Softwaretechnik
Wintersemester 2024/25, Übungsblatt 7

Prof. Dr.-Ing. Norbert Siegmund
Annemarie Wittig, B. Sc.

Diskussion am: 18.12.2024

Aufgabe 1: Grundlagen

a) Was ist Qualität?

b) Nennen und erläutern Sie die fünf Kriterien für gutes Design.

c) Nennen und erläutern Sie die fünf Regeln für gutes Design.

d) Nennen und erläutern Sie drei beliebige GRASP Patterns.

Aufgabe 2: SOLID-Kriterien im SantaGiftManager: Das perfekte Tool für stressfreie Weihnachten

In der Weihnachtswerkstatt herrscht Hochbetrieb, und die Elfen kommen kaum mit dem Verpacken hinterher. Aber dieses Jahr ist alles anders – dank des SantaGiftManagers! Dieses revolutionäre Tool bringt Ordnung ins Geschenkchaos und sorgt dafür, dass kein Geschenk ohne Schleife bleibt. Egal ob rechteckige Box oder sternförmiges Meisterwerk, der SantaGiftManager organisiert alles mit Stil und Präzision.

Aber damit nicht genug: Die Weihnachtself & Co. KG hat Großes vor. Die Elfen planen eine Erweiterung des Tools, um *mehrere Geschenke gleichzeitig zu verwalten* und für diese *automatisierte Routen zu planen*. Erstmal wird sich auf Kuscheltiere, wie Bären, Hasen oder Katzen konzentriert, für die in einem Arbeitsschritt Lieferanweisungen kinderleicht organisiert werden. Und das ist erst der Anfang! Die Elfen haben angedeutet, dass bald auch neue Geschenktypen hinzukommen sollen. Bleibt gespannt – Weihnachten wird nie wieder so chaotisch wie früher! 🎄

Für den Anfang haben sich die Elfen einen Prototypen überlegt, der dafür genutzt werden soll, die Kuscheltiere automatisiert zu versenden. Da sie aber nicht allzu bewandert mit der Programmiertechnik sind, haben sie Sie hinzugezogen, um den Prototypen zu bewerten und Tipps zu erhalten, was sie anpassen könnten und sollten.

```java
public class SantaGiftManager {
    private RoutePlanner routePlanner;
    private List<PlushRabbit> rabbitGifts;
    private List<PlushBear> bearGifts;
    private List<PlushCat> catGifts;

    public void addAddress() {
        for (PlushRabbit rabbit: rabbitGifts) {
            rabbit.addAddress(addressList);
        }
        for (PlushBear bear: bearGifts) {
            bear.addAddress(addressList);
        }
        for (PlushCat cat: catGifts) {
            cat.addAddress(addressList);
        }
    }
}

public class PlushRabbit {
    private Address address;

    public void addAddress(RoutePlanner routePlanner) {
        routePlanner.addStop(address, <mehr Geschenkinformationen>)
    }
}

public class PlushBear {
    private Address address;

    public void addAddress(RoutePlanner routePlanner) {
        routePlanner.addStop(address, <mehr Geschenkinformationen>)
    }
}

public class PlushCat {
    private Address address;

    public void addAddress(RoutePlanner routePlanner) {
        routePlanner.addStop(address, <mehr Geschenkinformationen>)
    }
}

```

a) Gegen welches SOLID-Prinzip verstößt der gegebene Prototyp? Begründen Sie Ihre Antwort.

b) Passen Sie den Prototypen dem SOLID-Prinzip entsprechend an. Hierfür können Sie entweder (a) Quelltext oder (b) eine genaue Beschreibung der Änderungen, die am Prototypen vorgenommen werden müssen, angeben.

Bonusaufgabe

Die Elfen haben Ihren Anmerkungen konzentriert zugehört und einen neuen Prototyp entworfen. Sie bitten Sie darum, diesen genauso zu bewerten, wie den ersten.

```java
public class SantaGiftManager {
    private RoutePlanner routePlanner;
    private List<PlushGift> plushToys;

    public void addAddress() {
        for (PlushGift toy : plushToys) {
            if (toy instanceof PlushRabbit) {
                ((PlushRabbit) toy).addAddress(routePlanner);
                System.out.println("Hase entdeckt!");
            } else if (toy instanceof PlushBear) {
                ((PlushBear) toy).addAddress(routePlanner);
                System.out.println("Bär entdeckt!");
            } else if (toy instanceof PlushCat) {
                ((PlushCat) toy).addAddress(routePlanner);
                System.out.println("Katze entdeckt!");
            }
        }
    }
}

abstract class PlushGift {
    private Address address;

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public abstract void addAddress(RoutePlanner routePlanner);
}

public class PlushRabbit extends PlushGift {
    @Override
    public void addAddress(RoutePlanner routePlanner) {...}
}

public class PlushBear extends PlushGift {
    @Override
    public void addAddress(RoutePlanner routePlanner) {...}
}

public class PlushCat extends PlushGift {
    @Override
    public void addAddress(RoutePlanner routePlanner) {...}
}
```

c) Gegen welches SOLID-Prinzip verstößt der neue Prototyp? Begründen Sie Ihre Antwort.

d) Passen Sie den Prototypen dem SOLID-Prinzip entsprechend an. Hierfür können Sie entweder (a) Quelltext oder (b) eine genaue Beschreibung der Änderungen, die am Prototypen vorgenommen werden müssen, angeben.

Nach den viele Prototyp-Anpassungen sind die Elfen doch nicht mehr so überzeugt davon, den SantaGiftManager nochmal vor der Weihnachtssaison anzupassen. Vielleicht reicht Stift und Papier noch ein weiteres Jahr aus…