Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

Übungen zur Vorlesung Softwaretechnologie

- Wintersemester 2019/20 -

Dr. Günter Kniesel

Übungsblatt 9

Zu bearbeiten bis: 06.12.2019, 16 Uhr

Bitte fangen Sie **frühzeitig** mit der Bearbeitung an, damit wir Ihnen bei Bedarf helfen können. Checken Sie die Lösungen zu den Aufgaben bitte in Ihr Repository ein, „Erklärungen“ bitte als Textdatei. Fragen zu Übungsaufgaben respektive zur Vorlesung können Sie auf der Mailingliste swt-tutoren@lists.iai.uni-bonn.de, bzw. swt-vorlesung@lists.iai.uni-bonn.de stellen.

Praktische Aufgaben

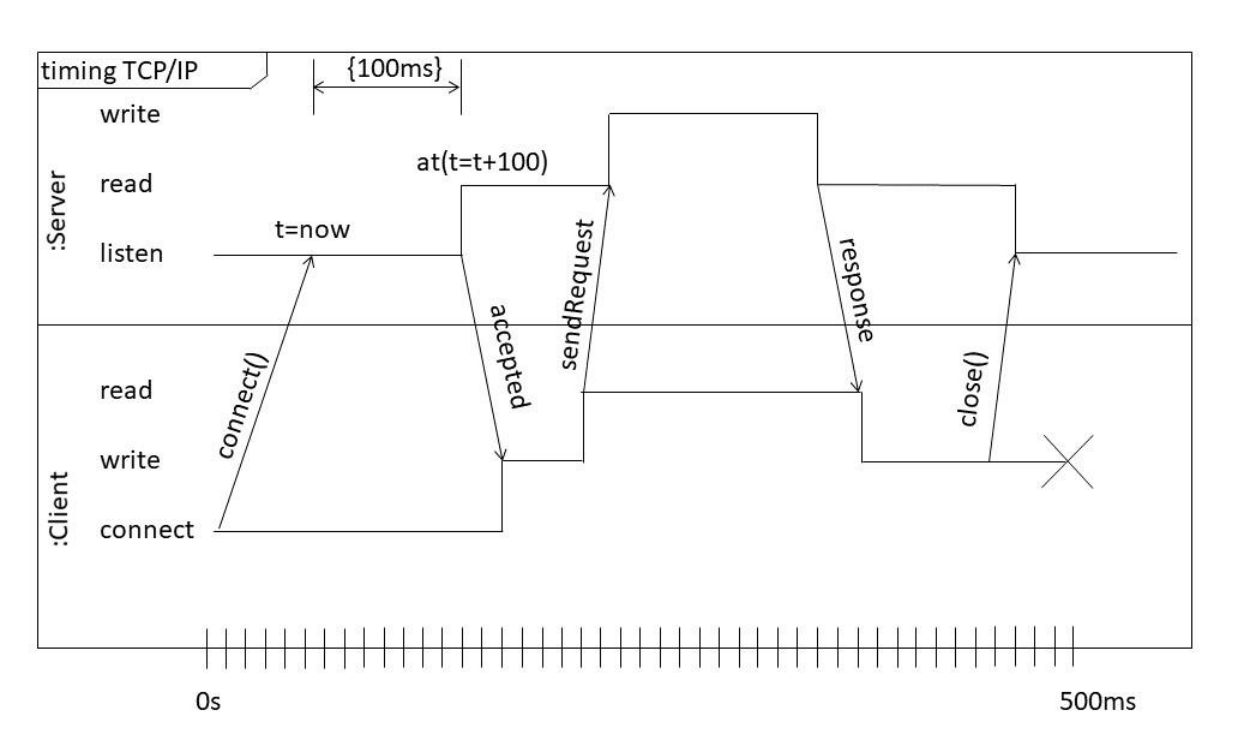
Aufgabe 1. *Zeitverlaufsdiagramm* **7 Punkte**

In dieser Aufgabe sollen Sie den groben systemnahen Ablauf einer TCP-Kommunikation zwischen Server und Client als Zeitverlaufsdiagramm modellieren. Der Ablauf ist folgendermaßen gegeben:

1) Wir gehen hier davon aus, dass zwischen zwei Prozessen, einem Client und einem Server bereits eine Kommunikationsverbindung aufgebaut ist. Der Server ist anfangs immer im „listen“-Modus, in dem er auf eingehende Verbindungen wartet.
2) Beim Aufruf von „connect()“ wird eine Verbindung hergestellt. Der Client bleibt solange im „connect“-Modus, bis der Server die Verbindung herstellt.
3) Der Server akzeptiert nach einer Wartezeit von 100 Millisekunden die Verbindung.
4) Der Server ist nun im „read“-Modus und wartet auf Nachrichten vom Client.
5) Der Client sendet eine Anfrage an den Server („write“-Modus des Clients). Der Server liest diese („read“-Modus).
6) Anschließend antwortet der Server („write“-Modus) und geht danach wieder in den „read“-Modus über.
7) Der Client verarbeitet im „read“-Modus die erhaltene Nachricht.
8) Der Client ist mit der empfangenen Antwort zufrieden, wechselt in den „write“-Modus und schließt die Verbindung mittels einer „close()“-Nachricht.
9) Daraufhin geht der Server in den „listen“-Zustand.

Bitte denken Sie daran, im Diagramm realistische Zeitabstände zu verwenden. Diese dürfen Sie sich, wenn nicht explizit angegeben, selbst aussuchen. Nutzen Sie, falls passend, die vordefinierte Syntax für feste Zeitpunkte/Intervalle.

Nutzen Sie für die Übergänge passende Events.



2x0,5 Modus-Beschriftung und Zuordnung (1xClient, 1xServer) + 2x0.5 explizite Nachrichten (connect(), close()) + 1 für 100 Millisekunden Notation + 0,5 Verbindungabschluss + 1 für korrekte Zustandsübergänge + 1 für sinnvollen Nachrichtenaustausch + 0,5 Zeitachse + 0,5 „Schwimmbahnen“ + 0,5 Bezeichner = 7 Punkte

Aufgabe 2. *Verteilungsdiagramm* **8,5 Punkte**

Gegeben sei eine 2-Schichten Architektur eines Online-Shops, welche aus einem Client und einem Server für die Verwaltung des Shops.

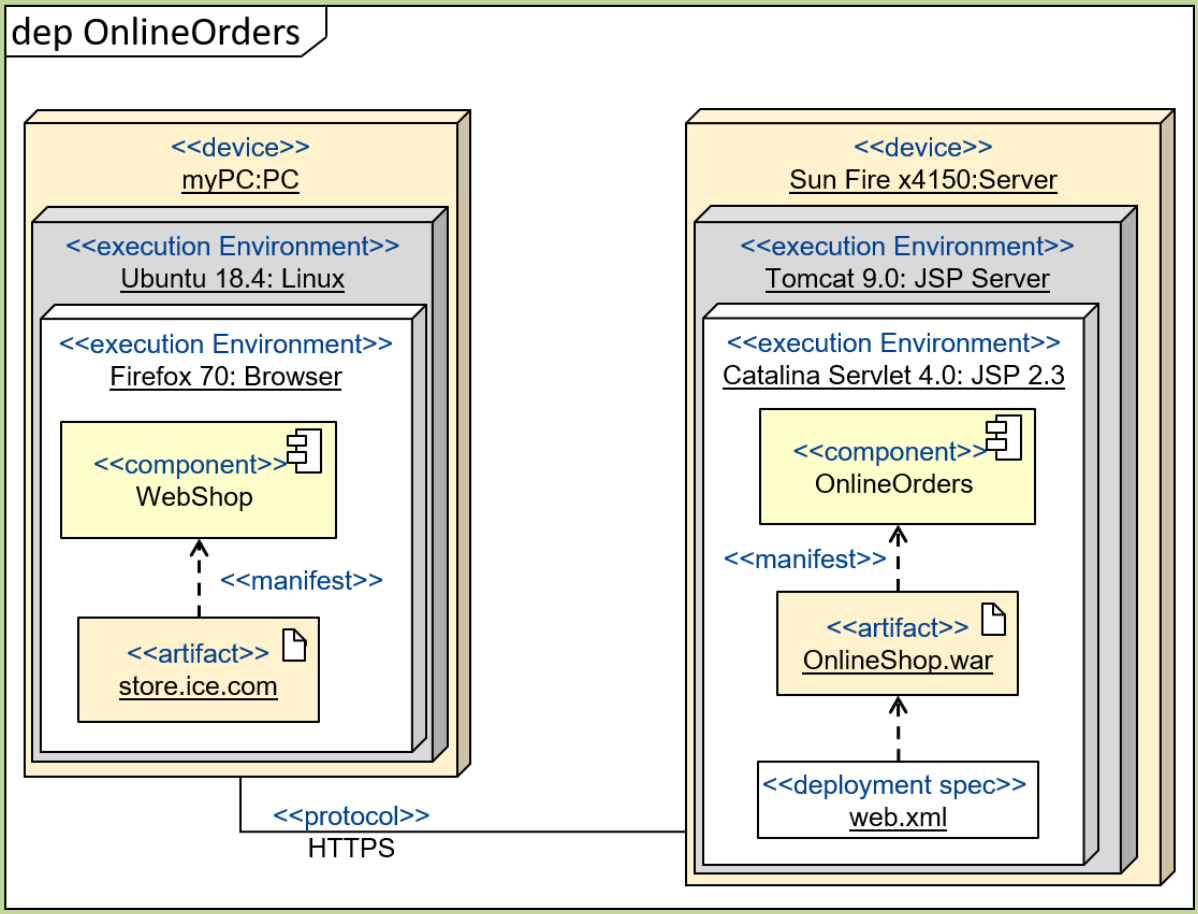
Im Folgenden seien die beiden Schichten genauer beschrieben:

1. Client
    - Der Kunde benutzt seinen *PC*, um den Online Shop zu besuchen
    - Auf dem PC ist *Ubuntu in der Version 18.4* installiert
    - Um die Seite zu besuchen nutzt er die Version *70.0 des Firefox Browsers*
    - Auf Clientseite ist *store.ice.com* eine Realisierung der *WebStore* Komponente
2. Server

- Das Modell des Servers ist ein *Sun Fire x4150 Server*
- Betrieben wird der Online Shop mithilfe eines *Catalina Servlet 4.0 Containers (JSP 2.3)*
- Das Servlet läuft auf einem *Tomcat 9.0 JSP Server*
- Die Komponente *OnlineOrders* wird durch *OnlineShop.war* realisiert
- *web.xml* ist eine Konfiguration von OnlineShop.war

Der Client kommuniziert mit dem Shop über eine herkömmliche HTTPS Verbindung.

Modellieren Sie diesen Sachverhalt in einem Verteilungsdiagramm.



2x0,5 Devices + 4x0,5 Exec.Env. + 2x0,5 Komponente + 2x0,5 Artefakte + 2x0,5 Manifestation + 1 für Deploym.Spec. + 1 für Protokoll + 0,5 Bezeicher = 8,5 Punkte

Aufgabe 3. *Entwurfsmuster für eine Bibliotheksverwaltung* **13 Punkte**

Gegeben sei der Kern einer Bibliotheksverwaltung, deren Java-Programmcode Sie hier finden:

ssh://gitolite-se-swt@git.iai.uni-bonn.de/swt2019_readonly

Ziel ist es nun, dieses Minimal-System an geeigneter Stelle mit Hilfe von Entwurfsmustern zu erweitern.

a) **(5 Punkte)** In unserem System führt jedes Objekt der Klasse Ausleihgegenstand eine Warteliste, auf der Kunden sich bei Interesse für ein aktuell entliehenes Medium eintragen können. Realisieren Sie mittels eines geeigneten Musters, dass alle Kunden aus der Warteliste eines Mediums informiert werden, sobald dieses Medium zurückgegeben wurde. Als Reaktion auf die Benachrichtigung versucht jedes Kunden-Objekt erneut, das Medium auszuleihen.

Hinweise:

- Der mitgelieferte Programmcode enthält im Paket `test` die Klasse `BibliotheksTest`. Benutzen Sie diese als Hauptprogramm.
- Überlegen Sie genau welche Rolle in dem Entwurfsmuster von vorhandenen Programmelementen (Klassen, Methoden, Felder, etc.) übernommen werden können und für welche Rollen Sie neue Elemente brauchen.

```
public class Ausleihgegenstand {
    ...
    // 3 Punkte
    public void notifyObservers(){
        for(Benutzer b : warteliste) {
                b.update(this);
        }
    }

    public void gibtZurück(Benutzer b) {
        // Ausleihgegenstand ist derzeit gar nicht ausgeliehen
        if(this.ausleiher == null){
                System.out.println("'"+titel+"' ist derzeit nicht ausgeliehen.
                Rückgabe nicht möglich!");
        }
        // Ausleihgegenstand ist derzeit ausgeliehen
        else{
                // Vom Benutzer selbst
                if(this.ausleiher == b){
                        this.ausleiher = null;
                        System.out.println(b.getName()+" hat '"+titel+"'
                        erfolgreich zurückgegeben!");
```

```
                this.notifyObservers();
            }
            // Von jemand anderem
            else{
                System.out.println("Rückgabe von '"+titel+"' nicht möglich."
                    + " Der Gegenstand ist nicht von "+b.getName()+"
                    ausgeliehen!");
            }
        }
    }
    ...
}

public class Benutzer {
    ...
    // 2 Punkte
    public void update(Ausleihgegenstand medium){
        //benachrichtige Kunden
        System.out.println("Benachrichtige " + getName() +
                ": Das Medium '" + medium.getTitel() +
                "' wurde zurückgegeben!");

        //versuche erneute Ausleihe
        medium.fragtAn(this);
    }
    ...
}
```

Hinweis: Hier wurde die Implementierung mit Hilfe einer Variante des Observer-Pull-Modells durchgeführt, in der ConcreteObserver-Instanzen Referenzen auf ConcreteSubject-Instanzen nicht in einem Feld speichern, sondern als Parameter der Update-Methode erhalten. Das ist auch dann anwendbar, wenn der Observer viele Subjects beobachtet. Dadurch, dass jeder Aufruf der update()-Methode explizit angibt, von welchem Objekt er aufgerufen wurde, kann der ConcreteObserver gezielt in dem richtigen Objekt nach dem geänderten Zustand suchen. Dies ist immer noch ein Pull-Modell, denn der geänderte Zustand muss durch explizite Nachrichten an den ConcreteSubject abgefragt werden.

b) **(3 Punkte)** In einem nächsten Schritt sollen Erfolgs- und Misserfolgs-Meldungen der Bibliotheks-verwaltung per Mail an den Benutzer geschickt werden, der davon betroffen ist. Erstellen Sie dazu eine Klasse `EmailSystem`, die das E-Mail-System repräsentiert und eine Methode `sendeMail(String name, String nachricht)` unterstützt. Implementieren Sie die Funktion, so dass sie auf `System.out "Email an " + name + ": "` und in der nächsten Zeile `nachricht` ausgibt.

Siehe c).

c) **(2 Punkte)** Garantieren Sie durch die korrekte Anwendung eines geeigneten Entwurfsmusters, das maximal eine Instanz von `EmailSystem` erzeugt und verwendet wird.

```
public class EmailSystem {

	protected static EmailSystem meineInstanz;	// 0,5 P

	// 0,5 P
	private EmailSystem() {
		// private! Verhindert die direkte Generierung der Klasse
		System.out.println("Initialisiere E-Mail-System");
	}

	// 3 Punkte <- zu b)
	public void sendeMail(String name, String nachricht) {
		System.out.println("Schicke E-Mail an " + name + ": "
		+ nachricht);
	}

	// 1 Punkt
	public static EmailSystem getInstance() {
		if (meineInstanz == null)
			meineInstanz = new EmailSystem();
		return meineInstanz;
	}
}
```

d) **(3 Punkte)** Binden Sie die neue Klasse und ihre Funktionalität in das Gesamtsystem ein. Wenn Sie nun das Hauptprogramm laufen lassen, sollten Sie abgesehen von den zusätzlichen "Email an …"-Meldungen die gleiche Ausgabe wie in a) erhalten.

```
public class Ausleihgegenstand {
	...
	// 1 Punkt
	private void ausleihErfolgsmeldung(Benutzer b) {
		EmailSystem.getInstance().sendeMail( b.getName() ,
			"'"+titel+"' wird an Sie ("+b.getName()+") ausgeliehen. Viel
			Spaß damit!\n" );
	}

	// 1 Punkt
	private void ausleihFehlermeldung(Benutzer b, String message) {
		EmailSystem.getInstance().sendeMail( b.getName(),
			"Ausleihe von '"+titel+"' an "+b.getName()+" nicht moeglich:\n
			-->\t"+ message+"\n" );
	}
	...
}
```

```
// Eine Einbindung kann folgendermaßen geschehen:

public void fragtAn(Benutzer b){

	// Ausleihgegenstand ist derzeit nicht ausgeliehen
	if(this.ausleiher == null){

		// Warteliste ist leer
		if(this.warteliste.isEmpty()){
			this.ausleiher = b;
			System.out.println("'"+titel+"' wurde ausgeliehen
				an" + b.getName() + ".");
			ausleihErfolgsmeldung(b);  // 1 Punkt
		}
	}
}
```

Theoretische Aufgaben

Aufgabe 4. *Übergeordnete Belange* **8,5 Punkte**

a) **(0,5 Punkte)** Was bezeichnen Realzeitanforderungen?
Die Software soll in einer bestimmten (typischerweise sehr kurzen) Zeitspanne, deutlich unterhalb Sekundenbereich, reagieren oder zu einem bestimmten, sehr genau definierten Zeitpunkt etwas tun.

b) **(1 Punkt)** Welche Diagramme werden zur Veranschaulichung von nebenläufigen Interaktionspartnern verwendet? Erklären Sie, welche davon in welchem Kontext besser geeignet sind.
Timing- und Sequenz-Diagramm.
Bei Realzeitanwendungen meist Timing-Diagramm zur Spezifikation zeitkritischer Zustandsübergänge. Zur Verwendung von Kontrollstrukturen, wie Schleifen, Alternativen etc. eher Sequenzdiagramm.
(Vgl. Skript Folie 7c)-11: „Sequenz- versus Zeitverlaufsdiagramm“)

c) **(0,5 Punkte)** Welches Diagramm wird für die Veranschaulichung von Hardware/Software-Zuordnung verwendet?
Verteilungsdiagramm.

d) **(1,5 Punkte)** Wie würden Sie die Logdaten eines Sensors speichern?
Als Dateien, da diese Daten meist nur textuell strukturiert sind und nur bis zur Auswertung benötigt werden. Außerdem sind die Schreibgeschwindigkeiten auf reinen Daten deutlich schneller. Dateien verbrauchen weniger Speicherplatz als Datenbanken (Gerade bei sehr umfangreichen Logs sinnvoll).

e) **(3 Punkte)** Wie können Sie ein Klassendiagramm auf eine Datenbank abbilden? Was können Sie nicht abbilden?
- Jede Klasse eine Tabelle
- Jedes Attribut eine Spalte
- Jede Instanz eine Zeile

- n:m-Beziehung als eigene Tabelle (da in UML auch bestenfalls eigene Klasse)
- 1:n-Beziehung als Fremdschlüssel
- Nicht abbildbar sind die Methoden

f) **(0,5 Punkte)** Wie ist die Zugriffskontrollmatrix aufgebaut?
Zeilen: Akteure, Spalten: Objekte, Zellen: legale Operationen

g) **(1 Punkt)** Welchem Kontrollparadigma entspricht das MVC?
Eventbasierte Kontrolle: Wenn sich das Model ändert (Event), gibt der Controller (zentral) den Views die Anweisung, sich zu aktualisieren, oder die Models benachrichtigen die Views selbst über Änderungen (dezentral).
(Vgl. Skript Folie 7c)-41: „Prozedurgesteuerte vs. eventgesteuerte Kontrolle“)

h) **(0,5 Punkte)** Welche Grenzfälle gibt es beim Systementwurf?
Initialisierung (Hochfahren), Terminierung (Runterfahren), Fehler.

∑ 37 Punkte