Übungen zur Vorlesung Softwaretechnologie

- Wintersemester 2018/19 -

Dr. Günter Kniesel

Übungsblatt 8 - Lösungen

Aufgabe 1. *Proxy-Pattern* (5 Punkte)

a) Erläutern Sie kurz die Motivation und die Funktionsweise des Proxy-Entwurfmusters.

- Absicht
  a. Stellvertreter für ein anderes Objekt
  b. bietet Kontrolle über Objekt-Erzeugung und -Zugriff
- Motivation
  a. kostspielige Objekt-Erzeugung verzögern (z.B.: große Bilder)
  b. verzögerte Objekterzeugung soll Programmstruktur nicht global verändern
  c. Beim Zugriff gewisse Zusatzoperationen ausführen (checks, Buchhaltung, Benachrichtigung, Pufferung, ...)

b) Welches sind die teilnehmenden Rollen (Klassen) des Proxy-Entwurfsmusters? Welche Aufgaben übernehmen diese? Welche Operationen sind notwendig?

- Proxy
  a. bietet gleiches Interface wie "RealSubject"
  b. referenziert eine "RealSubject"-Instanz
  c. kontrolliert alle Aktionen auf dem "RealSubject"
- Subject
  a. definiert das gemeinsame Interface
- RealSubject
  a. das Objekt das der Proxy vertritt
  b. eigentliche Funktionalität
- Zusammenspiel
  a. selektives Forwarding

c) Nennen Sie mindestens drei Beispiele für die Anwendung eines Proxy-Entwurfsmusters und erläutern Sie kurz die Motivation für den jeweiligen Pattern-Einsatz.

- Virtueller Proxy
  a. verzögerte Erzeugung "teurer" Objekte bei Bedarf
  b. Beispiel: Bilder in Dokument, persistente Objekte
- Remote Proxy
  a. Zugriff auf entferntes Objekt
  b. Objekt-Migration
  c. Beispiele: CORBA, RMI, mobile Agenten

- Concurrency Proxy
    a. nur eine direkte Referenz auf RealSubject
    b. locking des RealSubjects vor Zugriff (threads)
- Copy-on-Write
    a. kopieren erhöht nur internen "CopyCounter"
    b. wirkliche Kopie bei Schreibzugriff und "CopyCounter">0
    c. Verzögerung teurer Kopier-Operationen
- Protection Proxy (Zugriffskontrolle)
    a. Schmaleres Interface
        a.i. "Kritische" Operationen ausgeblendet
        a.ii. andere via Forwarding implementiert
    b. ganz anderes Interface
        b.i. Autorisierung prüfen
        b.ii. direkten Zugriff gewähren oder Forwarding
    c. Beispiel: "CHOICES" Betriebssystem, JDK

Aufgabe 2. *Entwurfsmuster im Einsatz* (13 Punkte)

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

ssh://gitolite-se-swt@git.iai.uni-bonn.de/swt2018_readonly

Ziel ist es nun, obiges 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, um Ihre Implementierung zu testen.
- 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.

Hinweis: Nachfolgend 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.

```
public class Ausleihgegenstand {
   ...

      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 {

   ...

      public void update(Ausleihgegenstand medium){

            //benachrichtige Kunden
            System.out.println("Benachrichtige " + getName() +
            ": Das Medium '" + medium.getTitel() +
            "' wurde zurückgegeben!");

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

```
...

}
```

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.

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;
	/* Die Variable "meineInstanz" spielt in diesem Beispiel
	* die Rolle des Arrays/der Liste "instancePool" des Allgemeinen
	* Singleton-Patterns, in dem auch eine feste Anzahl > 1 an Instanzen
	* moeglich sind.
	* In dieser Aufgabe ist nur eine Instanz gewuenscht, weshalb
	* keine Liste notwendig ist.
	*/

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

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

	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 {
...

	private void ausleihErfolgsmeldung(Benutzer b) {
```

```
            EmailSystem.getInstance().sendeMail( b.getName() ,
            "'"+titel+"' wird an Sie ("+b.getName()+") ausgeliehen. Viel
            Spaß damit!\n" );
      }

      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);
                  }
```

Aufgabe 3. *Entwurfsmuster im Einsatz (10 Punkte)*

a) *(6 Punkte)* Im Studiensekretariat wird ein Bandlaufwerk für die langfristige Speicherung von Daten genutzt. Jeden Tag wird eine Sicherungskopie dieser für das Prüfungsbüro wichtigen Daten gemacht. Ein Schreibvorgang der Daten dauert hierbei über 60 Minuten, da das Gerät relativ alt ist. Das Studiensekretariat hat aber keine Mittel um ein neues Gerät zu kaufen und stellt Ihnen die Aufgabe, eine Möglichkeit zu finden, das Gerät ohne Änderung der verwendeten Schnittstelle weiter zu verwenden und

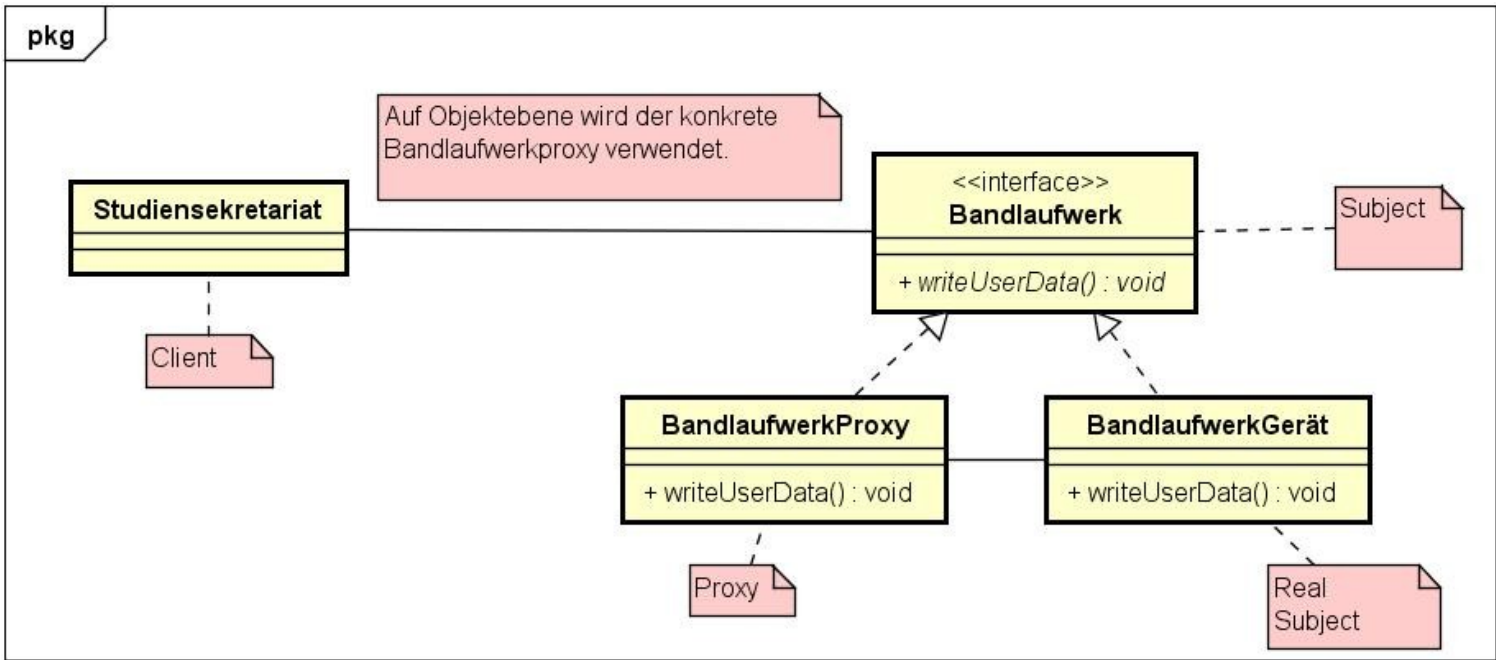
   a. Schreibvorgänge nur zwischen 22:25 und 05:45 Uhr zu erlauben bzw.

   b. Schreibvorgänge außerhalb dieses Zeitfensters oder während eines laufenden Schreibvorgangs in einer Warteschlange zwischen zu speichern und verzögert auszuführen.

   Überlegen Sie mit welchem Entwurfsmuster man die obigen Anforderungen realisieren kann, wie die Grundstruktur der Anwendung aussehen könnte, und wie das Muster auf diese Grundstruktur angewendet werden kann.

   Zeichnen Sie für Ihre Lösung ein Klassendiagramm. Für die Klassen und Methoden, die eine Rolle im Rahmen des Entwurfsmusters spielen, erläutern Sie die jeweilige Rolle in Form einer Notiz an dem zugehörigen Element.

➔ Adapter-Pattern:



b) *(4 Punkte)* Eine neue Version der Sekretariats-Software unterstützt keine Bandlaufwerke mehr, sondern nur noch die Hochgeschwindigkeits-Sicherung auf Blu-ray Discs, die eine andere Schnittstelle haben als Bandlaufwerke. Wie müssen Sie den Entwurf aus a) anpassen, um dennoch das Bandlaufwerk weiter verwenden zu können? Welches Entwurfsmuster setzen Sie ein?

Zeichnen Sie wieder ein Diagramm Ihrer Lösung und erläutern Sie die Rollen in Form von Notizen.

➔ Adapter-Pattern: