Prüfungsklausur zur Vorlesung

Programmierung von Systemen

im Sommersemester 2020

Institut für Softwaretechnik und Programmiersprachen
Fakultät für Ingenieurwissenschaften und Informatik
Universität Ulm

Prof. Dr. Matthias Tichy, Stefan Götz

M U S T E R L Ö S U N G

10. August 2020

(1. Prüfungszeitraum)

Aufgabe 1 - OOP (8 + 4 + 3 + 3 = 18 Punkte)

Gegeben sei der Java-Code, welchen Sie auf dem beiliegenden Extrablatt finden können.

a) Implementieren sie die Methoden `transferComputingUnits(int number, Cluster to)` und `transferTo(Cluster to)` in den Klassen `Cluster` und `ComputingUnit`. Die Methode `transferComputingUnits` solle die ersten `number` ComputingUnits aus dem aktuellen Cluster entfernen und dem übergebenen Cluster anhängen. Dabei sollen auch die belongsTo Referenzen via der Methode `transferTo` geändert werden. Sollten im aktuellen Cluster nicht genug ComputingUnits vorhanden sein sollen keine übertragen werden und `false` zurückgegeben werden, ansonsten true. Beachten Sie dabei, dass das manipulieren einer Collection während darüber iteriert wird undefiniertes Verhalten aufweist und desshalb vermieden werden muss!

```
public abstract class Cluster {
  private final LinkedList<ComputingUnit> cus = new LinkedList<>();

  public boolean transferComputingUnits(int number, Cluster to) {
    if (cus.size() >= number) {
        var i = 0;
        for (Iterator<ComputingUnit> iterator = cus.iterator()
            ; iterator.hasNext() && i < number;i++) {
            var cuToTransfer = iterator.next();
            cuToTransfer.transferTo(to);
            iterator.remove();
        }
        return true;
    } else {
        return false;
    }
  }
}

public class ComputingUnit {
    private Cluster belongsTo;
    private final int computingPower;

    @Override
    public void transferTo(Cluster to) {
        belongsTo = to;
    }
}
```

- 1.5P number check
- 2P Loop
- 1.5P transferTo call
- 0.5P löschen
- 2P 2. Loop oder andere sinnvolle Lösung
- 0.5P transferTo Implementierung

b) Welche Ergänzungen müssen an der Klasse `Computer` vorgenommen werden um sicher zu stellen, dass nur maximal eine Instanz der Klasse erzeugt werden kann. Ergänzen Sie hierzu den folgenden Codeabschnitt:

```
public class Computer extends Cluster {

    private static Computer singleton;

    private Computer() {}

    public static Computer getSingleton() {
        if (singleton == null) {
            singleton = new Computer();
        }
        return singleton;
    }
}
```

- 0.5P static Computer attribute
- 1P private Konstruktor
- 1P singleton-getter mit return
- 1P getter static
- 0.5P null check und Konstruktor aufruf

c) Welche der folgenden Codestücke sind korrekt und welche führen zu **Compile**- oder **Laufzeitfehlern**? Begründen Sie, warum ein Codestück nicht korrekt ist. Sie können davon ausgehen, dass die Aufrufe aus einer `main`-Methode in einem anderen *Package* gemacht werden welches die nötigen imports vorweist.

```
var computer = new Computer();
System.out.println(computer.transferComputingUnits(100, new Cluster()));
```

Falsch, weil `Cluster` eine Abstrakte Klasse ist und somit nicht erzeugt werden kann.

0.5P falsch, 0.5P abstrakt

```
Computer computer = new Computer();
Cluster cluster   = new Computer();
var cu = new ComputingUnit(computer, 100);
```

Richtig, `Computer` ist eine Subklasse von Cluster und somit kann ein Computer ein Cluster sein.

0.5P richtig, 0.5P Vererbung

```
ComputingUnit cu = new ComputingUnit();
var computer = new Computer();
cu.transferTo(computer);
```

Falsch, Der parameterlose Konstruktor von ComputingUnit existiert nicht.

0.5P falsch, 0.5P Konstruktor fehlt

d) Der folgende Codeabschnitt braucht sehr lange um zu überprüfen ob die Variable `myCollection` eine 42 enthält oder nicht. Nennen Sie eine andere Datenstruktur aus der CollectionsAPI die die Zeit zum Überprüfen ob 42 enthalten ist drastisch reduzieren kann. Erklären Sie außerdem warum dies der Fall ist und welche weiteren Auswirkung die Verwendung der anderen Datenstruktur haben wird.

```
public class CollectionsAdding {
    public static void main(String[] args) {
        var myCollection = new LinkedList<Integer>();
        var random = new Random();
        for (int i = 0; i<1000000; i++) {
            myCollection.add(random.nextInt());
        }
        var containsFourtyTwo = myCollection.contains(42);
        System.out.println(containsFourtyTwo);
    }
}
```

1P HashSet/Map o.Ä. Andere interne Struktur der Daten führt zu schnellerem Zugriff auf Elemente und zu schnellerem Check (1P). Dafür braucht aber das adden der Elemente jeweils länger (1P).

Aufgabe 2 - JavaFX (1 + 1.5 + 1 + 4.5 = 8 Punkte)

Gegeben Sei der folgende Scenegraph.:

```
<BorderPane prefHeight="500.0" prefWidth="700.0"
 xmlns="http://javafx.com/javafx/8.0.111" xmlns:fx="http://javafx.com/fxml/1">
  <center>
    <BorderPane>
      <top>
        <ToolBar>
          <items>
            <Button background-color="blue">
              <graphic>
                <ImageView pickOnBounds="true" preserveRatio="true">
                  <image>
                    <Image url="@/open.png" />
                  </image>
                </ImageView>
              </graphic>
            </Button>
          </items>
        </ToolBar>
      </top>
      <center>
        <SplitPane orientation="VERTICAL">
          <items>
            <AnchorPane minHeight="0.0" minWidth="0.0">
              <children>
                <Label text="no file loaded..."/>
                <ScrollPane fx:id="imageScrollPane">
                  <content>
                    <ImageView fx:id="imageView"/>
                  </content>
                </ScrollPane>
              </children>
            </AnchorPane>
          </items>
        </SplitPane>
      </center>
    </BorderPane>
  </center>
</BorderPane>
```

a) Schreiben Sie einen CSS Ausdruck um die Hintergrundfarbe aller Buttons rot zu färben.

```
1 .button {
2   -fx-background-color: red;
3 }
```

- 0.5P .button
- 0.5P bg-c: red (0P falls fx- fehlt)

b) Schreiben Sie einen CSS Ausdruck um den ImageView mit der Id *imageView* zu finden

```
1 ImageView[fx-id="imageView"] {}
```

- 0.5P ImageView
- 1P fx-id=‘iv‘ (0.5P falls fx- fehlt)

c) Schreiben Sie einen CSS Ausdruck um alle ScrollPanes innerhalb der AnchorPane zu finden

```
1 AnchorPane ScrollPane {}
```

- 1P ScrollPane hinter AnchorPane

d) Welchen Code müssen Sie schreiben, damit beim Mausklick eines Buttons auf dem Button angezeigt wird ob dieser mit dem linken (primären) Maus-Button gedrückt wurde oder nicht?

Hinweis:
Verwenden Sie setzen des Textes die Methode `void setText(String text)` der Klasse `Button`.

```
button.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        button.setText("" + event.isPrimaryButtonDown());
    }
});
```

- 1P setOnMouseClicked
- 1P new EventHandler<MouseEvent>
- 1P handle-Methode mit MouseEvent parameter (0.5P falls falscher parameter typ)
- 0.5P setText
- 1P event.isPrimaryButtonDown()

Aufgabe 3 - Java IO (7 + 4 = 11 Punkte)

a) Schreiben Sie eine Methode die die Inhalte von allen übergebenen Channels abwechselnd byteweise in die Datei **out.data** schreibt.

Achten Sie auf Behandlung der folgenden Exceptions mit einer jeweils passenden Meldung:

- `IOException`

Beachten Sie hierfür auch den JavaIO Teil des **CheatSheets** am Ende der Klausur.

Hinweis: Verwenden Sie die Methode `long getMaxSize(FileChannel[] ics)` um die Länge des längsten Channels zu bestimmen.

```
public static void combineBytes(FileChannel[] ics) {
    try {
        FileOutputStream fos = new FileOutputStream(new File("out.data"));

        var targetChannel = fos.getChannel();

        for (long i = 0; i < getMaxSize(ics); i++) {
            for (FileChannel inputChannel : ics) {
                if (inputChannel.size() >= i) {
                    inputChannel.transferTo(i, 1, targetChannel);
                    // close input
                    inputChannel.close();
                }
            }
        }
        // close target
        targetChannel.close();
        fos.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

- 1P FileOutputStream erzeugen
- 1P Channel aus outputStream generiern
- 0.5P maxSize Loop
- 0.5P Loop über ics
- 1P channel size check
- 1P transfer
- 0.5P inputChannel close
- 0.5P targetChannel close
- 0.5P fos close; 0.5P zusätzlich für IOException tryCatch

b) Java bietet eine Vielzahl an Streams für die Verarbeitung von Daten an. Darunter auch gepufferte und un-gepufferte Varianten. Erläutern Sie anhand von sinnvoll gewählter Beispiele die Vor-und Nachteile von **zwei** Streams. Beachten Sie hierfür auch den Dateiein- und -Ausgabe Teil des **CheatSheets** am Ende der Klausur.

z.B BufferedOutputStream vs OutputStream vs ObjectOutputStream Buffered kann bei großen Daten Mengen schneller abgearbeitet werden weil vorgepuffert werden kann. Rohdaten auf die an beliebigen Stellen zugegriffen werden soll ist man mit Output/InputStream besser bedient da man hier random access hat. ObjectStreams können mit serializierbaren Objekten direkt umgehen und man muss die Objekte nicht gestückelt lesen/schreiben.

- 0.5P Pro Nennung
- 1.5P mit sinnvoller Erklärung, Beispiel und Abwägung

Aufgabe 4 - XML (2 + 8 + 2 = 12 Punkte)

Betrachten Sie das folgende XML-Dokument:

```
1  <xs:schema attributeFormDefault="unqualified"
2   elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="ComputingUnit">
4      <xs:complexType>
5        <xs:simpleContent>
6          <xs:extension base="xs:string">
7            <xs:attribute type="xs:byte" name="power" use="optional"/>
8          </xs:extension>
9        </xs:simpleContent>
10     </xs:complexType>
11   </xs:element>
12   <xs:element name="Location" type="xs:string"/>
13   </xs:element name="Computer">
14     <xs:complexType/>
15       <xs:sequence>
16         <xs:element ref="ComputingUnit" maxOccurs="unbounded" minOccurs="0"/>
17       </xs:Sequence>
18       <xs:attribute type="xs:byte" name=id/>
19     </xs:complexType>
20   </xs:element>
21   <xs:element name="Cluster">
22     <xs:complexType mixed="true">
23       <xs:sequence>
24         <xs:element ref="Location" minOccurs="1" maxOccurs="1">
25         <xs:element ref="Computer" minOccurs="0"/>
26         <xs:element ref="Cluster" minOccurs="0"/>
27       </xs:sequence>
28     </xs:complexType>
29   </xs:element>
30 </xs:schema>
```

a) Das XML Dokument ist nicht wohlgeformt. Geben Sie die Zeilennummern von **vier** der **fünf** vorkommenden Fehler an und erklären Sie wie diese korrigiert werden können. Konformität mit dem Schema für XSDs muss hierbei nicht berücksichtigt werden.

- *jeweils 0.5P*
- *Z13 schließender Tag sollte öffnender sein*
- *Z14 falsch selfclosing*
- *Z17 Groß-/Kleinschreibung mit Z15*
- *Z18 fehlende Anführungszeichen*
- *Z24 fehlender schließender Tag*

b) Die obige XML-Datei beschreibt ein XML-Schema. Schreiben Sie eine DTD-Datei so, dass diese die selben Dokumente validiert. Markieren Sie die Stellen an denen dies nicht möglich ist.

```
<!ELEMENT Cluster (Location,(Computer|Cluster)*)>
<!ELEMENT Location (#PCDATA)>
<!ELEMENT Computer (ComputingUnit*)>
<!ELEMENT ComputingUnit>

<!ATTLIST Computer id CDATA #REQUIRED> <-- typ kann nicht spezifiziert werden
<!ATTLIST ComputingUnit power CDATA #IMPLIED> <-- ditto
```

- 0.5P Cluster mit Contents
- 1P korrekter Inhalte von Cluster
- 1P Location
- 1P Computer mit korrektem Inhalt
- 1P Computing Unit
- 0.5P Computer id Attribute
- 0.5P Computer id REQUIRED
- 0.5P ComputingUnit power Attribute
- 0.5P ComputingUnit power IMPLIED
- 1.5P erkannt, dass typen nicht gleich spezifiziert werden können

c) Erläuteren Sie welchen XML-Parser Ansatz Sie verwenden würden um eine XML-Datei die dem obigen Schema entspricht in ein passendes JSON-Format zu transformieren.

z.B SAX-Parser da von oben nach unten alles abgearbeitet werden muss und zu jedem Element ein passendes JSON-Objekt mit Attributen etc. erstellt werden muss.
Mit sinnvoller Erklärung wäre auch DOM-Parser ok.

- 0.5P Parser-Ansatz Nennung
- 1.5P sinnvolle Begründung

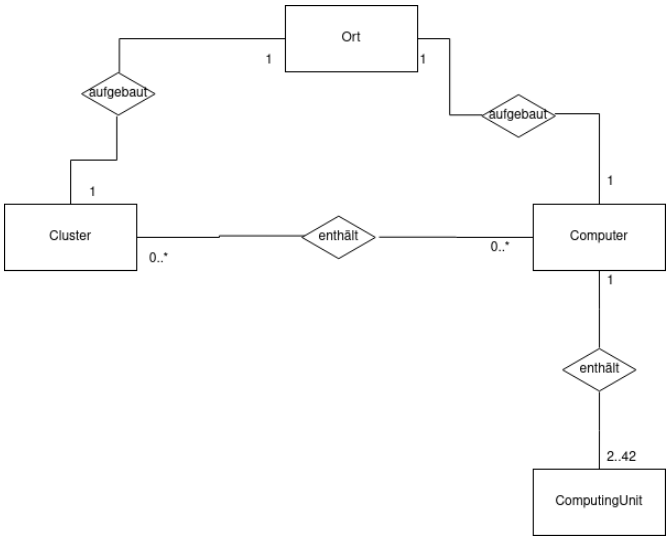
Aufgabe 5 - ER-Modellierung (1.5 + 6 + 3.5 = 11 Punkte)

Ein Rechencluster besteht aus einer beliebigen Menge an Computern. Sowohl Computer als auch Rechencluster sind an einem bestimmten Ort aufgebaut. Computer können mehr als einem Cluster zugeordnet werden dafür müssen aber die Aufbauorte des Computers und der Cluster übereinstimmen. Damit Computer rechnen können werden mindestens 2 und maximal 42 ComputingUnits darin installiert welche nicht zwischen Computern geteilt werden können.

a) Welche Anforderung aus dem obigen Sachverhalt können Sie nicht im **E-R-Diagramm** oder dem **relationalen Datenbankschema** modellieren und wo könnte diese stattdessen realisiert werden?

Gleicher Ort für Computer und Cluster (1P). Könnte später in Code realisiert werden der die DB anspricht/ verwaltet (0.5P).

b) Modellieren Sie den beschriebenen Sachverhalt als **E-R-Diagramm**.



- 0.5P pro Entity (Ort als Attribut für Cluster UND Computer ist auch ok gibt dann 0.5P wenn in beiden Vorhanden)
- 1P pro Beziehung (0.5P falls kardinalitäten falsch)
- WICHTIG Notation der Kardinalitäten ist egal SOLANGE sie einheitlich ist!

c) Erstellen Sie zu dem E-R-Diagramm das dazugehörige **relationale Datenbankschema**. Achten Sie hierbei auf sinnvoll gewählte **Primärschlüssel** und stellen Sie sicher, dass das Schema mindestens in 3. Normalform ist. Sie können bei Bedarf auch IDs einführen, um die Eindeutigkeit eines Primärschlüssels zu gewährleisten.

Cluster(<u>CID</u>, *PLZ*)
Computer(<u>CPID</u>, *PLZ*)
CC(<u>*CID*</u>,<u>*CPID*</u>)
ComputingUnit(<u>CUID</u>, *CPID*)
Ort(<u>PLZ</u>)

- 0.5P pro korrektes Schema Cluster, Computer, ComputingUnit, Ort
- 0.5P für FK Beziehung von Cluster und Computer zu Ort
- 1P korrektes Schema CC (0.5P falls PK/FK Beziehung zu C/CP falsch)

Aufgabe 6 - SQL (1 + 3 + 2 + 5 + 4 = 15 Punkte)

Gegeben seien die folgenden Relationenschemata (auch zu finden auf dem beiliegenden Extrablatt):

Schiffe		
SID	Name	**MID**

Rennen			
RID	RennName	Startzeit	Preisgeld

Schiffsrennen	
SID	**RID**

Matrosen	
MID	MName

Besatzung	
MID	**SID**

Primärschlüssel sind unterstrichen, Fremdschlüssel sind **fett** dargestellt. Formulieren Sie, insofern nicht anders spezifiziert, die **SQL-Statements** zur Lösung folgender Teilaufgaben:

a) Setzen Sie das Preisgeld des Rennens 'Segl101' auf 1000.

```
UPDATE Rennen
SET Preisgeld = 1000
WHERE Name = 'Segl101'
```

- 0.5P UPDATE, SET
- 0.5P WHERE

b) Geben Sie an, wieviele Schiffe existieren bei denen 'Titanic' im Namen vorkommt.

```
SELECT COUNT(*)
FROM Schiffe
WHERE Name LIKE '%Titanic%'
```

- 1P COUNT
- 0.5P FROM
- 1.5P WHERE (0.5P davon sind die % und weitere 0.5P das LIKE statt =)

c) Übersetzen Sie den folgenden Relationenalgebra-Ausdruck nach SQL.
$\pi_{Preisgeld}(\sigma_{(Name=Blitzkugel)}(Rennen \bowtie Schiffsrennen \bowtie Schiffe))$

```
SELECT Preisgeld
FROM Rennen NATURAL JOIN Schiffsrennen NATURAL JOIN Schiffe
WHERE Name = 'Blitzkugel'
```

- 0.5P SELECT
- 1P FROM (0.5P falls joins falsch)
- 0.5P WHERE

d) Geben Sie die Namen aller Schiffe aus, die mehr als 3 Besatzungsmitglieder haben. Besatzungen werden mittels der Tabelle Besatzungen die zwischen die Schiffe und Matrosen Tabelle gestellt ist definiert. *Hiweis: Besitzer sind als MID in den Schiffen vermerkt und wird nicht als verbindende ID zwischen Schiffe und Besatzung verwendet!*

```
SELECT Name
FROM Schiffe as S JOIN Besatzung as B ON S.SID = B.SID
GROUP BY Name
HAVING COUNT(*) > 3
```

- 0.5P SELECT
- 2P FROM (1P falls join falsch, WICHTIG Natural Join geht hier nicht!)
- 1P GROUP BY
- 1.5P HAVING (0.5P falls bedingung falsch)

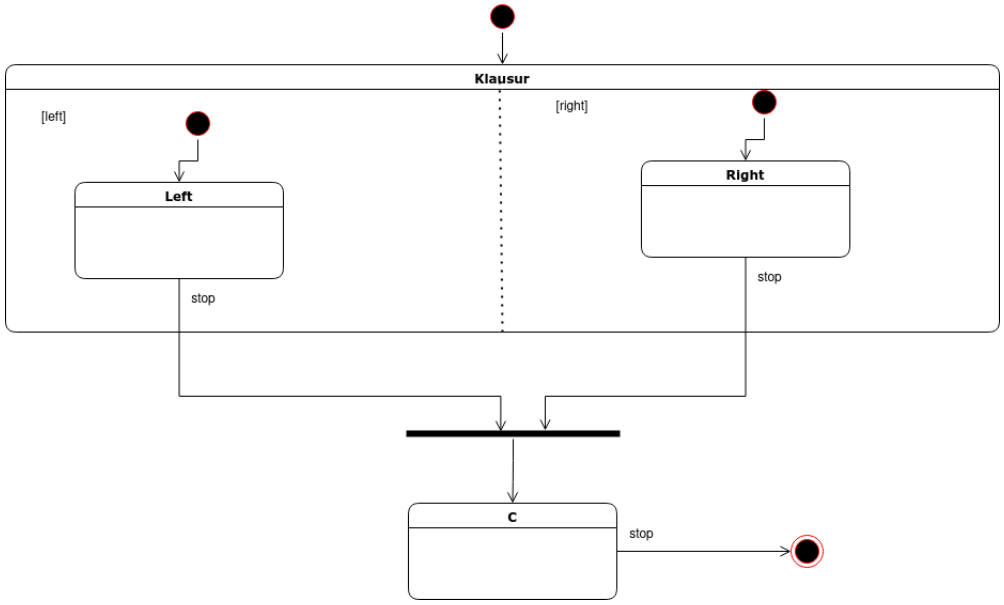
e) Geben Sie die Namen aller der Schiffe an, deren Besitzer nicht Teil der Besatzung ist. *Hiweis: Besitzer sind als MID in den Schiffen vermerkt!*

```
SELECT Name
FROM Schiffe AS S
WHERE S.MID NOT IN (
  SELECT MID
  FROM Schiffe AS SS JOIN Besatzung AS B ON SS.SID = B.SID
  WHERE S.SID = SS.SID
)
```

- 0.5P SELECT
- 0.5P FROM
- 2P Subquery (0.5P SELECT; 1P FROM!KEIN NJOIN MÖGLICH; 0.5P WHERE)
- 1P WHERE NOT IN

Aufgabe 7 - Threads (4 + 3 + 3 = 10 Punkte)

a) Betrachten Sie den folgenden Zustandsautomaten:



Ergänzen Sie den folgenden Code so, dass er den im Zustandsautomaten abgebildeten Ablauf darstellt. Sie können hierbei davon ausgehen, dass die Klassen `Left`, `Right` und `C` die internen Abläufe der linken bzw. rechten Seite des Klausur Zustandes und des C Zustandes abbilden. Um deren Implementierung muss sich nicht gekümmert werden. Beachten Sie, dass die drei Klassen alle von `Thread` erben.

```
1  public class StateManager {
2      public static void main(String... args) throws Exception{
3          Thread left  = new Left();
4          Thread right = new Right();
5          Thread c     = new C();
6          //1P starten der threads
7          left.start();
8          right.start();
9          //1P warten bis beide durch sind via join
10         left.join();
11         right.join();
12         //1P l und r beide gestartet bevor einer der beiden gejoint wird
13         //0.5P starten + warten auf C thread; 0.5P starten von c NACH join von l u
14         c.start();
15         c.join();
16         System.out.println("end");
17     }
18 }
```

b) Gegeben Sei der folgende Codeabschnitt welcher nicht immer die erwarteten 10000 ausgibt. Erklären Sie, mit Nennung der relevanten Zeilen warum dies nicht der Fall ist und Skizzieren Sie, wieder mit Nennung der relevanten Zeilen, zwei Arten um den Ablauf des Programmes Threadsicher zu machen.
Hinweis: Mehrfachnennungen des selben Prinzips sind nicht erlaubt.

```
1  public class MyThreads extends Thread {
2      public static int counter = 0;
3
4      public static void main(String... strings) throws Exception {
5          var myThreads = new LinkedList<Thread>();
6          for (int i = 0; i < 100; i++) {
7              var mt = new Thread();
8              mt.start();
9          }
10         System.out.println("counter = " + counter);
11     }
12
13     @Override
14     public void run() {
15         for (int i = 0; i < 100; i++) {
16             counter++;
17         }
18     }
19 }
```

run ist nicht synchronized (1P). z.B: Statt einen Int als counter zu nehmen einen Atomic Integer nehmen und dann statt counter++ mit getAndAdd() oder addAndGet() arbeiten (1P). synchronized Block/run-Methode verwenden (1P).

c) Erläutern Sie warum es als schwierig erachtet wird Deadlocks in Code auszuschließen. Gehen Sie hierbei auch darauf ein wieso Deadlocks überhaupt entstehen.

Da Deadlocks entstehen wenn sich 2 Threads gegenseitig die Ressourcen blockieren die benötigt werden (1.5P) und es in komplexerer Software schwierig ist alle Ressourcen die alle Threads benötigen im Überblick zu behalten (1.5P).

Aufgabe 1 - OOP

```
1
2  public class ComputingUnit implements Transferable {
3  private Cluster belongsTo;
4  private final int computingPower;
5
6  public ComputingUnit(Cluster parent, int power) {
7  belongsTo = parent;
8  computingPower = power;
9  }
10
11 @Override
12 public void transferTo(Cluster to) {
13 // TODO Auto-generated method stub
14 }
15 }
16
17 public abstract class Cluster {
18 private final LinkedList<ComputingUnit> cus = new LinkedList<>();
19
20 public boolean transferComputingUnits(int number, Cluster to) {
21 // TODO Auto-generated method stub
22 return false;
23 }
24
25 }
26
27 public interface Transferable {
28 public void transferTo(Cluster to);
29 }
30
31 public class Computer extends Cluster {
32
33 }
```

Aufgabe 6 - SQL

Schiffe		
SID	Name	**MID**

Rennen			
RID	RennName	Startzeit	Preisgeld

Schiffsrennen	
SID	**RID**

Matrosen	
MID	MName

Besatzung	
MID	**SID**

JavaIO

FileOutputStream

Modifier and Type	Method and Description
void	**close**() Closes this file output stream and releases any system resources associated with this stream.
protected void	**finalize**() Cleans up the connection to the file, and ensures that the `close` method of this file output stream is called when there are no more references to this stream.
FileChannel	**getChannel**() Returns the unique **FileChannel** object associated with this file output stream.
FileDescriptor	**getFD**() Returns the file descriptor associated with this stream.
void	**write**(byte[] b) Writes `b.length` bytes from the specified byte array to this file output stream.
void	**write**(byte[] b, int off, int len) Writes `len` bytes from the specified byte array starting at offset `off` to this file output stream.
void	**write**(int b) Writes the specified byte to this file output stream.

JavaIO

FileChannel

Modifier and Type	Method and Description
abstract **MappedByteBuffer**	**map**(**FileChannel.MapMode** mode, long position, long size) Maps a region of this channel's file directly into memory.
static **FileChannel**	**open**(**Path** path, **OpenOption**... options) Opens or creates a file, returning a file channel to access the file.
static **FileChannel**	**open**(**Path** path, **Set**<? extends **OpenOption**> options, **FileAttribute**<?>... attrs) Opens or creates a file, returning a file channel to access the file.
abstract long	**position**() Returns this channel's file position.
abstract **FileChannel**	**position**(long newPosition) Sets this channel's file position.
abstract int	**read**(**ByteBuffer** dst) Reads a sequence of bytes from this channel into the given buffer.
long	**read**(**ByteBuffer**[] dsts) Reads a sequence of bytes from this channel into the given buffers.
abstract long	**read**(**ByteBuffer**[] dsts, int offset, int length) Reads a sequence of bytes from this channel into a subsequence of the given buffers.
abstract int	**read**(**ByteBuffer** dst, long position) Reads a sequence of bytes from this channel into the given buffer, starting at the given file position.
abstract long	**size**() Returns the current size of this channel's file.
abstract long	**transferFrom**(**ReadableByteChannel** src, long position, long count) Transfers bytes into this channel's file from the given readable byte channel.
abstract long	**transferTo**(long position, long count, **WritableByteChannel** target) Transfers bytes from this channel's file to the given writable byte channel.
abstract **FileChannel**	**truncate**(long size) Truncates this channel's file to the given size.
FileLock	**tryLock**() Attempts to acquire an exclusive lock on this channel's file.
abstract **FileLock**	**tryLock**(long position, long size, boolean shared) Attempts to acquire a lock on the given region of this channel's file.
abstract int	**write**(**ByteBuffer** src) Writes a sequence of bytes to this channel from the given buffer.
long	**write**(**ByteBuffer**[] srcs) Writes a sequence of bytes to this channel from the given buffers.
abstract long	**write**(**ByteBuffer**[] srcs, int offset, int length) Writes a sequence of bytes to this channel from a subsequence of the given buffers.
abstract int	**write**(**ByteBuffer** src, long position) Writes a sequence of bytes to this channel from the given buffer, starting at the given file position.

Dateiein- und -ausgabe