse Objekt (Wurzel alle Klassen) vererbt diese Methode an alle anderen Klassen

(c2003)Zählen Sie alle primitiven Typen von Java auf und geben Sie ihre Werte den Speicherbedarf in Bit an!

Quiz 1 Frage 14
Byte: 8-Bit, Short: 16-Bit, int: 32-Bit, long: 64-Bit, float: 32-Bit, double: 64-bit, care: 16-bit, unicode boolian: true/ false

(d)Unter welchen Umständen wird die Methode finalize () der Klasse Object ausgeführt? Ist es garantiert, dass diese Methode für *jedes* Objekt irgendwann einmal aufgerufen wird?

- Entfernt der garbage Collector ein Objekt aus dem Speicher, so wird zuvor die Methode finalize() für dieses abgearbeitet.
- Es ist nicht garantiert, möchte man absolut sicher sein dass, die Methode finalize() für jeder Objekt abgearbeitet wird so muss sie explizit vom Programmierer aufgerufen werden.

(d2002)Ist es möglich,dass der für ein Objekt benötigte Speicher nicht auf dem Heap angelegt wird? Begründen Sie!

Nein, da Objekte dynamisch erzeugt und über Referenzen angesprochen werden.
Nur auf dem Heap möglich

(e)Wie kann man in einem Java-Programm den nicht mehr benötigten Speicherplatz eines Objektes explizit wieder freigeben?

- Die Speicher Freigabe übernimmt der Garbage Collector.
- Explizit kann der Speicher jedoch mit der finalize()-Methode freigegeben werden.

(e2003)Kann die Methode Obj ect wait () auch auf ein Array angewendet werden? Begründen Sie!

Ja, die Wurzelklasse Object vererbt diese Methode an alle anderen Klassen und an Arrays.

(e2002)Welchen Einfluss hat ein Java Programm auf die Freigabe von Speicherplatz auf dem Heap?

- keinen
- Garbeige-Kollektor der VM übernimmt die Speicherplatzbereinigung

(f2002) Sind Felder (d.h. Arrays) auch Objekte? Kann also auch auf ein Feld die Methode toString () angewendet werden?

Ja

a) Wenn eine Klasse C eine Schnittstelle I implementiert, welche Methoden von I muessen dann in C oder einer Vorgaengerklasse von C implementiert sein?

Es muessen alle Methoden implementiert werden, da die Klasse sonst abstrakt wird und nicht instantiert werden kann.

c) Welche Typen werden in Java unterschieden? Zu welcher Kategorie gehoert ein Feldtyp?

Es gibt Referenzen und Primitives(Datentypen). Ein Feldtyp gehoert zu der Kategorie Referenzen.

b) Kann man einen Feldtyp definieren, dessen Element-Typ ein Schnittstellentyp ist? Ist also z.B. *ActionListener[ ]* ein zulaessiger Typ ?

Nein, Weil ein Schnittstellentyp auch ein Referenztyp und kein Elementtyp ist.

(e)Kann die Methode Obj ect wait () auch auf ein Array angewendet werden? Begründen Sie!

Ja, die Wurzelklasse Object vererbt diese Methode an alle anderen Klassen und an Arrays.

(a)Jn Java werden Referenztypen und primitive Typen unterschieden. Erläutern Sie den Unterschied anhand der Zuweisung x =y und anhand des booleschen Ausdrucks x ==y!

Referenztypen zeigen auf Objekte

- primitive Typen werden mit Werten initialisiert, Zugrif erfolgt über Name
  X =Y
- Referenztyp: X zeigt auf dasselbe Objekt wie Y
  Primitiver Typ: X bekommt Wert von Y zugewiesen
  X ==Y
  Referenztyp: Es wird verglichen, ob X und Y auf das gleiche Objekt zeigen
  Primitive Typen: Es werden die Werte von X und Y verglichen

3.Speicherbereinigung

(a)Zählen Sie alle Arten der Initialisierung von Datenfeldern einer Klasse auf, in der Reihenfolge ihrer Ausführung!

Default-Werte von Datenfeldern werden durch eine manuelle Initialisierung überschrieben.
Initialisierungen, die im Konstruktor durchgeführt werden, überschreiben

sowohl Default-Werte als auch die Werte einermanuellen Initialisierung und die Werte der Initialisierungen einer Initialisierungsblock.

(a2003)In Java werden Referenztypen und primitive Typen unterschieden. Erläutern Sie den Unterschied anhand der Zuweisung x = y und anhand des booleschen Ausdrucks x == y

Referenztypen zeigen auf Objekte

- primitive Typen werden mit Werten initialisiert, Zugrif erfolgt über Name
  X = Y
- Referenztyp: X zeigt auf dasselbe Objekt wie Y
  Primitiver Typ: X bekommt Wert von Y zugewiesen
  X == Y
  Referenztyp: Es wird verglichen, ob X und Y auf das gleiche Objekt zeigen
  Primitive Typen: Es werden die Werte von X und Y verglichen

(b)Ist es möglich, die genaue Ausführungsdauer einer Anweisung anzugeben? Begründen Sie!

(b2003)Warum ist ein Laufzeitsystem mit automatischer Speicherbereinigung für eine Echtzeitanwendung ungeeignet?

- Es ist nicht bekannt, wann der Garbeige-Kollektor Speicherplatz freisetzt.
- Da es bei einem Echtzeitsystem auf Sekunden ankommt, könnte es fatal sein, wenn genau zu einem entscheidenden Zeitpunkt der Garbeige-Kollektor beginnt, aufzuräumen.

(b2002)Welche Arten von Referenztypen werden in Java unterschieden?

Klassen – Typen, Array-Typen, Schnittstellen-Typen

(c)Ist für ein Array, wie z.B. mit int [1 la = new int [8]; eingeführt, der Ausdruck la. toString ~zulässig? Begründen Sie!

- Der Ausdruck ist zulässig.
- Begründung : Klasse Objekt (Wurzel alle Klassen) vererbt diese Methode an alle anderen Klassen

(d)Unter welchen Umständen wird die Methode finalize () der Klasse Obj ect ausgeführt? Ist garantiert, dass diese Methode für *jedes* Objekt irgendwann einmal aufgerufen wird?

- Entfernt der garbage Collector ein Objekt aus dem Speicher, so wird

zuvor die Methode finalize() für dieses abgearbeitet.
- Es ist nicht garantiert, möchte man absolut sicher sein dass, die Methode finalize() für jeder Objekt abgearbeitet wird so muss sie explizit vom Programmierer aufgerufen werden.

(d2003)Unter welchen Umständen wird die Methode finalize () der Klasse Object aufgerufen?

- bevor der Garbage-Kollektor ein Objekt aus dem Speicher entfernt.
- Wenn die Methode explizit vom Programmierer aufgerufen wird.

(e)Wie kann man in einem Java-Programm den nicht mehr benötigten Speicherplatz eines Objektes explizit wieder freigeben?

Die Speicher Freigabe übernimmt der Garbage Collector.
Explizit kann der Speicher jedoch mit der finalize()-Methode freigegeben werden.

(e2003)Kann die Methode Obj ect wait () auch auf ein Array angewendet werden? Begründen Sie !

Ja, die Wurzelklasse Object vererbt diese Methode an alle anderen Klassen und an Arrays.

(e2002)Welchen Einfluss hat ein Java Programm auf die Freigabe von Speicherplatz auf dem Heap?

Keinen, Garbage-Kollektor der VM übernimmt die Speicherplatzbereinigung

(f2002) Sind Felder (d.h. Arrays) auch Objekte? Kann also auch auf ein Feld die Methode toString () angewendet werden?

Ja

a) Was bedeutet automatische Speicherbereinigung (garbage collection)?

Garbage Collection. In Java gibt es im Gegensatz zu C und C++ keine explizite Anweisung zur Freigabe nicht mehr benötigten Speichers (diese ist tatsächlich eine häufige Fehlerquelle). Statt dessen übernimmt ein nebenläufiger Thread die Aufgabe, nicht mehr referenzierte Objekte zu entdecken und ihren Speicher freizugeben. Dazu ruft er gegebenenfalls die finalize()-Methode des Objekts auf.

b) Warum sollte ein Java Laufzeitsystem eine automatische Speicherbereinigung besitzen?

Wenn zum Anlegen eines Objektes der vohandene Platz im Heap nicht ausreicht, muss die virtuelle Maschine versuchen, durch eine

Speicherbereinigung des Garbage Collectors Platz zu gewinnen. Es gibt keinen delete Befehl wie bei C++, wodurch der Speicher vom Programmierer wieder freigegeben werden koennte. Dies kann nur der Garbage Collector!

c) Warum ist ein Laufzeitsystem mit automatischer Speicherbereinigung fuer ein Echtzeitsystem fast immer ungeeignet?

Weil es keine Zusicherung gibt, wann der speicher freigegeben wird.Der Garbage Collector nutzt Zeiträum in denen de Prozessor frei ist und wickelt seine Aufgabe in dieser Zeit ab. In einem Echtzeitsystem kann es sein, dass dieser Zeitpunkt nie eintritt und damit der speicher nie bereinigt wird !

4. Bibliotheken

a) Wozu dient ein *LayoutManager*, wie z.B. die Klasse *java.awt.BorderLayout*?

Die Layout-Manager wurden eingefuehrt, um die Anordnung mehrerer Komponenten in einem Container zu organisieren. Bei jeder Veraenderung der Fenstergroesse wird der Layout-Manager angesprochen.
Mit einem Layout-Manager werden die Komponenten nach bestimmten Vorgaben auf einer Bedienoberflaeche bzw. in einem Container angeordnet.

b) Wie erkennen Sie bei einer Unterklasse von java.awt.Component, welche Ereignisse eines ihrer Objekte erzeugen kann?

Um in Java ein GUI zu erstellen, ordnet man die GUI-Komponenten (abgeleitet von java.awt.Component) in sogenannten Containem (von java.awt.Container).

Ein Container kann ein Fenster (von java.awt.Window) oder eine rechteckige Flaeche (von java.awt.Panel) innerhalb eines Fensters, beziehungsweise einer anderen rechteckigen Flaeche sein. Container koennen ineinander verschachtelt werden.

Es gibt grundsaetzlich drei Arten von Komponenten :
- Schnittstellen zum Benutzer, wie Buttons, Scrollbars, Text-Labels,...
- Bereiche, die andere Komponenten enthalten koennen: Containers
- Fenster: Windows, Dialogs, Frames

AWT: Abstract Window Toolkit

c) Wozu dient die Methode *String toString()*?

Fuer eine kurze Angabe welcher Fehler aufgetreten ist.Mit der Methode erfolgt die Konvertierung zweier Objekte.

d) Besitzt jedes Objekt eine Methode *String toString()*?

Ja , Siehe c)

5.) Threads

(a2002)Zeichnen Sie ein Zustandsübergangsdiagramm für einen Java Thread. Tragen Sie insbesondere die Übergänge ein, die durch Aufrufe der Methoden Start(),yield (), sleep () , wait ()und notify () bewirkt werden. Welchen Zustand nimmt der Thr~ad nach Ablauf der Methode run () ein, welchen direkt nach der Erzeugung?

Siehe Script…

(b)Auf ein Objekt der Klasse java. lang. String sollen mehrere Threads zugreifen können. Was ist zu tun? Begründen Sie!

Es muss nicht synchronisiert werden, da string-Objekte nicht veränderbar sind.

(b2002)Was unterscheidet einen Java Thread von einem Betriebssystem-Prozess?

OS-Prozess: Ready-to-run, running, blocked
Thread: new, ready-to-run, running, blocked, dead

(c)Innerhalb eines Methodenrumpfes wird die Methode wait () aufgerufen. Bei Ausführung des Programmes wird an dieser Stelle aber eine IllegalMonitorstateException ausgelöst. Woran mag das liegen?

Wird wait () zu einem Objekt aufgerufen, das gerade nicht als Schlüsselobjekt für einen synchronisierten Abschnitt benutzt wird, so wird die oben genannte Exception geworfen.

(c2002)Wann endet ein Thread?

wenn Methode run() beendet wird.

(d)Wie lassen sich in Java kritische Abschnitte spezifizieren?Welche Objekte können als Sperre fungieren?

Mit synchonized (synchronisation).
Bei synchronisierten Instanzmethoden das eigene Objekt.
Bei synchronisierten Klassenmethoden das Objekt der Klasse class.
Bei synchronisierten Blöcken das Objekt, auf das die übergebene Referenz zeigt.
Es muss nicht synchronisiert werden, da string- objekte nicht verändert sind.
Bei StringBuffer-Objekt muss synchronisiert werden.

(d2002)Wie lassen sich in Java kritische Abschnitte spezifizieren?
Mit synchronized (Synchronisation)

(e)Innerhalb eines Methodenrumpfes wird die Methode wait () aufgerufen. Bei Ausführung des Programmes wird an dieser Stelle aber eine IllegalMonitorStateException ausgelöst. Woran mag das liegen?

Wird wait() zu einem Objkt aufgeführt, das gerade nicht als Schlüsselobjekt für einen sychronisierten Abschnitt benutzt wird, so wird die oben genannte Exeption geworfen.

a) Jeder Thread besitzt die Methoden start() und run(). Was bewirken Sie? Wann wird die Methode run() aufgerufen?

start(): Es bewirkt das diese Methode aufgerufen wird damit der Thread gestartet wird.
run(): Es bewirkt das das laufende Programm innerhalb des Threads implementiert wird. Dies geschieht mit dem Ausdruck public void run(). Sie wird innerhalb des Threads ausgefuehrt.

b) Koennen mehrere nebenlaeufige Threads auf ein einzelnes String-Objekt beliebig zugreifen, oder muessen sie diese Zugriffe synchronisieren, um Wettbewerbssituationen (race conditions) auszuschliessen? Begruenden Sie Ihre Antwort! Aendert sich etwas, wenn statt auf ein String-Objekt auf ein Stringbuffer-Objekt zugegriffen wird?

(b)Auf ein Objekt der Klasse java. lang. String sollen mehrere Threads zugreifen können. Was ist zu tun? Begründen Sie!

Es muss nicht synchronisiert werden, da string-Objekte nicht veränderbar sind.

(c)Innerhalb eines Methodenrumpfes wird die Methode wait () aufgerufen. Bei Ausführung des Programmes wird an dieser Stelle aber eine IllegalMonitorstateExcept Ion ausgelöst. Woran mag das liegen?

Wird wait () zu einem Objekt aufgerufen, das gerade nicht als Schlüsselobjekt für einen synchronisierten Abschnitt benutzt wird, so wird die oben genannte Exception geworfen.

(d)Wie lassen sich in Java kritische Abschnitte spezifizieren? Welche Objekte können als Sperre fungieren?

- Mit synchonized (synchronisation).
- Bei synchronisierten Instanzmethoden das eigene Objekt.
- Bei synchronisierten Klassenmethoden das Objekt der Klasse class.
- Bei synchronisierten Blöcken das Objekt, auf das die übergebene Referenz zeigt.

(c)Wann endet ein Thread?

wenn Methode run() beendet wird.

(d)Wie lassen sich in Java kritische Abschnitte spezifizieren?

Mit synchronized (Synchronisation)

6.) Ausnahmen

---

a) Welche Ausgabe bewirkt ein Aufruf von foo() im untenstehenden Programm?

```
public static foo(){
  char c ='a';
  int i =0;
  try {
    i=i/i;
    c ='b';
  }
catch (Error e)              {System.out.println("Fehler")}
catch (ArithmeticException e) {System.out.println("Teiler gleich null")}
catch (Exception e)          {System.out.println("Ausnahme")}
}
```

b) Kann man im folgenden Programmfragment Anweisungen fuer den try-Block finden, so dass die letzte catch-Anweisung ausgefuehrt wird , also ein Aufruf von foo() zur Ausgabe von Werfbar fuehrt? Wenn ja, wie lauten sie, wenn nein, begruenden Sie Ihre Antwort.

```
public static foo(){
try{
...
}
catch (Error e)              {System.out.println("Fehler")}
catch (Exception e)          {System.out.println("Ausnahme")}
catch (Throwable t)          {System.out.println("Werfbar")}
}
```

a) Welche Ausgabe bewirkt ein Aufruf von foo() im untenstehenden Programm?

```
public static foo(){
  char c ='a';
  int i =0;
  try {
    i=i/i;
```

```
    c ='b';
  }
catch (Error e)                {System.out.println("Fehler")}
catch (ArithmeticException e) {System.out.println("Teiler gleich null")}
catch (Exception e)            {System.out.println("Ausnahme")}
}
```

Das programm gibt folgendes aus:
Teiler gleich null

b) Kann man im folgenden Programmfragment Anweisungen fuer den try-Block finden, so dass die letzte catch-Anweisung ausgefuehrt wird , also ein Aufruf von foo() zur Ausgabe von Werfbar fuehrt? Wenn ja, wie lauten sie, wenn nein, begruenden Sie Ihre Antwort.

```
public static foo(){
try{
...
}
catch (Error e)                {System.out.println("Fehler")}
catch (Exception e)            {System.out.println("Ausnahme")}
catch (Throwable t)            {System.out.println("Werfbar")}
}
```

Ja, kann man:
throw new Throwable();

Aufgabe 3 (Nebenläufigkeit)

(a) Was versteht man unter einem kritischen Abschnitt?

- Eine Folge von Befehlen, die ein Thread nacheinander vollständig abarbeiten muss.

(b)Wie lässt sich in Java wechselseitiger Ausschluss garantieren? Welche Objekte können als Sperre dienen?

- Mittels Monitorkonzept.
- Bei Synchronisierten Instanzmethoden das eigene Objekt.
- Bei Synchronisierten Klassenmethoden das Objekt der Klasse class.
- Bei Synchronisierten Blöcken der Objekt, auf das die übergebene Referenz zeigt.

(c)In der API-Dokumentation werden die Methoden suspend ()' resume () und Stopp () der Klasse Thread als *deprecated* gekennzeichnet und werden möglicherweise in zukünftigen Versionen der Klassenbibliothek nicht mehr vorhanden sein. Davon abgesehen, warum wird dringend davon abgeraten, diese Methoden zu verwenden?

- Stopp() gibt alle sperren der Thread frei- kann zu Inkonsistenzen führen.
- Suspend () und resum() können zu einem Deadlock führen
    - Suspend() gibt keine sperren Frei.
    - Angehaltener Thread kann sperren halten.
    - Thread, der resum() aufrufen will, blockiert an einer sperre.

(d)Auf ein Objekt der Klasse j ava. lang. String sollen mehrere Threads zugreifen können. Was ist zu tun? Begründen Sie! Gilt Ihre Antwort auch für ein StringBuffer-Objekt?

Aufgabe 4 (Vererbung)

(a)Die Methode A. meth , Y) wird in der Klasse B überschrieben (s.u.). In welchem Zusammenhang müssen die Klassen AException und BException stehen? Begründen Sie!

```
class A {..
        void meth(X x, Y y)throws AException{. .}

      dass B extends A{.,.
        void meth(X x, Y y)throws BException{.
      }
```

(a2002) Ist folgende Klassendefinition zulässig? Begrunden Sie

Class C extends D, F {/*Klassenkörper folgt hier ...*/ }

nein, in Java gibt es keine Mehrfachvererbung.

(b)Eine Methode B. meth (. ..) werde von einer Methode A . meth (. . .) überschrieben. Wie verhält es sich mit den Vor- und Nachbedingungen der beiden Methoden, dürfen sie abgeschwächt oder verschärft werden? Begründen Sie Ihre Antwort mit dem Ersetzungsprinzip!

- Methode der abgeleiteten Klasse darf
    - Eine Nachbedingung nicht abschwächen.
    - Eine Vorbedingung nicht verschärfen.

(b2002)Ist folgende Klassendefinition zulässig? Begründen Sie!
class C implementa 1, J , K {/*Klassenkörper folgt hier ... */ }

Ja, eine Schnittstelle kann nicht nur eine einzige Scnittstelle erweitern, sondern

mehrere gleichzeitig

(c)Es sei 3 eine Unterklasse von A. Wie wird gewährleistet, dass bei Erzeugung eines
B-Objektes auch ein Konstruktor von A aufgerufen wird. Wie lautet die Regel, wenn
3 überhaupt keinen Konstruktor definiert?

- Mittels super()
- Compiler stellt immer einen Default-Konstruktor für jede klasse zur Verfügung

(d) Warum werden Konstruktoren nicht an abgeleitete Klassen vererbt?

- Da Konstruktoren nicht zu den Methoden von Klassen gehören

(d2003) In Java ist nur einfache Vererbung flur Klassen zulässig. Welchen Vorteil bietet dies beim Binden einer Methode?

- Performance geht beim Binden nicht verloren
- Fehlende und falsche Methoden-Aufrufe werden nicht erst zur Laufzeit des Programms festgestellt.

(d2002) Welche Konsequenz hat es, wenn eine Klassendefinition das Schlüsselwort final verwendet? Was bedeutet das gleiche Schlüsselwort bei einer Methodendefinition?

- Von finalen Klassen lassen sich keine weiteren Klassen ableiten.
- Finale Methoden können in einer Subklasse nicht überschrieben werden.

(c)Es sei B eine Unterklasse von A. Wie wird gewährleistet, dass bei Erzeugung eines B-Objektes auch ein Konstruktor von A aufgerufen wird. Wie lautet die Regel, wenn B überhaupt keinen Konstruktor definiert?

- Mittels super()
- Compiler stellt immer einen Default-Konstruktor für jede klasse zur Verfügung

(d) Welche Konsequenz hat es, wenn eine Klassendefinition das Schlüsselwort final verwendet? Was bedeutet das gleiche Schlüsselwort bei einer Methodendefinition?

- Von finalen Klassen lassen sich keine weiteren Klassen

ableiten.
- Finale Methoden können in einer Subklasse nicht überschrieben werden.

(g)In Java ist nur einfache Vererbung flur Klassen zulässig. Welchen Vorteil bietet dies beim Binden einer Methode?

- Performance geht beim Binden nicht verloren
- Fehlende und falsche MethodenAufrufe werden nicht erst zur Laufzeit des Programms festgestellt

(a)Die Methode A.me th (X, Y) wird, in der Klasse B überschrieben (s.u.). In welchem Zusammenhang müssen die Klassen AException und BException stehen? Begründen Sie!

```
class A {..
void meth(X x, Y y)throws AException{.}

class 3 extends A{..
void meth(X x, Y y)throws BException{. .}
```

(b)Eine Methode 3. meth (...) werde von einer Methode A . meth (...) überschrieben. Wie verhält es sich mit den Vor- und Nachbedingungen der beiden Methoden, dürfen sie abgeschwacht oder verschärft werden? Begründen Sie Ihre Antwort mit dem Ersetzungsprinzip!

- Methode der abgeleiteten Klasse darf
  - Eine Nachbedingung nicht abschwächen.
  - Eine Vorbedingung nicht verschärfen

(c)Es sei 3 eine Unterklasse von A. Wie wird gewährleistet, dass bei Erzeugung eines B-Objektes auch ein Konstruktor von A aufgerufen wird. Wie lautet die Regel, wenn B überhaupt keinen Konstruktor definiert?

- Mittels super()
- Compiler stellt immer einen Default-Konstruktor für jede klasse zur Verfügung

(d) In Java ist nur einfache Vererbung flur Klassen zulässig. Welchen Vorteil bietet dies beim Binden einer Methode?

- Performance geht beim Binden nicht verloren
- Fehlende und falsche Methoden-Aufrufe werden nicht erst zur Laufzeit des Programms festgestellt.

(Schnittstellen)

(a)Im Paket java. lang ist die Schnittstelle Runnable deklariert Kann man mit

Runnable r =new Runnable ();

ein Runnable Objekt erzeugen? Begründen Sie!

nein, da runable ein interface ist.

(b)Können in einer Schnittstellendefinition Attribute eines Objektes definiert werden?

Nein

(c)Erläutern Sie den Unterschied zwischen einer Schnittstelle und einer abstrakten Klasse!

- Abstrakte Klassen können variablen Konstruktoren implementierte und abstrakte Methoden enthalten.
- Schnittstellen können nur Konstruktoren und abstrakte Methoden enthalten

(d)Kann eine Klasse mehrere Schnittstellen implementieren? Anders ausgedrückt, ist dass A impleinents Ii, 12{ ..}

zulässig?

JA

(e) Die Klasse Thread besitzt u.a. einen Konstruktor Thread (Runnable), dessen Parameter den Schnittstellentyp Runnable besitzen muss. Was bewirkt dieser Konstruktor? An welchen Stellen kann bei Verwendung dieses Konstruktors die Methode run () implementiert werden? Vergleichen Sie mit der Schnittstelle ActionListener und ihrer Verwendung bei der Registrierung an der. Ereignisquelle mittels addActionListener (..) .

- Es wird dafür gesorgt, das die Run-Methode des runable-Objektes ausgeführt wird.
- in der Klasse, welche die Schnittstelle runable implementiert.

(Implementierung eines Beispiel)

Gegegeben sei die folgende Schnittstelle für eine nach dem FIFO-Prinzip organisierte Warteschlange unbeschränkter Größe :

```
interface Schlange
{
   //Fügt ein von null verschiedenes Objekt am Ende der Schlange ein
   public void einfuegen(Object obj)throws IllegalArgumentException;

   // Entfernt das Objekt am Anfang der Schlange
   // Liefert null, wenn die Schlange leer ist
   public Object entfernen();

   // Liefert true, falls Schlange leer, false sonst
   public boolean istLeer();

   // Liefert Länge der Schlange
   public int gibLaenge();

   // Darstellung als String: beginnt mit "<",
   // gefolgt von einer String-Darstellung fuer jedes Element,
   // jede getrennt durch Leerraeume, endet mit ">"
   public String toString();
}
```

(a) Schreiben Sie ein Test-Programm zum Test der Implementierungen in (b) und (c): Darin soll eine Schlange q mit den 4 Elementen 1,4,'a',"Element" erzeugt werden, dann soll q auf die Standardausgabe mittels der Methode toString() ausgegeben werden und schliesslich soll es q vollständig leeren.
Hinweis: Beachten Sie, dass int- und char-Werte nicht vom Typ Object sind ! Abhilfe?

```
//Test-Programm zu b.)

public static void main(String[] args)
{
      VerketteteSchlange q =new VerketteteSchlange();
      q.einfuegen(new Integer(1));
      q.einfuegen(new Integer(4));
      Character charObject =new Character('a');
      q.einfuegen(charObject);
```

```
        q.einfuegen("Element");
        System.out.println(q.toString());
        while(q.entfernen() !=null)
           q.entfernen();
    }
```

//Test-Programm zu c.)

```
    public static void main(String[] args)
    {
        VectorSchlange q =new VectorSchlange();
        q.einfuegen(new Integer(1));
        q.einfuegen(new Integer(4));
        Character charObject =new Character('a');
        q.einfuegen(charObject);
        q.einfuegen("Element");
        System.out.println(q.toString());
        while(!q.istLeer())
           q.entfernen();
    }
```

(b) Schreiben Sie eine Klasse VerketteteSchlange, die die Schnittstelle Schlange implementiert und die Elemente miteinander verkettet.
Hinweis: Führen Sie eine geschachtelte Top-Level Klasse Element mit Attributen inhalt, nachfolger... ein.

```
    public class VerketteteSchlange implements Schlange
    {
       static class Element{//Top Level Klasse Element
          Object inhalt;
          Element nachfolger =null;
          Element vorgaenger =null;
          Element(Object obj){
             inhalt =obj;
          }
       }
       private Element erstes =null;
       private Element letztes =null;
       int laenge =0;
       public VerketteteSchlange()
       {
          super();
       }
       //Fügt ein von null verschiedenes Objekt am Ende der Schlange ein
       public void einfuegen(Object obj)throws IllegalArgumentException
       {
          Element neu =new Element(obj);
          if(erstes ==null)
          {          erstes =neu;
```

```
        letztes = neu;
    }
    else
    {           letztes.nachfolger = neu;
        neu.vorgaenger = letztes;
        letztes = neu;
    }
    laenge++;
}


// Entfernt das Objekt am Anfang der Schlange
// Liefert null, wenn die Schlange leer ist
public Object entfernen()
{
    if(erstes != null)
    {
        Element buffer = null;
        buffer = erstes.nachfolger;
        erstes = buffer;
        if(erstes != null)
            erstes.vorgaenger = null;
        else
            letztes = null;
        laenge--;
        return erstes;
    }
    else
        return null;
}
// Liefert true, falls Schlange leer, false sonst
public boolean istLeer()
{
    return(erstes == null);
}
// Liefert Länge der Schlange
public int gibLaenge()
{
    return laenge;
}
// Darstellung als String: beginnt mit "<",
// gefolgt von einer String-Darstellung fuer jedes Element,
// jede getrennt durch Leerraeume, endet mit ">"
public String toString()
{
    StringBuffer ergebnis = new StringBuffer("<");
    Object inhalt;
    for(Element e = erstes;e != null;e = e.nachfolger)
```

```
        {
            inhalt = e.inhalt;
            if(inhalt != null)
                ergebnis.append(inhalt.toString()+" ");
            else
                ergebnis.append("null ");
        }
        ergebnis.append(">");
        return new String(ergebnis);
    }
}
```

(c) Schreiben Sie eine Klasse VectorSchlange, die die Schnittstelle Schlange implementiert und sich dabei abstützt auf die Klasse Vector des Packets java.util mit den Methoden
void *addElement(Object),*
*Object remove(**int**),*
int *size(),*
*Object elementAt(**int**) und **boolean** isEmpty().:*

```
import java.util.*;
public class VectorSchlange implements Schlange
{
    Vector schlange;
    public VectorSchlange()
    {
        super();
        schlange =new Vector();
    }
    //Fügt ein von null verschiedenes Objekt am Ende der Schlange ein
    public void einfuegen(Object obj)throws IllegalArgumentException
    {       schlange.addElement(obj);
    }
    // Entfernt das Objekt am Anfang der Schlange
    // Liefert null, wenn die Schlange leer ist
    public Object entfernen()
    {       return(schlange.remove(0));
    }
    // Liefert true, falls Schlange leer, false sonst
    public boolean istLeer()
    {       return(schlange.isEmpty());
    }
    // Liefert Länge der Schlange
    public int gibLaenge()
    {       return schlange.size();
    }
    // Darstellung als String: beginnt mit "<",
    // gefolgt von einer String-Darstellung fuer jedes Element,
    // jede getrennt durch Leerraeume, endet mit ">"
```

```
    public String toString()
    {       StringBuffer ergebnis = new StringBuffer("<");
        Object inhalt;
        for(int i = 0;i<schlange.size();i++)
        {
            inhalt = schlange.elementAt(i);
            if(inhalt != null)
                ergebnis.append(inhalt.toString()+" ");
            else
                ergebnis.append("null ");
        }
        ergebnis.append(">");
        return new String(ergebnis);
    }
}
```

(d) Wie können Sie Ihre Implementierungen "thread-sicher" machen, d.h. den nebenläufigen Zugriff auf einen Stapel durch mehrere Threads. Beachten Sie bei der Antwort zur Klasse VectorSchlange den folgenden Satz der API-Dokumentation der Klasse java.util.Vector: „Vector is synchronized".

Gegeben sei die folgende Schnittstelle für einen Stapel(Stack):

```
interface Stapel
{
   //legt elem auf dem Stapel ab
   public void push(Object elem);

   //liefert das oberste Element des Stapels und entfernt es
   public Object pop() throws java.util.EmptyStackException;

   //gibt an, ob der Stapel leer ist oder nicht
   public boolean isLeer();

   //liefert das oberste Element des Stapels, ohne es zu entfernen
   public Object peek();
}
```

(a) **Definieren Sie eine Klasse StapelImp, die diese Schnittstelle implementiert. Die Klasse soll einen parameterlosen Konstruktor besitzen. Die Klasse java.util.EmptyStackException soll als vordefiniert angesehen werden. Hinweis: Besonders einfach wird es, wenn Sie die Klasse StapelImp als .Erweiterung der Klasse java.util.Vector anlegen.**

```
import java.util.Vector;
public class StapelImp extends java.util.Vector implements Stapel{
   private Vector stapel;
   private int naechsterIndex;
   public StapelImp()
   {
      stapel =new Vector();
      naechsterIndex =0;
   }
   //legt elem auf dem Stapel ab
   public void push(Object elem)
   {
      stapel.ensureCapacity(stapel.size() +1);//Anpassen der Größe des
Vector
      stapel.add(naechsterIndex, elem);
      naechsterIndex++;
   }
   //liefert das oberste Element des Stapels und entfernt es
   public Object pop() throws java.util.EmptyStackException
   {
      if (isLeer())
         throw new java.util.EmptyStackException();
      else
      {
         int i =stapel.lastIndexOf(stapel.lastElement());
         return stapel.remove(i);
      }
   }

   //gibt an, ob der Stapel leer ist oder nicht
   public boolean isLeer()
   {
      return(stapel.isEmpty());
   }

   //liefert das oberste Element des Stapels, ohne es zu entfernen
   public Object peek()
   {
      if(isLeer())
         return null;
      return stapel.lastElement();
   }
```

(b) Schreiben Sie ein Testprogramm, dass die Zeichenketten „unten“, „mitte“, „oben“ ablegt und wieder entfernt. Weiterhin soll das Programm prüfen, ob die Ausnahmen java.util.EmptyStackException tatsächlich auftreten!

```
public static void main(String[] args)
{
   try
   {
      StapelImp stapel = new StapelImp();

      //Die Objekte einzeln auf denn Stack legen
      stapel.push((Object) "unten");
      stapel.push((Object) "mitte");
      stapel.push((Object) "oben");

      //Ausgeben des Inhalts und löschen des Eintrags
      while(!stapel.isLeer())
      {
         System.out.println(stapel.pop());
      }
   }
   catch(java.util.EmptyStackException ese)
   {
      ese.getMessage();
      System.err.println("Der Stack ist Leer");
   }
   catch(ArrayIndexOutOfBoundsException aiobe)
   {
      aiobe.getMessage();
      System.err.println("Index out of bound");
   }
}
```

(b)Wie können Sie Ihre Implementierung „thread-sicher“ machen, d.h. den nebenläufigen Zugriff auf einen Stapel durch mehrere Threads so einschränken, dass keine Wettbewerbssituationen(race conditons) eintreten können ?

Einfach nach public synchronized einfügen, z.b. public synchronized Object pop() throws java.util.EmptyStackException. Für alle Methoden der Klasse StapelImp