Software Engineering

Department of Computer Science
Software Technology Group
Dr. Michael Eichberg, Leonid Glanz, Sven Amann

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Übungszettel 09: Generelle Informationen zur Abgabe

Abgabefrist: 21.01.2016, 08:00 Uhr
(Alle Lösungen müssen bis zu den jeweiligen Zeitpunkten eingereicht sein)

Erzeugen Sie aus den Lösungen aller Teilaufgaben, die keinen Quelltext erfordern, ***eine*** *Datei „****solution_xxx_ex09.pdf****". Fügen Sie für xxx Ihre Gruppennummer ein. Gliedern Sie die Abgabedatei den Teilaufgaben entsprechend.* Achten Sie auf korrekte Rechtschreibung und Grammatik. Die Tutoren sind angewiesen chaotische, unverständliche oder fehlerüberhäufte Lösungen zu ignorieren.

Alle Lösungen sind in Moodle einzustellen. Abgaben per E-Mail oder auf anderem Weg werden nicht berücksichtigt. Beachten Sie unbedingt die in der Übung vorgestellten Namenskonventionen. Ihre Abgabe wird nicht bewertet, wenn der Tutor die Lösungen nicht finden oder nicht zuordnen kann.

Die Korrekturen werden ebenfalls in Moodle eingestellt. Wir versuchen dies im Normalfall eine Woche nach Abgabe der Übung zu schaffen. Wenn Sie Fragen bzgl. der Korrekturen haben, schreiben Sie dem korrigierenden Tutor eine E-Mail.

Aufgabe 1: Bewertung eines Designs durch Betrachtung der Kopplung (3 Punkte)

Analysieren Sie im Folgenden die Datei *JarInputStream.java*. Die Analyse ist sehr einfach und kann manuell erfolgen.

a) **Kopplung (2 Punkte)**
Die Kopplung einer Klasse ist die Anzahl der Klassen von denen die Klasse abhängt. Bestimmen Sie die Kopplung der Klasse *JarInputStream*. Geben Sie hierzu alle Klassen an mit denen die Klasse gekoppelt ist.
Hinweis: *Bei der Kopplung müssen nur die direkten Abhängigkeiten betrachtet werden.*

b) **Softwaredesign und Kopplung (1 Punkt)**
Gegeben der folgende Code:

```
class SingleFileWriter {

	public static FileWriter getFileWriter(String fileName) throws IOException{
		return new FileWriter(fileName,true);
	}
}
class Logger {

// …

	public void log(String message) throws Exception{
		FileWriter fw=SingleFileWriter.getFileWriter("log.txt");
		fw.write(message);
		fw.flush();
	}
}
```

Hat *log(String message)* hier eine Abhängigkeit zu *IOException*?

Aufgabe 2: Bewertung eines Designs durch Betrachtung der Kohäsion (5 Punkte)

a) Verständnis der Metrik Lack of Cohesion in Methods (LCOM) (2 Punkte)
Nach LCOM hat eine Klasse hohe Kohäsion, wenn alle Methoden möglichst ähnlich bzgl. ihrer Feldvariablennutzung sind. Daher definieren wir die Ähnlichkeit mit **LCOM=1-Summe(MF)/M*F**, wobei M die Anzahl an Methoden ist (einschließlich Konstruktoren, Getter, Setter, Events, add und remove Methoden). F stellt die Anzahl der Instanzfelder einer Klasse dar. Mit MF wird die Anzahl der Methoden beschrieben, die auf ein Feld einer Klasse zugreifen. Zuletzt stellt Summe(MF) die Summe aller Methodenzugriffe auf Instanzfelder dar. Somit gilt, dass Klassen mit einer hohen Kohäsion einen Wert von LCOM=0 erreichen.
Berechnen Sie den LCOM Wert der Klassen *LinkedList* und *Node*.
Hinweis: *Für MF sollen Felder pro Methode nur einmal gezählt werden.*

b) Bewertung der LCOM Metrik (2 Punkt)
Beschreiben Sie die Aussagekraft der LCOM Metrik und welche Methoden in den Klassen *LinkedList* und *Node* zu problematischen Aussagen der Kohäsion führen können.

c) Bewertung der Kohäsion (1 Punkt)
Bewerten Sie unabhängig vom errechneten Ergebnis der Metrik, ob die Klassen *LinkedList* und *Node* in Ihrer Implementierung eine hohe oder eine niedrige Kohäsion aufweisen. Begründen Sie Ihre Bewertung.

Aufgabe 3) Kopplung und Kohäsion (2 Punkte)

Gegeben sei eine Klasse A, die von einer Klasse B und einer Klasse C abhängt. Wenn es jetzt ***einfach möglich*** ist die Klasse A in zwei Klassen A1 und A2 aufzuspalten so dass, A1 nur noch B und A2 nur noch C benutzt.
Wie würden Sie dann die Kopplung und Kohäsion vor und nach der Änderung bewerten?