Übungen zur Vorlesung

Softwaretechnologie

- Wintersemester 2010/11 -

Dr. Günter Kniesel

Übungsblatt 6

Aufgabe 1. *Proxy-Pattern* (4 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 (zB: große Bilder)
  b. verzögerte Objekterzeugung soll Programmstruktur nicht global verändern

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-Entwurfmusters 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
        i. "Kritische" Operationen ausgeblendet
        ii. andere via Forwarding implementiert
    b. ganz anderes Interface
        i. Autorisierung prüfen
        ii. direkten Zugriff gewähren oder Forwarding
    c. Beispiel: "CHOICES" Betriebssystem, JDK

d) Worin unterscheidet sich das „Proxy“-Entwurfsmuster von dem Entwurfsmuster „Adapter“?

- Proxy
    a. gleiches Interface
    b. kontrolliert Zugriff
- Adapter
    a. verschiedenes Interface

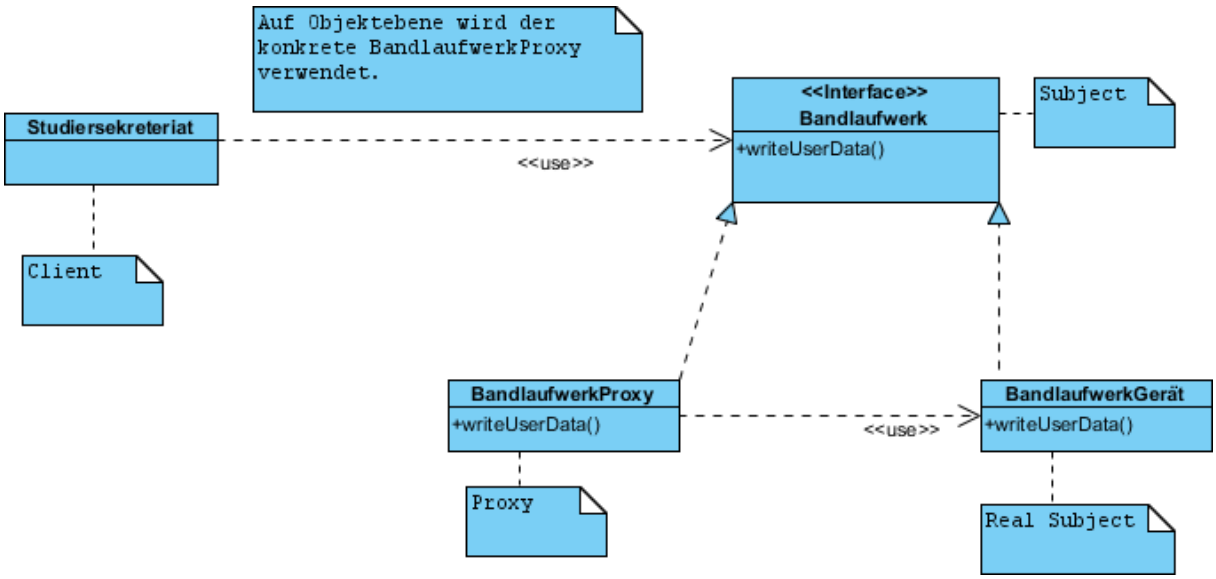
Aufgabe 2. *Entwurfmuster im Einsatz* (10 Punkte)

a) 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.

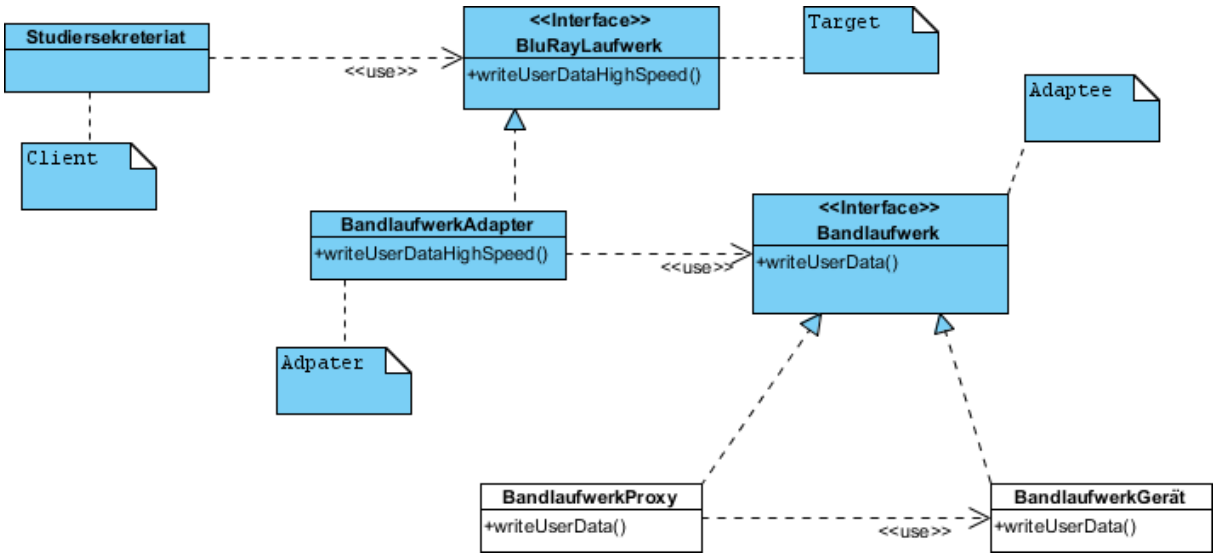
    b. Schreibvorgänge außerhalb dieses Zeitfensters oder während eines laufenden Schreibvorgangs in einer Warteschlange zwischen zu speichern und verzögert auszuführen.

    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.



b) 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:

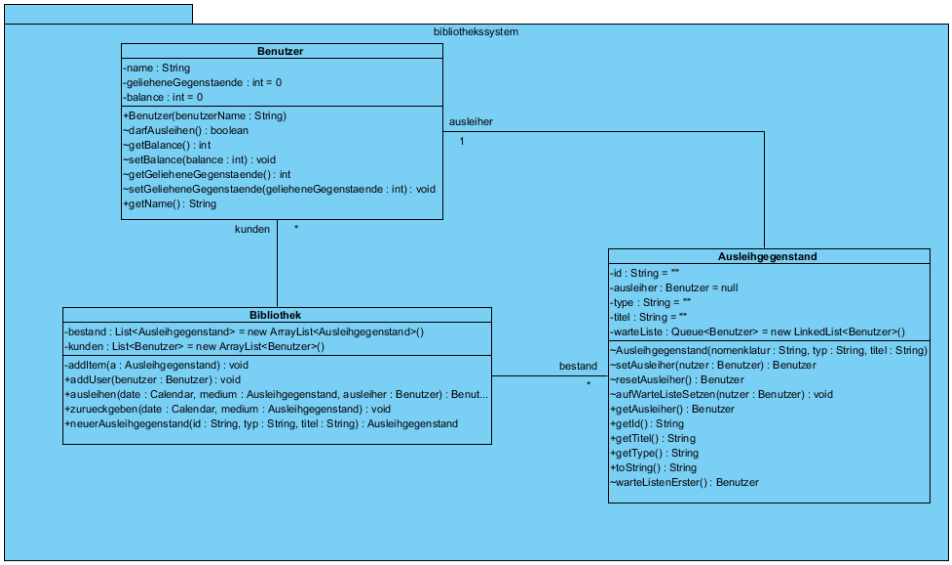


Aufgabe 3. *Modell-Generierung* (3 Punkte)

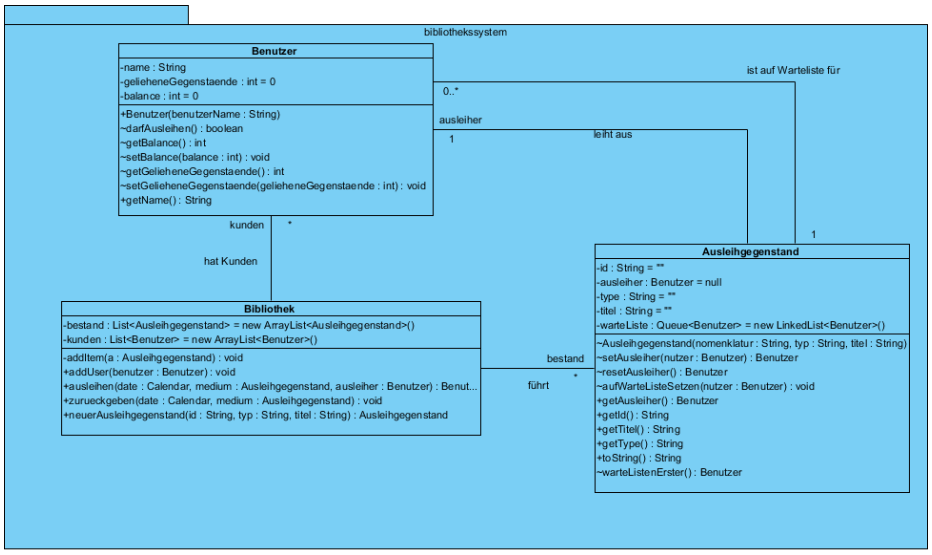
Gegeben sei der Kern einer Bibliotheksverwaltung, deren Java-Programmode Sie im SVN-Repository unter …se/swt2010/gruppe/share/Bibliothek.zip finden.

a) Erstellen Sie ein Projekt das den obigen Code enthält. Dann wählen Sie aus dem Kontextmenü des Projektes den Menüpunkt „Open SDE-EC“ aus und wechseln Sie zur Modell-Perspektive. Wählen Sie aus dem Menü „Modeling“ den Befehl „Instant Reverse…“ und fügen Sie als „Source Folder“ den „src“-Ordner des Projektes hinzu. Nach Klick auf „OK“ wählen Sie in der „Class Repository“ View die

importierten Pakete aus und wählen aus dem Kontextmenü „Reverse Resources to → New Class Diagram“. Speichern Sie das Ergebnis in Ihrem SVN-Repository.



b) Erörtern Sie, ob die generierten Assoziationen aus Ihrer Sicht sinnvoll sind und geben Sie den Assoziationen aussagekräftige Namen. Ergänzen Sie fehlende As soziationen.



Aufgabe 4. *Entwurfsmuster* (13 Punkte)

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

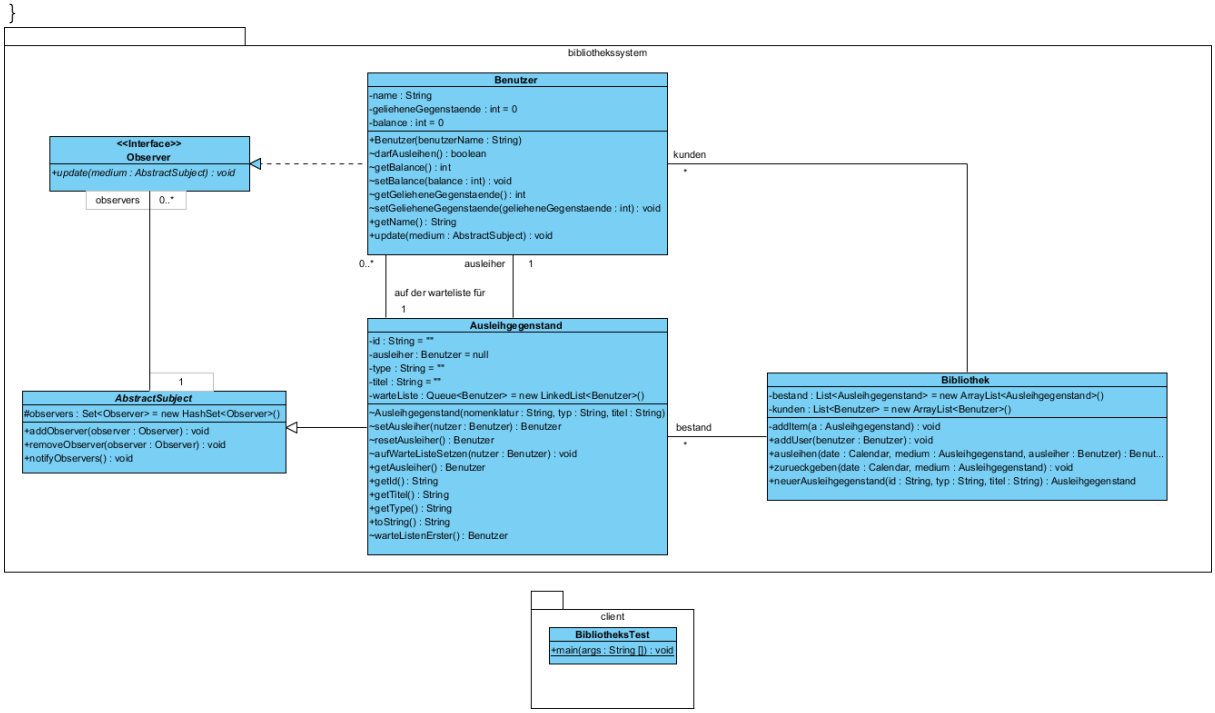
Hinweis: Der mitgelieferte Programmcode enthält im Paket `client` die Klasse `BibliotheksTest` . Benutzen Sie diese als Hauptprogramm, um Ihre Implementierung zu testen.

a) 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 der erste Kunde aus der Warteliste eines Mediums informiert wird, sobald dieses Medium zurückgegeben wurde. Als Reaktion auf die Benachrichtigung soll das Kunden-Objekt eine E-Mail an den entsprechenden Kunden schicken (symbolisiert durch die Text-Ausgabe „Schicke E-Mail an *...Name...*: *... Inhalt...*“).

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 ist, 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 interface Observer {

      void update(AbstractSubject medium);

   }


Public abstract class AbstractSubject {
      protected Set<Observer> observers = new HashSet<Observer>();

      public void addObserver(Observer observer) {
            if (observer == null)
                  return;
            observers.add(observer);
      }

      public void removeObserver(Observer observer) {
            if (observer == null)
                  return;
            observers.remove(observer);
      }

      public void notifyObservers() {
            for(Observer observer:observers) {
                  observer.update(this);
            }
      }
   }
```

```
public class Benutzer implements Observer{

…

	@Override
	public void update(AbstractSubject medium) {
		System.out.println("Schicke E-Mail an " + getName() +
		": Das Medium '" + ((Ausleihgegenstand) medium).getTitel() +
		"' ist verfügbar!");
	}
    }

public class Ausleihgegenstand extends AbstractSubject {

	…
	Benutzer setAusleiher(Benutzer nutzer){
		…

		if(warteListe.peek()==nutzer){
			removeObserver(nutzer);
			warteListe.poll();
			addObserver(warteListenErster());

			…
		} …
	}

	Benutzer resetAusleiher(){
		ausleiher = null;
		notifyObservers();
		…
	}

	void aufWarteListeSetzen(Benutzer nutzer) {
		if(!warteListe.contains(nutzer)){
			warteListe.offer(nutzer);
			addObserver(warteListenErster());
		}
	}
	…
    }
```



b) In einem nächsten Schritt soll das E-Mail-System der Bibliothek erweitert werden. Erstellen Sie dazu eine Klasse `EmailSystem`, die das E-Mail-System repräsentiert und eine Methode `sendeMail(String name, String nachricht)` unterstützt (die eine Zeichenkette auf `System.out` ausgibt). Garantieren Sie durch die korrekte Anwendung eines geeigneten Entwurfsmusters, das maximal eine Instanz von `EmailSystem` erzeugt und verwendet wird.
Binden Sie zuletzt die neue Klasse und ihre Funktionalität in das Gesamtsystem ein.

```
package bibliothekssystem;

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);
	}

	protected static EmailSystem getInstance() {
		if (meineInstanz == null)
			meineInstanz = new EmailSystem();
		return meineInstanz;
	}

}

public class Benutzer implements Observer{

…

	@Override
	public void update(AbstractSubject medium) {
		EmailSystem.getInstance().sendeMail(getName(), "Das Medium '" +
		((Ausleihgegenstand) medium).getTitel() + "' ist verfügbar!");
	}
   }
```