Technische Hochschule Ingolstadt

Ausgangssituation

Als Mitarbeiter eines Projektteams werden Sie mit der Entwicklung einer Online-Plattform für die Vermietung von Fahrzeugen beauftragt.

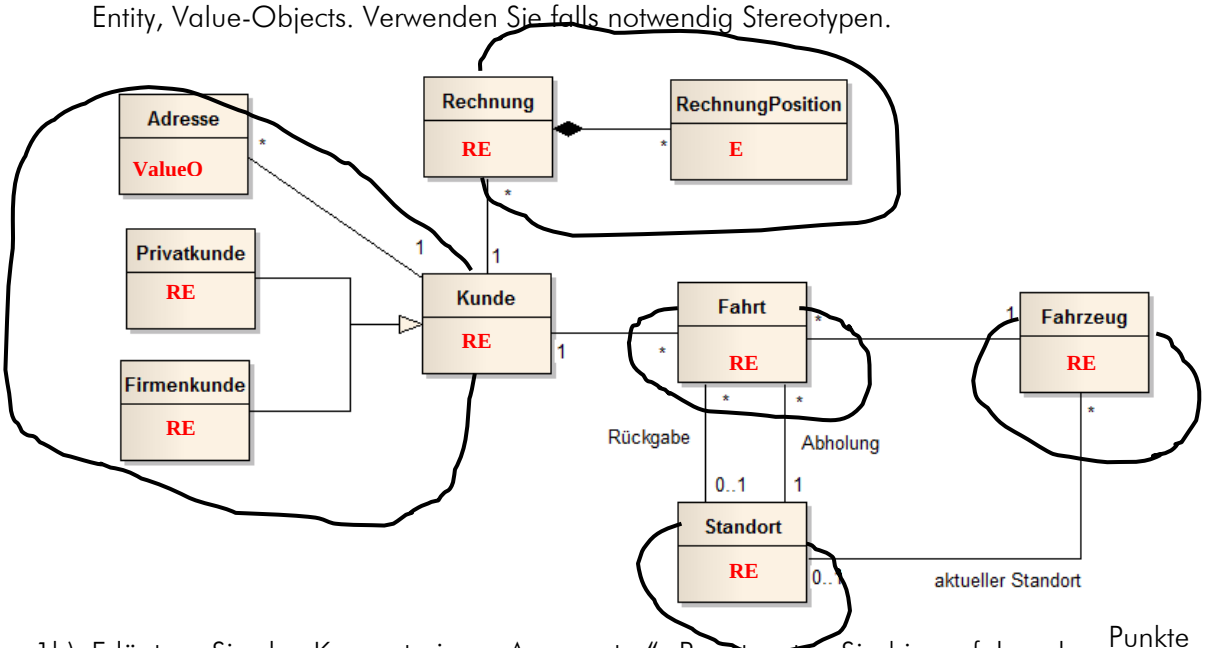
Produktvision:

- Ein Kunde soll über das System Fahrzeuge reservieren können.
- Bei einer Reservierung werden der gewünschte Fahrzeugtyp sowie der Ort der Abholung angegeben.
- Sind Fahrzeuge des gewünschten Typs nicht verfügbar, kann das System automatisch ein höherwertiges Fahrzeug zur Vermietung anbieten.
- Ein Fahrzeug wird an einem Standort abgeholt und zurückgegeben.
- Nach der Rückgabe des Fahrzeuges wird der Fahrzeugzustand überprüft und evtl. notwendige Reparaturmaßnahmen werden durchgeführt.
- Das System soll einen hohen Sicherheitsstandard erfüllen. Jeder Benutzer, der mit dem System arbeitet, unterliegt bestimmten Restriktionen, welche durch ein Berechtigungssystem definiert werden.
- Damit ein Benutzer eine Funktion ausführen kann, muss er sich am System anmelden.
- Das System soll sowohl über einen Web-Browser als auch für gängige Smartphones angeboten werden.
- Bei der Umsetzung ist auf eine möglichst hohe Wiederverwendbarkeit zu achten.

 Erzielte Punkte:_____

1. Domain Driven Design

1a) Im Rahmen des Designs wenden Sie nun die Methode „Domain Driven Design“ auf das nachstehende Produktmodell an. Analysieren Sie das nachfolgende Diagramm und kennzeichnen Sie alle Klassen in diesem Diagramm. Stellen Sie folgende Kernelemente innerhalb des Diagramms heraus: Aggregate, Entity, Root-Entity, Value-Objects. Verwenden Sie falls notwendig Stereotypen. Punkte ___



1b) Erläutern Sie das Konzept eines „Aggregates“. Beantworten Sie hierzu folgende Aspekte: Was versteht man unter einem Aggregat? Was ist die sog. Root-Entity? Punkte ___

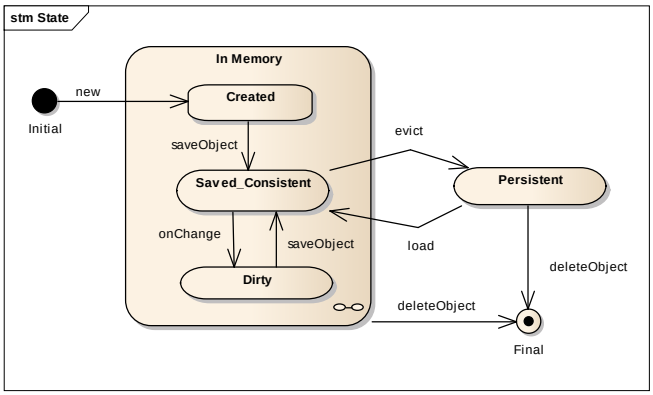
- Zusammenschluss von Objekten die einen gemeinsamen Lebenszyklus habe
- Alle Objekte innerhalb des Aggregates werde als Ganzes geladen und gespeichert
- Referenzen dürfen nur of die Root-Entity, den Einstiegspunkt in das Aggregat zeigen
- Innerhalb des Aggregates wird über Traversierung navigiert.

1c) Definieren Sie das Konzept einer Enity. Was ist eine Entity. Was ist eine globale bzw. lokale Identität? Geben sie jeweils ein Beispiel für die Identität an. Punkte ___

- Ein Objekt welches nicht ausschließlich durch seine Attribute beschrieben wird sondern primär durch seine Identität wird Entity genannt
- locale Identität welche nur innerhalb des Aggregats eindeutig sein muss (Position einer Rechnungsposition)
- Globale Identiät muss global eindeutig sein. (Die rechnungsnr selbst)

Erzielte Punkte:_____

1d) Erläutern Sie das nebenstehende Zustandsdiagramm. Dieses Zustandsdiagramm stellt den Lebenszyklus eines Domänen-Objekts dar. Erläutern Sie die Zustände sowie die zugeordneten Ereignisse. Stellen Sie den Unterschied zwischen einem Objekt einer Programmiersprache und einem Domänenobjekt heraus. Punkte ___



- Domänenobjekte existieren solange eine Repräsentation des Objektes im Hauptspeicher oder auf der Persistenz existiert
- Wir ein Objekt angelegt ist es nur im Speicher vorhanden. Durch das Speichern wird der Zustand des Objekts persistiert.
- Wird ein Objekt im Hauptspeicher verändert so ist der DB-Zustand inkonsistent. Erst ein erneutes speichern sorgt dafür dass die Zustände konsistent sind.
- Ein Programmiersprachenobjekt ist nur eine von vielen möglichen Repräsentationen.

1e) Erläutern Sie die Aussage „For every traversable association in the model, there is a mechanism in the software with the same properties“. Welche Auswirkung hat diese Aussage auf die Implementierung von Assoziationen? Wie wirkt diese Aussage im Kontext von Aggregaten? Punkte ___

- Besitzt eine Domänenmodell navigierbare Assoziationen so sind in der Software effiziente Mechanismen zu implementieren welche einen einfachen, effizienten Zugriff auf die assoziierten Objekte zulassen
- Es wird nicht vorausgesetzt dass es sich hierbei immer um Objektreferenzen handeln muss
- Auch der Lookup über die Identität ist ein effizienter Zugriffsmechanismus

2. Software Architektur & Design Prinzipien

2a) Beschreiben Sie die Aufgaben der „Konfiguration" bzw. des „Konfigurationsmanagers". Welches Architekturprinzip wird durch den Konfigurationsmanager realisiert? Geben Sie eine Begründung an. Punkte __

- Hat als zentraler Bestandteil wissen über die Implementierung und Schnittstellen der Komponenten
- Wichtig für die Umsetzung der losen Kopplung da nur die Konfiguration die Implementierung von Komponenten kennt. Dadurch bleiben die Komponenten unabhängig voneinander.
- Der Konfigurationsmanager muss das „Inversion of Control" – Prinzip anwenden um das wissen der Komponenten in Richtung des Containers zu verlagern.

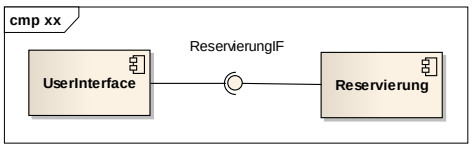
2b) Erläutern Sie die Begriffe „Inversion of Control" und „Dependency Injection". Punkte __

- Inversion of Control: Die Komponenten wissen nichts über Ihre Umgebung. Die Umgebung steuert die Komponenten und deren Beziehungen zueinander

- Die Abhängigkeit zu den importierten Schnittstellen wird durch den Container(=Laufzeitumgebung) verwaltet.

- Wird eine Komponente erzeugt werden die Abhängigkeit von außen durch den Container injiziert.

2c) Erweitern Sie das nachfolgende Programmfragment so, dass die Implementierung der Komponente „UserInterface" das Prinzip „setter-Injection" implementiert. Punkte __




```
public class UserinterfaceImpl  {

 private ReservierungIF reservierung_____________________________________________;

 public UserinterfaceImpl  (ReservierungIF reservierung) {

    this. Reservierung = reservierung;
  }

  .....
}
```

 Erzielte Punkte:_____

2d) Definieren Sie das liskovsche Substitutionsprinzip. Skizzieren Sie ein Beispiel und erläutern Sie daran was passiert wenn gegen das Prinzip verstoßen wird. Punkte __

- Alle beweisbaren Eigenschaften der Oberklasse müssen auch für alle Unterklassen gelten.
- Wird gegen das Prinzip verstoßen ist die polymorphe Verwendung problematisch weil sich Unterklassen anders verhalten als die Oberklassen. Dadurch kann es zu instabilen Programmsystemen kommen.
- Beispiel:

2e) Erläutern Sie das Konzept „Design by Contract". Punkte __

- Ein System wird durch die Spezifikation von Verträgen spezifiziert
- Für eine Klasse existieren folgende Zusicherung: Vorbedingungen, Nachbedingung, Invarianten
- Invariante: Ein Zusicherung welche sich auf die Eigenschaft der Klasse bezieht.
- Der Vertrag einer Methode besteht aus Vor- und Nachbedingung
- Vorbedingung: Der Aufrufer hat die Pflicht die Vorbedingung der aufgerufenen Methode zu erfüllen. Der Aufgerufen geht davon aus dass die Vorbedingung eingehalten wird
- Nachbedingung: Stellt den Zustand eines Objektes nach Aufruf der Methode dar Der Aufgerufene ist Verpflichtet die Nachbedingung sicherzustellen

2f) Begründen Sie warum die Verwendung von globalen Variablen problematisch im Hinblick auf Invarianten ist. Geben Sie ein Beispiel an. Punkte __

Weil die Einhaltung der Invarianten bei der Verwendung von globalen Variablen nicht garantiert werden kann

Beispiel: Person.age Invariante age >= 0. Bei globalen Zugriff kann dies geändert warden und die Invariante ist verletzt

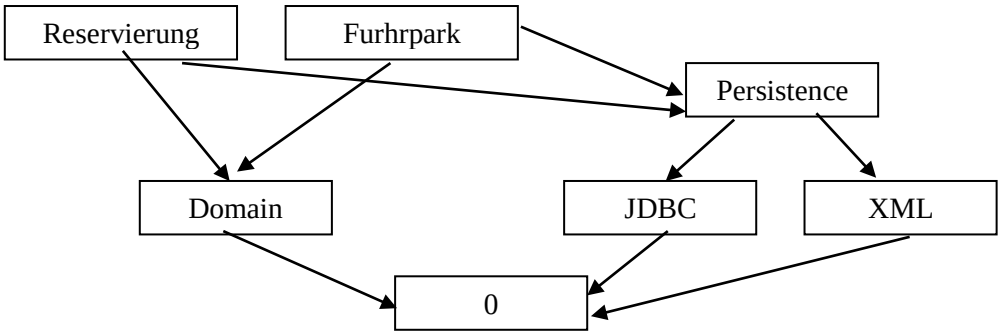
3.) Software Architektur – Die Struktursicht

Im Rahmen einer Architekturbewertung sollen Sie nun Komponenten, sowie deren Schnittstellen bewerten. Hierzu wird ihnen für das Repository des Reservierungssystems folgendes Quellcodefragment vorgelegt (Schnittstellenspezifikation):

```
public interface Repsoitory {
   // Fahrzeug laden
   XMLDoc loadFahrzeug( jdbc.Connection conn, String fahrzeugID );
   // Gericht in der Datenbank speichern
   void saveGericht(jdbc.Connection conn, XMLDocument docGericht );
}
```

3a) Skizzieren Sie den Software-Kategoriegraphen unter der Annahme, dass die Anwendungskomponente „Reservierung" sowie die Anwendungskomponente „Fuhrparkmanagement" auf das Repository zugreifen. Darüber hinaus gelten folgende Annahmen: Punkte ___

- Alle Datenbank spezifischen Klassen/Interfaces (jdbc) liegen in der Kategorie „JDBC".
- Alle XML spezifischen Klassen liegen in der Kategorie XML
- Die Klassen des Produktmodells werden in der Kategorie „A" (Anwendung) angesiedelt



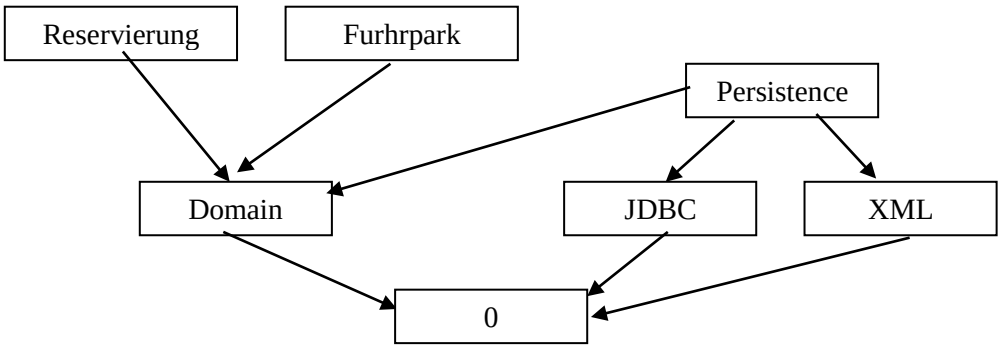
3b) Nennen Sie zwei Probleme, die in dieser Architektur enthalten sind. Sind die Probleme hinderlich für die Flexiblität (Begründung)? Punkte ___

- Persistenz kann nicht ausgetauscht werden
- Anwendungskomponenten benötigen Wissen über die Persistenztechnolog
- Ja weil eine hohe Abhängigkeit zwischen Anwendung und technologie Besteht. Damit eist eine einfache Weiterentwicklung und Austausch nicht möglich (Stabilität)

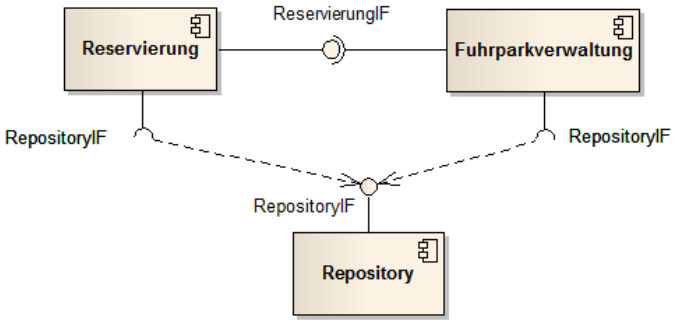
3c) Definieren Sie das Konzept der R-Software. Welche besondere Einschränkung muss bei R-Software gelten. Punkte ___

- Verwende Transformationssoftware um die Daten zwischen verschiedenen Welten zu „transformieren"
- R-Software: Software die nur transformiert aber keine fachlichen Aufgaben übernimmt
- Die Vermengung der Kategorien ist ausschließlich an dieser Stelle erlaubt.
    - R-Software kann generiert werden
    - R-Software kann extern definiert werden

3d) Zeichnen Sie nun einen verbesserten Kategoriegraphen. Begründen Sie warum Ihr Vorschlag die genannten Probleme löst. Punkte ___



3e) Ordnen Sie die Artefakte des nebenstehenden Diagramms sinnvoll den Softwarekategorien aus Aufgabe 3d zu. Punkte ___



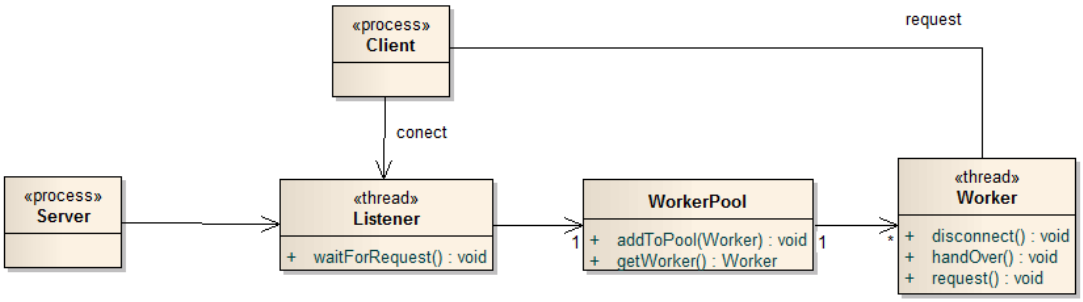
Artefakt	Typ	Kategorie
Implementierung Reservierung	Klasse(n)	Reservierung
Implementierung Fuhrparkverwaltung	Klasse(n)	Fuhrpark
Implementierung Repository	Klasse(n)	Persistence
RepositoryIF	Schnittstelle	Domain
ReservierungIF	Schnittstelle	Domain

3f) Erläutern Sie das Prinzip „Kommunikation mit neutralen Schnittstellen“. Warum hilft dieses Prinzip reine Komponenten zu definieren. Bei welchem Artefakt aus Aufgabe 3d kommt/kann dieses Prinzip zum Einsatz kommen? Punkte ___

- Definiere eine Schnittstelle in Kategorie C welche von A und B verfeinert
- Im allgemeinsten Fall ist das Kategorie „0“
- Die Parametertypen sind so zu definieren das nur allgemeine Typen verwendet werden. (Parametertypen die in C oder einem Vorgänger liegen)
- Es dürfen keine Details der beteiligten Komponenten A und B verwendet werden
- Gesichtspunkt „Gemeinsame Gesprächsbasis“

Erzielte Punkte:_____

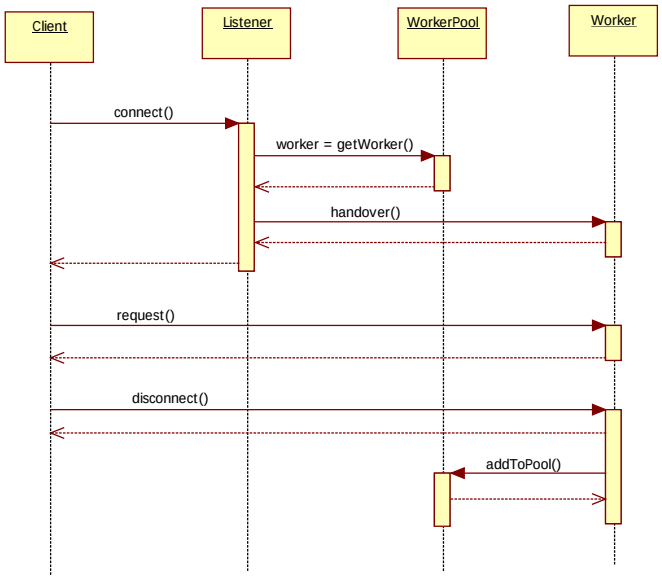
4.) Die Prozessicht/Die physische Sicht



4a) Erläutern Sie die Grundidee des Worker-Pool-Patterns. Punkte ___

- Der Listener-Thread nimmt Connect-Requests von einem Client entgegen
- Er frägt bei dem Workerpool nach einem freien Worker.
- An den freien Worker wird der Client-Request zur ausführung übergeben
- Der Worker kommuniziert mit dem client bis die Verbindung abgebaut wird
- Danach steht der Client wieder zur Verfügung

4b) Erweitern Sie das nachfolgende Sequencediagramm so dass der prinzipielle Ablauf des WorkerPool-Patterns ersichtlich wird. Punkte ___

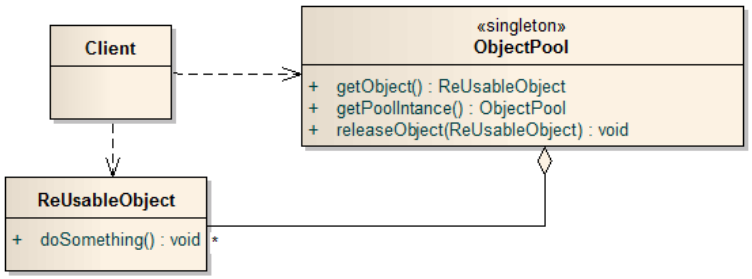


4c) Wie verhält sich das Workerpool-Pattern bei steigender Anzahl von gleichzeitigen Clientrequests? Punkte ___

- Solange Worker vorhanden sind skaliert das Pattern wunderbar
- Andernfalls müssen eigehende Aufrufe warten bis wieder eine Worker zur Verfügung steht
- Alternativ: Können eitere Worker erzeugt werden??

5 Erzeugermuster

5a) Erläutern Sie die Idee des nachfolgend dargestellten Patterns „Objectpool". Punkte ___
Geben Sie ein Beispiel für den Einsatz des Patterns an.



- Definiere einen Pool von Objekten
- Stelle diese Objekte einem Client zur Verfügung
- Mach die Objekte wiederverwendbar wenn es nicht mehr benötigt wird.

5b) Ergänzen Sie das nachfolgende Programmfragment so dass die Arbeitsweise des Clients verdeutlicht wird. Punkte ___

```
public class Client {

 ObjectPool pool = new ObjectPool();

 public  doSomethingWithObjectPool() {

   __ReusablObject o = pool.getObject();
   o.doSomething();

   Pool.releaseObject( o );
}
```

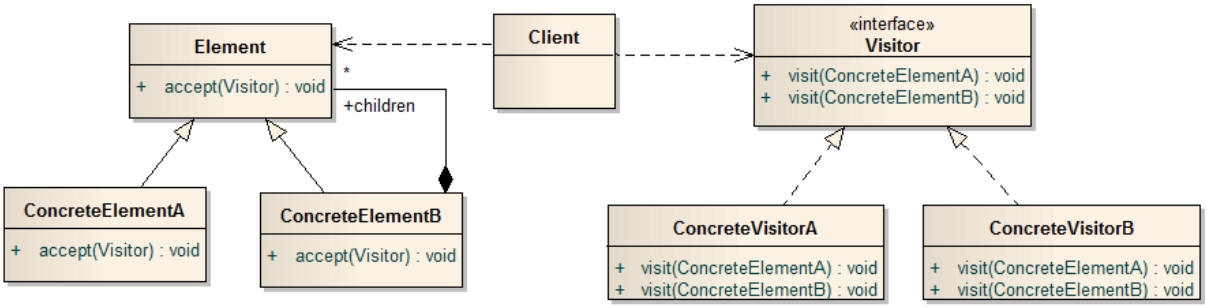
5c) Welche Vorteile bzw. Nachteile/Probleme bringt der Einsatz dieses Patterns mit sich? Punkte ___

- Vorteile:
- Erzeugung von aufwendigen Objekten wird optimiert
- Systemressourcen werden geschont
- Nachteil:
- Rückgabe von Objekten muss durch Client erfolgen (Programmierfehler)
- Die Wiederverwendbaren Objekte müssen sich vor Ihrer Wiederverwendung wieder in einem Initialzustand befinden
- Der Pool ist eine zentrale Stelle welche unter Umständen zu einem Deadlock führen kann.

5d) Wie kann Ihnen das Pattern ObjectPool bei der Implementierung einer multithreaded Applikation helfen? Punkte ___

- Der Zugriff auf die Objekte über den Worker-Pool muss synchronisiert werden
- Geht man davon aus das ein Objekt immer für eine Thread zur verfügung steht müssen die Pool Objekte unter Umständen nicht synchronisiert werden.

6 Verhaltensmuster



6a) Erweitern Sie das nachfolgende Programmfragment so, dass die Implementierung der Methode „ConcreteElementB:accept“ und ConcreteElementA.accept“ der Idee des Visitor-Patterns entspricht — Punkte __

```
public class ConcreteElementA {

 public  accept ( Visitor v) {

    _____v.visit( this )
}
```

```
public class ConcreteElementB {

 public  accept ( Visitor v) {

 ¡  for( Element e : children )
     _____e.accpet( children )

 v.visit(this);
}
```

6b) Erläutern Sie den wesentlichen Unterschied zwischen dem Visitor und dem Interpreterpattern. Wann sollte der Visitor und wann der Interpreter eingesetzt werden? — Punkte __

- Das Visitor pattern stellt nur methoden zur Traversierung bereit. Die eigentliche Logik ist in den Visitor selbst verlagert. D.h. die Traversierung der Datenstruktur ist wiederverwendbar
- Im Interpreterpattern sind Traversierung und Logik zusammengefasst
- Interpreter: Dynamische Ausführung von Skripten
- Visitor: Komplexe Datenstrukturen die unterschiedlich bearbeitet werden

6c) Welches Pattern findet man sehr häuft wenn man mit Collections arbeitet? Was ist die Aufgabe des Patterns und warum erlangt man dadurch höhere Flexibilität? — Punkte __

- Iterator-Pattern
- Durchlaufen einer Ansammlung von Objekten
- Das Pattern verwendet Abstraktion. Dadurch kann die Implementierung der Collection getauscht werden ohne die Iteration ändern zu müssen.

6d) Viele der Patterns verwenden das Prinzip der Abstraktion. Begründen Sie warum dieses Prinzip nützlich für den Bau von flexiblen Softwaresystemen ist. — Punkte __

- Abstraktion versteckt die eigentliche Implementierung hinter einer Schnittstelle
- Wird dieses Prinzip konsequent eingesetzt kann die Implementierung ausgetauscht werden ohne dass der client zu ändern ist.
- Dadurch erhält man Systeme die flexibler angepasst werden können.

Erzielte Punkte:_____