Semesterschlussprüfung

Einführung in Software Engineering

Wintersemester 2000/2001

Name:	
Vorname:	
Matrikelnummer:	

- Für jede Aufgabe ein neues Blatt verwenden.
- Bitte auf jedem Blatt Name und Vorname angeben.
- Bitte leserlich schreiben.
- Wenn Sie fertig sind, bitte alle Blätter (inkl. Aufgabenblätter) ins Couvert legen; das Couvert mit Namen, Vornamen, Matrikelnummer anschreiben und abgeben.

Viel Erfolg

Aufgabe 1 : “Software Engineering” (8 Punkte)

Beantworten Sie die folgenden Fragen:

a) Welche Rolle spielt Prototyping in der Software-Entwicklung? Geben Sie zwei Beispiele an. (1 Punkt)

b) Erklären Sie den Unterschied zwischen *Coupling* und *Cohesion*. Welche Rolle spielen sie was die Qualität von Software betrifft? (1 Punkt)

c) Wie vereinfacht die Verwendung von Polymorphismus die Anpassung von Software an sich verändernde Benutzeranforderungen? (1 Punkt)

d) Ist die Anzahl Zeilen (lines of code) eines Software-Produktes ein gutes Mass für die Beurteilung von Softwarekomplexität? Begründen Sie. Fällt Ihnen eine weitere Metrik ein, die die Komplexität von Software noch besser erfassen kann? (1 Punkt)

e) Begründen Sie, weshalb es von Vorteil ist, das Qualitäts-Management eines Software-Projektes vom Projekt-Management zu trennen. (1 Punkt)

f) Erklären Sie den Unterschied zwischen funktionalen (*functional*) und nicht-funktionalen (*non-functional*) Requirements. (1 Punkt)

g) Nachfolgend sehen Sie einen Screenshot von **Pizza Designer**. Dieses Programm soll in einer Pizzeria verwendet werden. Am Eingang der Pizzeria steht ein Computer, auf dem **Pizza Designer** von einem Angestellten bedient wird. Der Kunde gibt dem Angestellten seine Wünsche bekannt, der sie wiederum in **Pizza Designer** eingibt. Wird ein Order ausgeführt, wird eine e-mail in die Küche geschickt, damit der Pizzaiolo weiss, was er zu tun hat. **Pizza Designer** befindet sich im Anfangsstadium, und Sie werden jetzt als GUI Consultant hinzugezogen um Tipps zu geben.
Machen Sie eine möglichst vollständige Liste aller Bloopers (GUI Design Fehler), die Ihnen auffallen. (2 Punkte)

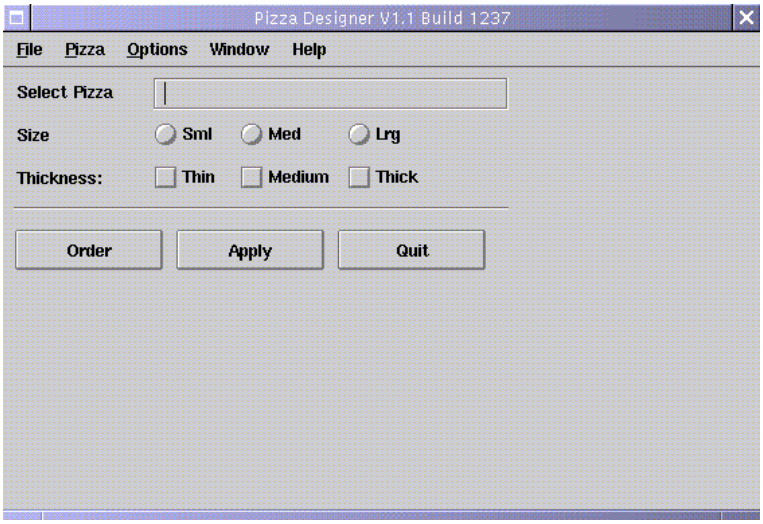

Figure 1: Das Hauptfenster von Pizza Designer.

Aufgabe 2 : “Ein UML Editor” (10 Punkte)

Auf dem beigelegtem Blatt (Rückseite auch beachten!) sehen Sie einen Teil des Codes eines graphischen Editors, der für UML Diagramme gedacht ist. Der Editor ist zur Zeit in einem rudimentären Zustand und sollte erweitert werden. Wir haben im Beispielscode zum Teil bewusst auf die “Inhalte” der Methoden (method bodies) verzichtet um Ihnen das Lesen zu erleichtern. Die Kommentarfragmente die Sie im Code sehen stehen für die Funktion des konkreten Codes der an diesen Stellen steht.

Tipp: Lesen Sie zuerst den Code, dann die gesamte (!) Aufgabe und beantworten Sie dann die einzelnen Fragen.
Merke: Der untenstehende Code ist nicht komplett. Wir haben auf unnötige Details verzichtet um die Lesbarkeit zu vergrössern.

a) Zeichnen Sie ein UML-Diagramm des Codes. Achten Sie insbesondere darauf, **möglichst komplett** die Implementation im Diagramm zu widerspiegeln. (3 Punkte)

b) Wir wollen die Funktionalitäten des Editors erweitern um auch sogenannte ”Notes” darstellen koennen. Eine Note ist eine rechteckige Figur, die an der Ecke rechts oben ”Eselsohren” hat (siehe auch ESE Script Seite 121), die einen Text enthält. Schreiben Sie die dafür nötige Klassendefinition (möglichst in Java code) inklusive Attribute und Methoden. (2 Punkte)

c) Der Editor versteht zur Zeit nur einen Typ von ”Verbindung” zwischen zwei ClassViews, wir wollen nun den Editor um einige Funktionalitäten erweitern. Die Beschreibung fuer die neuen Verbindungen sind die Folgenden (für graphische Darstellungen dieser Connections siehe auch ESE Script Seite 118).

- Constraint: ist meistens “annotiert”, besitzt also unter Umständen eine Note, die neben dieser Verbindung dargestellt wird.
- Dependency: ist als gestricheltes Pfeil dargestellt, und drueckt eine Abhängigkeit aus zwischen zwei Figuren. Sollte auf die Richtung gefragt werden können.
- Inheritance: drückt eine Vererbungsrelation zwischen zwei Klassen aus. Sollte auf die Richtung gefragt werden können.
- Aggregation: wird mit leerem “Diamantenkopf” dargestellt und drückt eine Aggregationsrelation (“a besteht aus b”) aus.
- Composition: wird mit ausgefülltem “Diamantenkopf” dargestellt und drückt eine Kompositionsrelation (“a entält b”) aus.

Erweitern Sie die Klassenhierarchie um die neuen Klassen aus Aufgabe b) und c).
Beachten Sie dabei, dass das System in der Zukunft massive Veränderungen erfahren soll, versuchen Sie also eine möglichst elegante und flexible Struktur zu erreichen, ohne dabei zu übertreiben. Zeichnen Sie das resultierende UML Diagramm aller Klassen. Überlegen Sie sich welche Methoden und Attribute die neuen Klassen haben müssen, um den obigen Requirements zu entsprechen. Für Klassen die keinen Unterschied aufweisen zu Aufgabe a) ist es nicht nötig die Attribute und Methoden anzugeben.

Als Resultat wird ein komplettes UML-Diagramm mit Attributen und Methoden erwartet. (5 Punkte)

Vorbemerkung zum Code: In diesem Code machen wir zum Teil Gebrauch von den reflexiven Möglichkeiten von Java, i.e. eine Klasse (Class) weiss welche Attribute (Fields) und Methoden (Methods) sie implementiert. In UML gibt es abweichende Namenskonventionen: eine Methode wird in diesem Fall Operation gennant.