Ausgangssituation

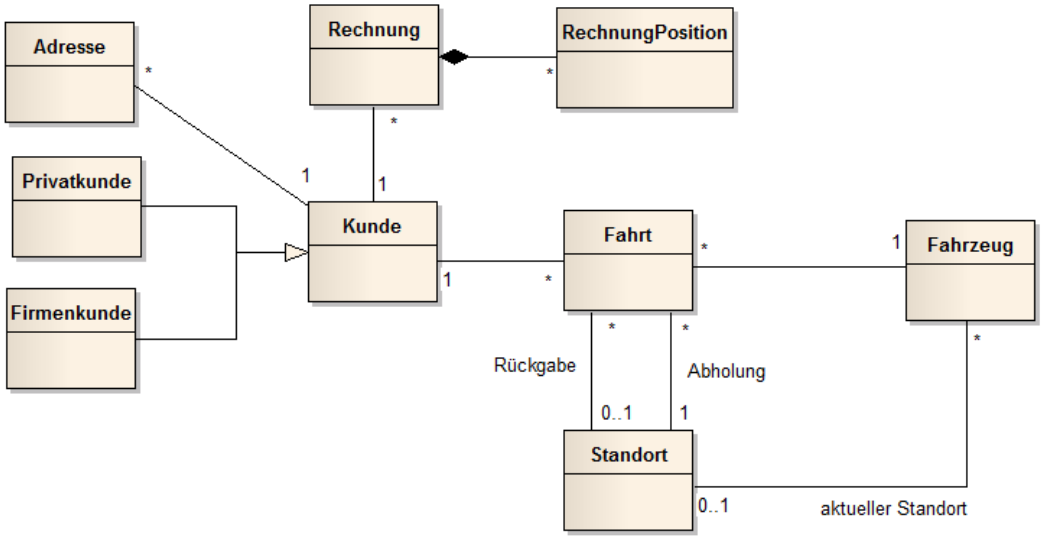
Als Mitarbeiter eines Projektteams werden Sie mit der Entwicklung einer Online-Plattform für die Vermietung von Fahrzeugen beauftragt.

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.

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 ___

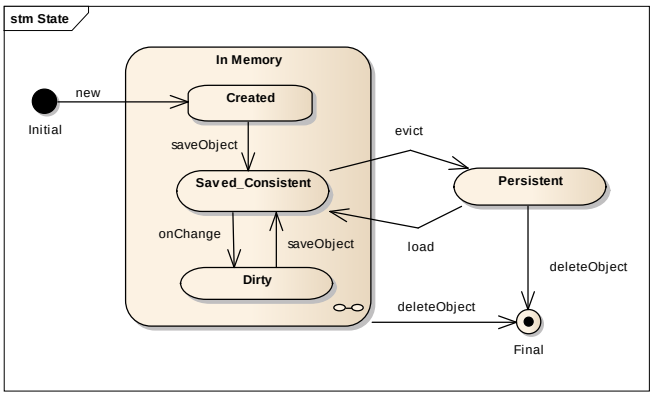


1b) Erläutern Sie das Konzept eines „Aggregates". Beantworten Sie hierzu folgende Aspekte: Was versteht man unter einem Aggregat? Was ist die sog. Root-Entity? Punkte ___

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 ___

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 ___



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 ___

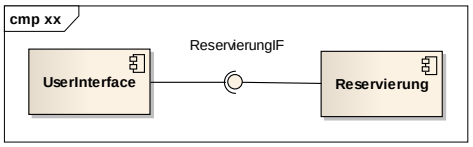
Erzielte Punkte:_____

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 __

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

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




```
public class UserinterfaceImpl  {

  private ____________________________________________________________________;

  public ____________________________  ( ____________________________________ ) {

     ____________________________________________________________________;
  }

  …..
}
```

 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 __

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

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

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

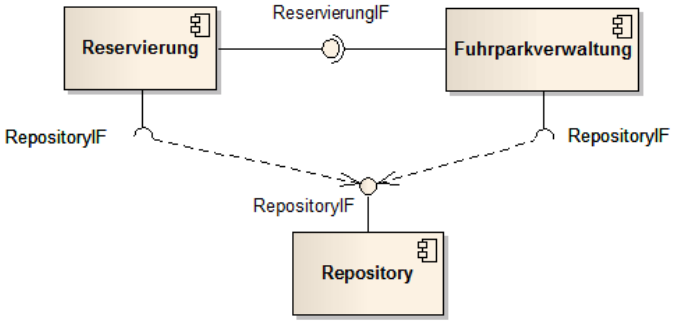
3b) Nennen Sie zwei Probleme, die in dieser Architektur enthalten sind. Sind die Probleme hinderlich für die Flexiblität (Begründung)? Punkte ___

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

 Erzielte Punkte:_____

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 ___

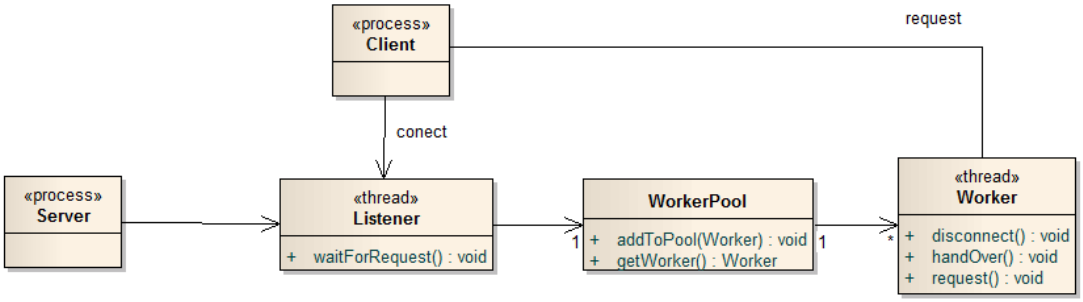


Artefakt	Typ	Kategorie
Implementierung Reservierung	Klasse(n)	
Implementierung Fuhrparkverwaltung	Klasse(n)	
Implementierung Repository	Klasse(n)	
RepositoryIF	Schnittstelle	
ReservierungIF	Schnittstelle	

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 ___

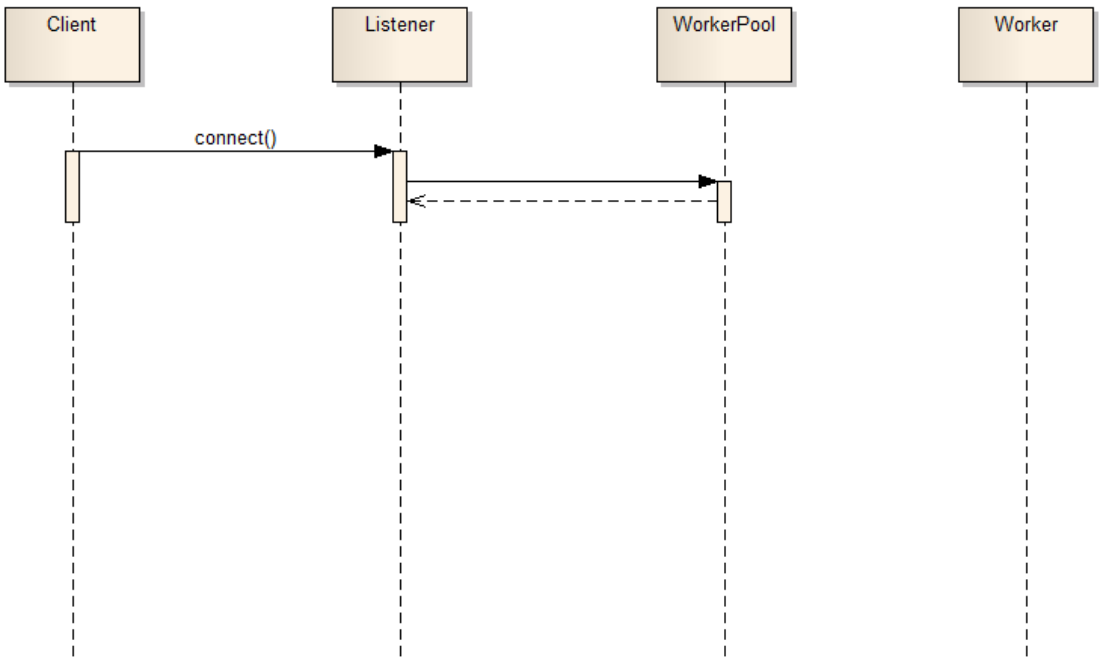
Erzielte Punkte:_____

4.) Die Prozessicht/Die physische Sicht



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

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

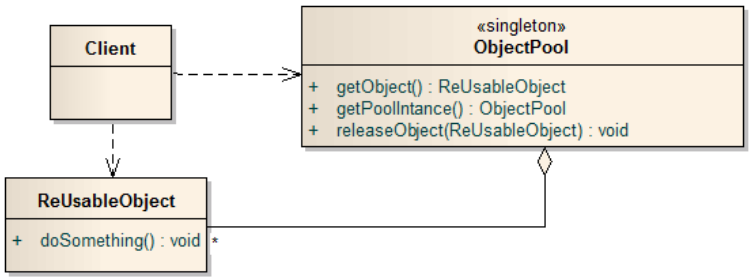


4c) Wie verhält sich das Workerpool-Pattern bei steigender Anzahl von gleichzeitigen Clientrequests? Punkte ___

 Erzielte Punkte:_____

5 Erzeugermuster

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



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

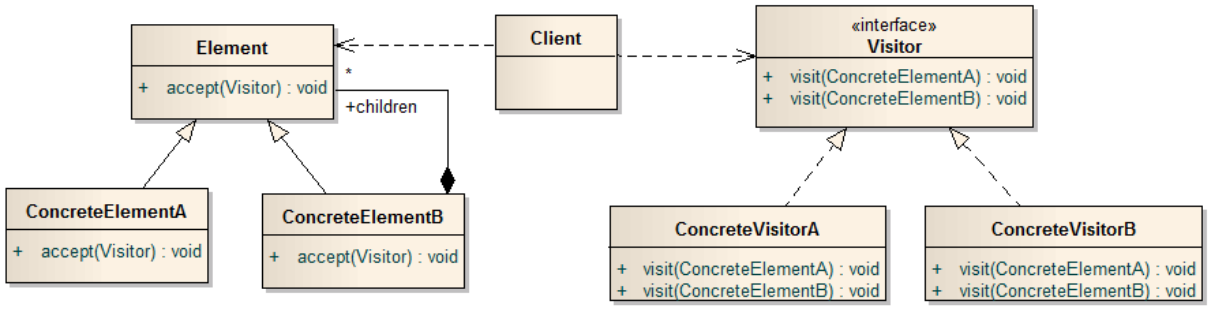
```
public class Client {

  ObjectPool pool = new ObjectPool();

  public  doSomethingWithObjectPool() {

     _____________________________________________________________________

     _____________________________________________________________________

     _____________________________________________________________________
}
```

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

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

 Erzielte Punkte:_____

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  visit ( ____________________________) {

     ____________________________________________

     ____________________________________________

     ____________________________________________
}
```

```
public class ConcreteElementB {

  public  visit ( ____________________________){

     ____________________________________________

     ____________________________________________

     ____________________________________________
}
```

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

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 __

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

Erzielte Punkte:_____