Prof. Dr. P. Liggesmeyer
Dipl.-Inf. K. Bizik
M.Sc. K. Nehring

TU Kaiserslautern
Fachbereich Informatik
AG Software Engineering: Dependability

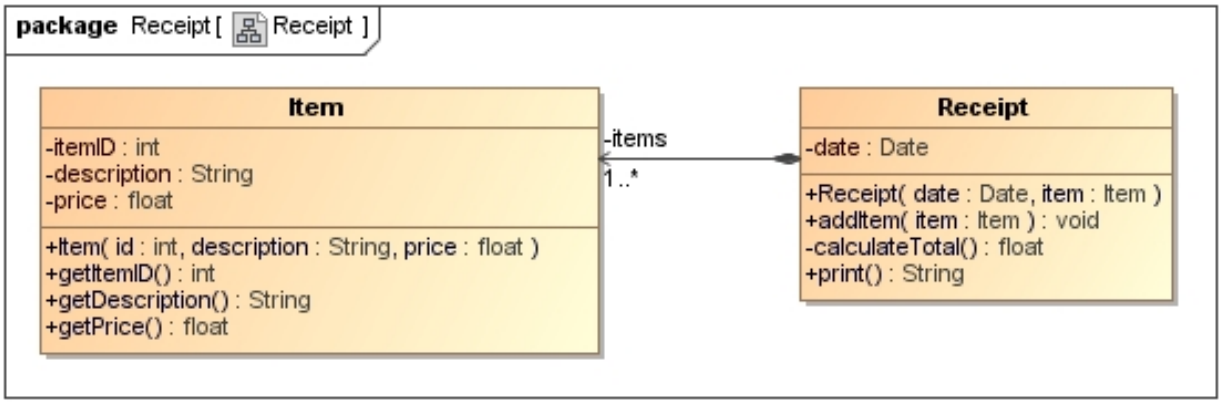
Software Entwicklung II (SS12)

Übung 4

Ausgabe: 21.05.2012
Abgabe: bis 04.06.2012, 09:30 Uhr im Übungskasten Geb. 32, 4. Stock
Hinweis: Geben Sie nur eine Lösung pro Gruppe ab und versehen Sie diese mit einem Deckblatt, welches Ihre Gruppennummer, Ihren Übungstermin (inkl. Name des Tutors), sowie die Namen aller Gruppenmitglieder enthält.

Aufgabe 1: (7 Punkte) Klassendiagramm umsetzen

Sie erhalten die Aufgabe, einen Prototyp für ein neuartiges Kassensystem in Java umzusetzen. Vom Designer erhalten Sie lediglich eine E-Mail mit dem Hinweis „Bon Drucker“ und das folgende Klassendiagramm



Setzen Sie das Klassendiagramm in Java-Code um.

Hinweis: Implementieren Sie auch die Methoden!

Aufgabe 2: (12 Punkte) Klassendiagramm aus Java-Code erstellen

Die Firma LearnIT führt Schulungen im Bereich UML und Programmiersprachen durch. Der unten stehende Code gehört zu einem Schulungsbeispiel, jedoch ist das dazugehörige Klassendiagramm unauffindbar. Des Weiteren ist gerade keine Lizenz für das Reverse-Engineering Tool verfügbar, so dass Sie den Code selbständig analysieren und das Klassendiagramm daraus erstellen müssen.

Datei BreedingDuck.java

```
package duck;
public class BreedingDuck extends Duck {

    public BreedingDuck() {
        fly = new WingFly();
```

```
        sound = new QuackSound();
        colour = "brown";
        size = 2;
    }

    @Override
    public String type() {
        return "I am very gracious.";
    }
}
```

Datei DecorDuck.java

```
package duck;
public class DecorDuck extends Duck {

    public DecorDuck() {
        fly = new NoFly();
        sound = new NoSound();
        colour = "shining brown";
        size = 2;
    }

    @Override
    public String type() {
        return "Don't move, don't talk, don't think, "
            + "don't worry, everything's just fine!";
    }
}
```

Datei Duck.java

```
package duck;
public abstract class Duck {

    protected String colour;
    protected int size;
    protected ISound sound;
    protected IFly fly;

    public abstract String type();

    public void sound() {
        sound.makeNoise();
    }

    public void fly() {
        fly.fly();
    }

    public String getColour() {
        return colour;
    }

    public int getSize() {
        return size;
    }
}
```

Datei IFly.java

```
package duck;
public interface IFly {
    void fly();
}
```

Datei ISound.java

```
package duck;
public interface ISound {
    void makeNoise();
}
```

Datei NoFly.java

```
package duck;
public class NoFly implements IFly {

    @Override
    public void fly() {
        // cannot fly
    }
}
```

Datei NoSound.java

```
package duck;
public class NoSound implements ISound {

    @Override
    public void makeNoise() {
        // makes no sound!
    }
}
```

Datei PirateDuck.java

```
package duck;
public class PirateDuck extends Duck {

    public PirateDuck() {
        fly = new WingFly();
        sound = new QuackSound();
        colour = "black and brown";
        size = 3;
    }

    @Override
    public String type() {
        return "Be ye lookin' fer trouble, eh?";
    }
}
```

Datei QuackSound.java

```
package duck;
public class QuackSound implements ISound {
```

```
    @Override
    public void makeNoise() {
        System.out.println("Quack");
    }
}
```

Datei QuiekSound.java

```
package duck;
public class QuiekSound implements ISound {

    @Override
    public void makeNoise() {
        System.out.println("Quiek");
    }
}
```

Datei RocketBoosterFly.java

```
package duck;
public class RocketBoosterFly implements IFly {

    @Override
    public void fly() {
        System.out.println("Whoooosh");
    }
}
```

Datei RubberDuck.java

```
package duck;
public class RubberDuck extends Duck {

    public RubberDuck() {
        fly = new NoFly();
        sound = new QuiekSound();
        colour = "black";
        size = 1;
    }

    @Override
    public String type() {
        return "I'll never sink!";
    }
}
```

Datei SuperDuck.java

```
package duck;
public class SuperDuck extends Duck {

    public SuperDuck() {
        fly = new RocketBoosterFly();
        sound = new QuiekSound();
        colour = "red and blue";
        size = 3;
    }
```

```
    @Override
    public String type() {
        return "Ultra-Awesome-SuperDuck";
    }
}
```

Datei WingFly.java

```
package duck;
public class WingFly implements IFly {

    @Override
    public void fly() {
        System.out.println("Flap flap");
    }
}
```

Aufgabe 3: (12 Punkte) OOA/OOD

Im Folgenden betrachten wir ein Flugsystem. Dieses Flugsystem enthält Fluglinien und Flüge. Jede Fluglinie beschäftigt eine Menge von Angestellten, führt Flüge durch und kooperiert mit anderen Fluglinien. Angestellte sind entweder Piloten oder Mitarbeiter des Bodenpersonals. Jeder Flug wird von einem bis maximal drei Piloten der Fluglinie geflogen und hat einen Abflug- und Zielflughafen.

a) Beschreiben Sie den obigen Sachverhalt mit der UML durch ein Klassendiagramm mit geeigneten bidirektionalen oder unidirektionalen Assoziationen.
Hinweis: Geben Sie keine Attribute und Methoden an!
b) Fügen Sie die Kardinalitäten und eventuell Rollennamen zu den Assoziationsenden hinzu.