Prüfung 10 August 2011, Fragen und Antworten - (SS 2011)

Einführung in die Softwaretechnik (IN0006) (Technische Universität München)

Technische Universität München

Fakultät für Informatik
Lehrstuhl für Informatik 19

Klausur zur Veranstaltung „Einführung in die Softwaretechnik“

Lehrstuhl für Informatik 19 (sebis), 10. August 2011, Sommersemester 2011

Nachname:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Wichtige Hinweise

- Füllen Sie auf diesem Blatt gut lesbar die obigen Felder (Nachname, Vorname etc.) aus.
- Notieren Sie auf jeder Seite des Arbeitspapiers gut lesbar Ihren Namen und die laufende Seitennummer!
- Es stehen 120 Minuten Bearbeitungszeit zur Verfügung. Maximal zu erringen sind 72 Punkte.
- Für korrekte Lösungsansätze werden auch dann Punkte vergeben, wenn das Endergebnis fehlerhaft ist oder fehlt. Erläutern Sie daher Ihre Lösungswege möglichst genau!
- Bei einem Täuschungsversuch wird die Klausur mit 5,0 bewertet.

Erlaubte Hilfsmittel

- Ein handbeschriebenes DIN A4 Blatt.

Technische Universität München

sebis

Fakultät für Informatik
Lehrstuhl für Informatik 19

Aufgaben

Teil 1: Wissensfragen

1. **Erläutern Sie die Begriffe *White-Box-Test* und *Black-Box-Test* und grenzen Sie sie voneinander ab. Gehen Sie dabei auch darauf ein, wann und wie diese Tests erstellt werden und welche Ziele dabei verfolgt werden. (5 Punkte)**
2. **Nennen Sie die sechs Qualitätsattribute des ISO 9126 Qualitätsmodells. Beschreiben Sie für zwei dieser Attribute, wie sich diese weiter unterteilen lassen. (5 Punkte)**
3. **Erläutern Sie kurz die Begriffe *Refactoring* und *Regressionstest*, sowie die Bedeutung von Regressionstests für das Refactoring. (3 Punkte)**
4. **Erläutern Sie, was beim Architekturentwurf mit „geringer Kopplung“ und “hoher Kohäsion“ gemeint ist. (2,5 Punkte)**
5. **A) Grenzen Sie Pflichten- und Lastenheft voneinander ab. (2 Punkte)**

   B) Nennen Sie sechs Bestandteile eines Lastenhefts, so wie in der Veranstaltung vorgestellt, und erläutern Sie drei davon. (6 Punkte)
6. **Ein Vorgehensmodell beschreibt systematische, ingenieurmäßige und quantifizierbare Vorgehensweisen, um Aufgaben einer bestimmten Klasse wiederholbar zu lösen.**

   A) Nennen Sie die fünf Phasen des Wasserfallmodells, so wie in der Veranstaltung vorgestellt. (2,5 Punkte)

   B) Erläutern Sie die Idee von agiler Softwareentwicklung am Beispiel von eXtreme Programming (XP). Gehen Sie dabei auf die Werte agiler Softwareentwicklung sowie auf die Regeln und Prinzipien von XP ein. (4 Punkte)

Teil 2: Verständnis- und Modellierungsfragen

1. Reverse-Engineering

Gegeben ist folgendes Code-Fragment:

```
1  public class Rechner {
2      Algorithmus algorithmus;
3      int ergebnis;
4
5      public void setAlgorithmus(Algorithmus algorithmus) {
6          this.algorithmus = algorithmus;
7      }
8
9      public void rechne(int eingabe) {
10         ergebnis = algorithmus.rechneEsAus(eingabe);
11     }
12 }
13
14 public abstract class Algorithmus {
15     public abstract int rechneEsAus(int eingabe);
16 }
17
18 public class OptimierterAlgorithmus extends Algorithmus {
19     public int rechneEsAus(int eingabe) {
20         return 5; // schon fertig!
21     }
22 }
23
24 public class RekursiverAlgorithmus extends Algorithmus {
25     public int rechneEsAus(int eingabe) {
26         if (eingabe == 0) {
27             return 4711;
28         }
29         else {
30             return this.rechneEsAus(eingabe - 1);
31         }
32     }
33 }
34
35
```

```
36  public class Main {
37      public static void main(String[] args) {
38          new Main();
39      }
40      public Main() {
41          Rechner rechner = new Rechner();
42          Algorithmus rekAlg = new RekursiverAlgorithmus();
43          rechner.setAlgorithmus(rekAlg);
44          rechner.rechne(1);
45          Algorithmus optAlg = new OptimierterAlgorithmus();
46          rechner.setAlgorithmus(optAlg);
47          rechner.rechne(1);
48      }
49  }
```

A) Erstellen Sie zu diesem Code-Fragment ein implementierungsnahes Klassendiagramm, das alle im Code enthaltenen Klassen, Attribute, Konstruktoren und Methoden sowie die Assoziationen mit entsprechenden Rollennamen, Multiplizitäten und der Navigierbarkeit beinhaltet. Sichtbarkeiten von Attributen und Methoden müssen nicht modelliert werden. Kennzeichnen Sie abstrakte Klassen mit dem Zusatz „{abstract}“. (6 Punkte)

B) Welches Entwurfsmuster wird hier verwendet? Benennen Sie für jede Klasse, die Teil des Musters ist, die Rolle, die sie in dem Muster einnimmt (die Rollennamen sind die Klassennamen aus der allgemeinen Beschreibung des Musters). (4 Punkte)

C) Erstellen Sie ein Sequenzdiagramm, das die Interaktionen der Objekte ausgehend vom Aufruf des Konstruktors `Main()` beschreibt. Beachten Sie dabei, dass Ihr Diagramm den konkreten Ablauf beschreiben soll, der sich bei der Ausführung des Programms ergibt. Vergessen Sie nicht, Parameter und Rückgabewerte anzugeben, wo dies sinnvoll ist. (14 Punkte)

2. Konzeptuelle Modellierung (18 Punkte)

Erstellen Sie ein konzeptuelles Klassendiagramm für die Videothek EIST-Movies. Modellieren Sie dabei die Konzepte aus folgender Beschreibung so genau wie möglich:

Das Softwaresystem der Videothek EIST-Movies soll alle in der Videothek ausleihbaren Filme, sowie die momentan verliehenen Exemplare erfassen. Außerdem können Kunden über das System Filme suchen und ausleihen. Ein Film hat einen Titel und eine Spieldauer, die jeweils in ganzen Minuten angegeben ist. Außerdem ist pro Film die Tagesgebühr für das Ausleihen festgelegt. Zu einem Film kann die Videothek mehrere Exemplare besitzen. Es

ist auch möglich, dass Filme im System erfasst sind, zu denen noch keine Exemplare vorrätig sind. Ein Exemplar ist entweder eine DVD oder eine BluRay-Disc. Zu BluRays ist jeweils angegeben, ob der Film in 3D abgespielt werden kann.

Im System werden außerdem alle Kunden der Videothek verwaltet. Jeder Kunde hat eine eindeutige Kundennummer, einen Vornamen und Nachnamen, sowie eine Anschrift, die aus Straße, Postleitzahl und dem Ort besteht. Wenn ein Kunde einen Film ausleiht, wird erfasst, welches Exemplar er ausgeliehen hat, wann er es ausgeliehen hat und wann die Leihfrist endet. Zusätzlich wird der Preis gespeichert, den der Kunde für das Ausleihen dieses Exemplars bezahlen muss. Ein Kunde kann verschiedene Filme gleichzeitig ausgeliehen haben und die Filme müssen dabei nicht alle für denselben Zeitraum ausgeliehen werden. Da die Videothek großen Wert auf Datenschutz legt, existieren keine Aufzeichnungen darüber, welche Filme in der Vergangenheit entliehen wurden, sondern nur darüber, welche Filme momentan entliehen sind.

Wenn sich ein Kunde neu bei der Videothek registriert, kann er, wenn er möchte, einen Lieblingsschauspieler angeben. Er erhält dann Rabatt auf alle Filme, in denen dieser Schauspieler mitspielt. Zu Schauspielern wird im System nur der volle Name gespeichert.

Um den Kunden das Durchsuchen der Filme zu erleichtern, ist zu den Filmen, die Fortsetzungen haben, hinterlegt, welcher Film das jeweils ist. So ist zum Beispiel der Film „Transformers 2“ die Fortsetzung von „Transformers“ und „Transformers 2“ hat die Fortsetzung „Transformers 3“. Außerdem können die Filme nach Spieldauer sortiert werden.

Hinweis: Selbstverständlich können in dem System auch Kundinnen und Schauspielerinnen verwaltet werden, dieser Umstand muss aber nicht modelliert werden, so dass in der Angabe oben durchgehend das generische Maskulinum verwendet wurde.

Klausur 2011 (Haupt)

Teil 1:

① **Black-Box-Test:** = funktionsorientierter Test: Es wird nicht die interne Struktur, sondern die Funktionalität eines Programms / Systems betrachtet. Tests sollten sich möglichst aus der Anwenderdomäne ableiten, also aus den Anforderungen an das System. Die Tests können bereits vor der Programmierung erstellt werden. Dabei wird auf die Ausgaben-, Funktions- und Ausnahmenüberdeckung eingegangen (für jeden Bereich mindestens ein Test) Mögliche Ansätze: Grenzwertanalyse, Äquivalenzklassenbildung, Statische Tests, Raten von Fehlern (z.B. aus Erfahrung)

Ziel: Anforderungen alle erfüllt? → prüfen

White-Box-Test: = strukturorientierter Test: Es wird die interne Struktur betrachtet. Der Code muss hierzu bereits geschrieben sein, sodass sich die Testfälle von diesem ableiten können. Hierbei wird mittels Überprüfung der Abdeckung des Kontrollflusses bzw Datenflusses Tests erstellt.

Ziel: Spezifikationserfüllung, Code-Coverage

② **ISO 9126**

- Funktionalität
  - Angemessenheit
  - Richtigkeit
  - Interoperabilität
  - Sicherheit
- Zuverlässigkeit
  - Reife
  - Robustheit
  - Fehlertoleranz
  - Wiederherstellbarkeit
- Effizienz
- Benutzbarkeit
- Änderbarkeit
- Übertragbarkeit

③ **Refactoring** = Strukturverbesserung unter Beibehaltung des Programmverhaltens

Regressionstest = Test, dass bereits behobene Fehler auch behoben bleiben
↳ Nach Modifikationen, Refactoring

↳ Nachdem Refactoring betrieben wurde muss mittels Regressionstest sichergestellt werden, dass alle Tests trotz Strukturverbesserung erfolgreich verlaufen.

④ Geringe **Kopplung** bei Architekturentwurf bedeutet, dass möglichst wenige Verbindungen zwischen den Komponenten bestehen. Dadurch soll die Komplexität reduziert werden.
Hohe **Kohäsion** beim Architekturentwurf meint die klare Trennung von Verantwortlichkeiten im System → eine enge Bindung verwandter Elemente ist das Ziel.

⑤ A) **Pflichtenheft**: Von Arbeitnehmer, Detaillierte Anforderungs- & Systemspezifikation, präzise, konkret Lösungsorientiert, detaillierte Projektplanung & Risikobetrachtung
Lastenheft: Von Arbeitgeber, Ziel, Strategie, Rahmenbedingung, abstrakt, problemorientiert, Projektplanung & Risikoanalyse

B) Zielbestimmung: Das Ziel, welches das Produkt erreichen soll, wird hier beschrieben.
Produkteinsatz: Legt fest, welche Akteure mit dem System arbeiten sollen
Produktumgebung: beschreibt den Nutzungskontext des Produkts & benachbarte Systeme
Produktfunktionen
Produktdaten
Produktleistungen

⑥ A) **Wasserfallmodell**:

Analyse & Anforderungsdefinition → Entwurf → Implementierung → Test → Einsatz & Wartung

B) **Agile Softwareentwicklung**

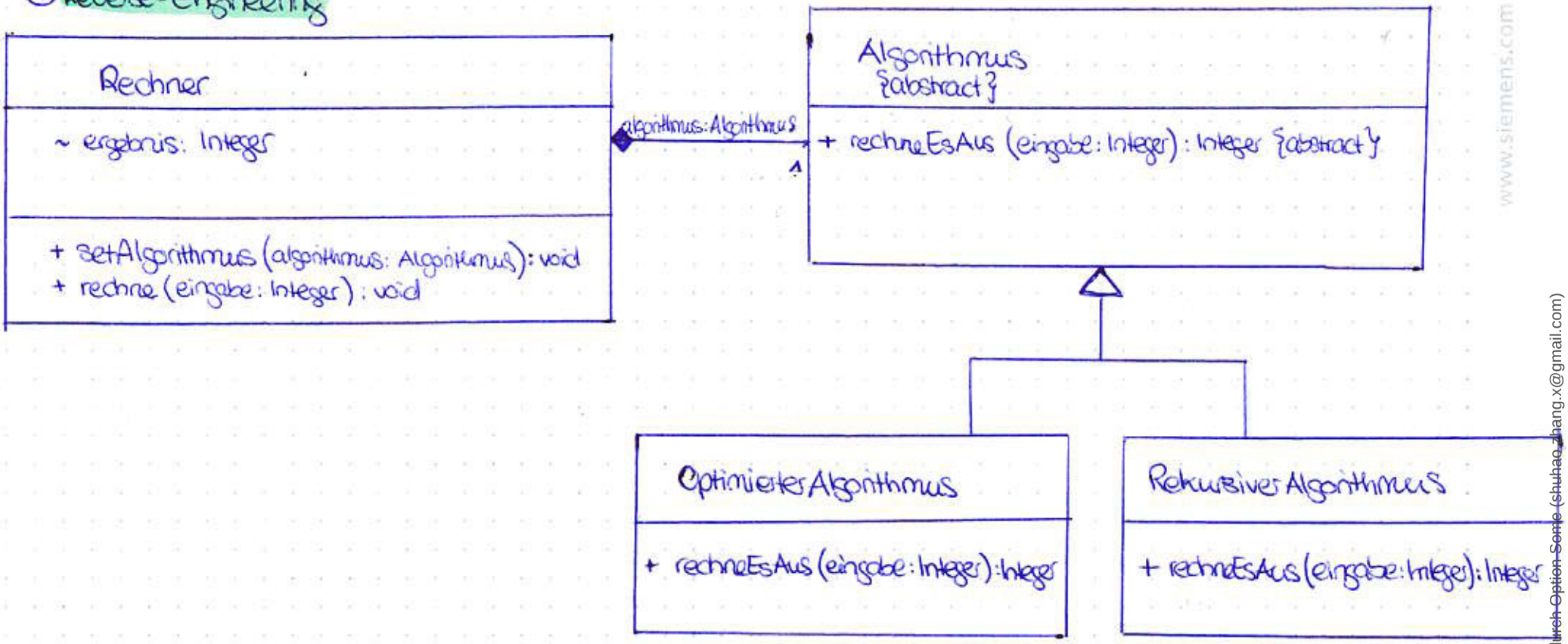
Werte: Individuen & Interaktionen sind wichtiger als Prozesse & Werkzeuge
Funktionierende Software im Fokus
Zusammenarbeit mit Kunden wichtig
Reagieren auf Veränderungen wichtiger als Planverfolgung

XP → code-zentriert, evolutionäre Prozesse: Prototyp weiter & weiter entwickeln

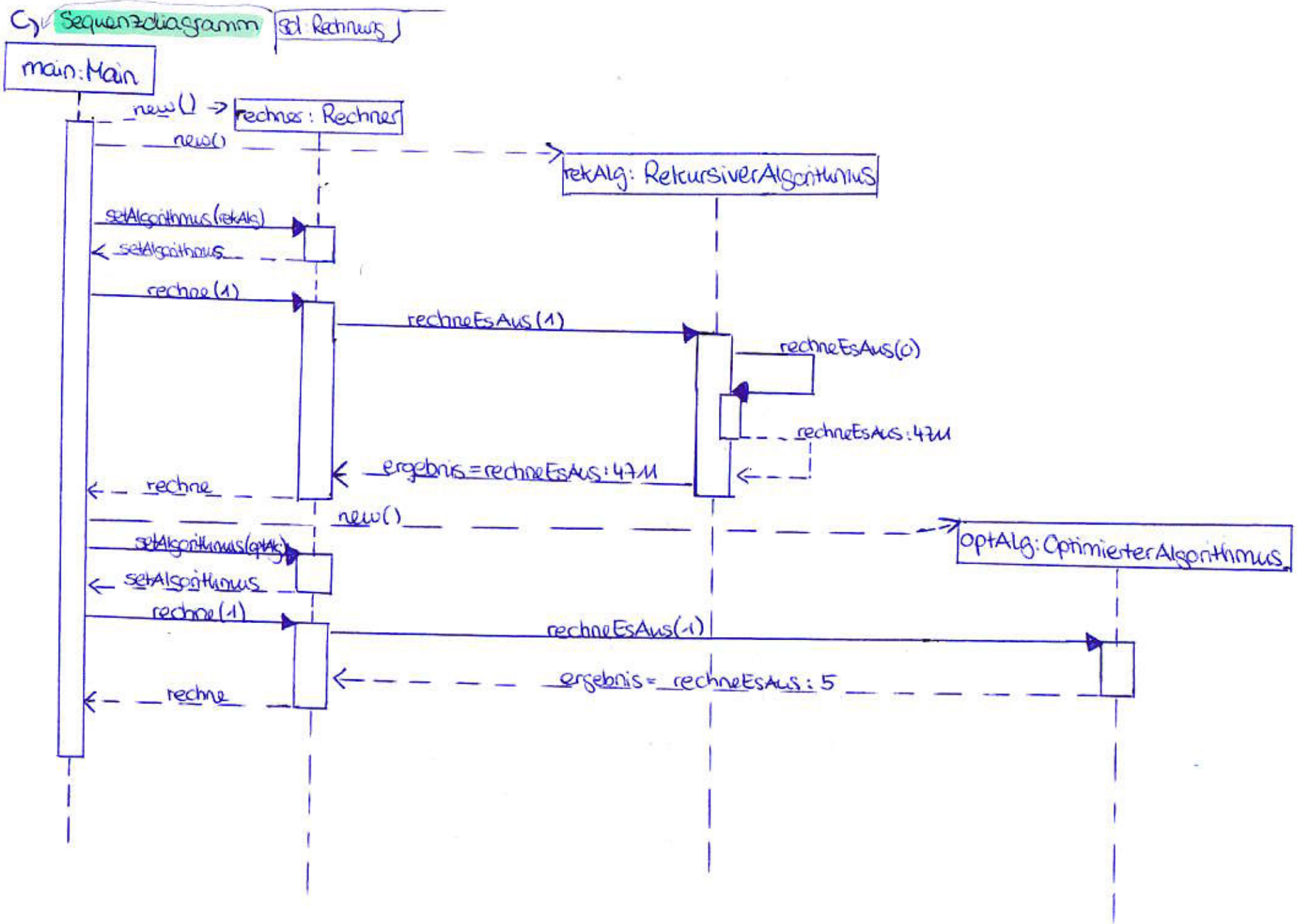
Regeln & Prinzipien:
- Test Driven Development
- Programmierung im Team
- hoher Wert auf Kommunikation (mit Kunden)
- einfache, schnell änderbare Architektur

Teil2: ① Reverse-Engineering

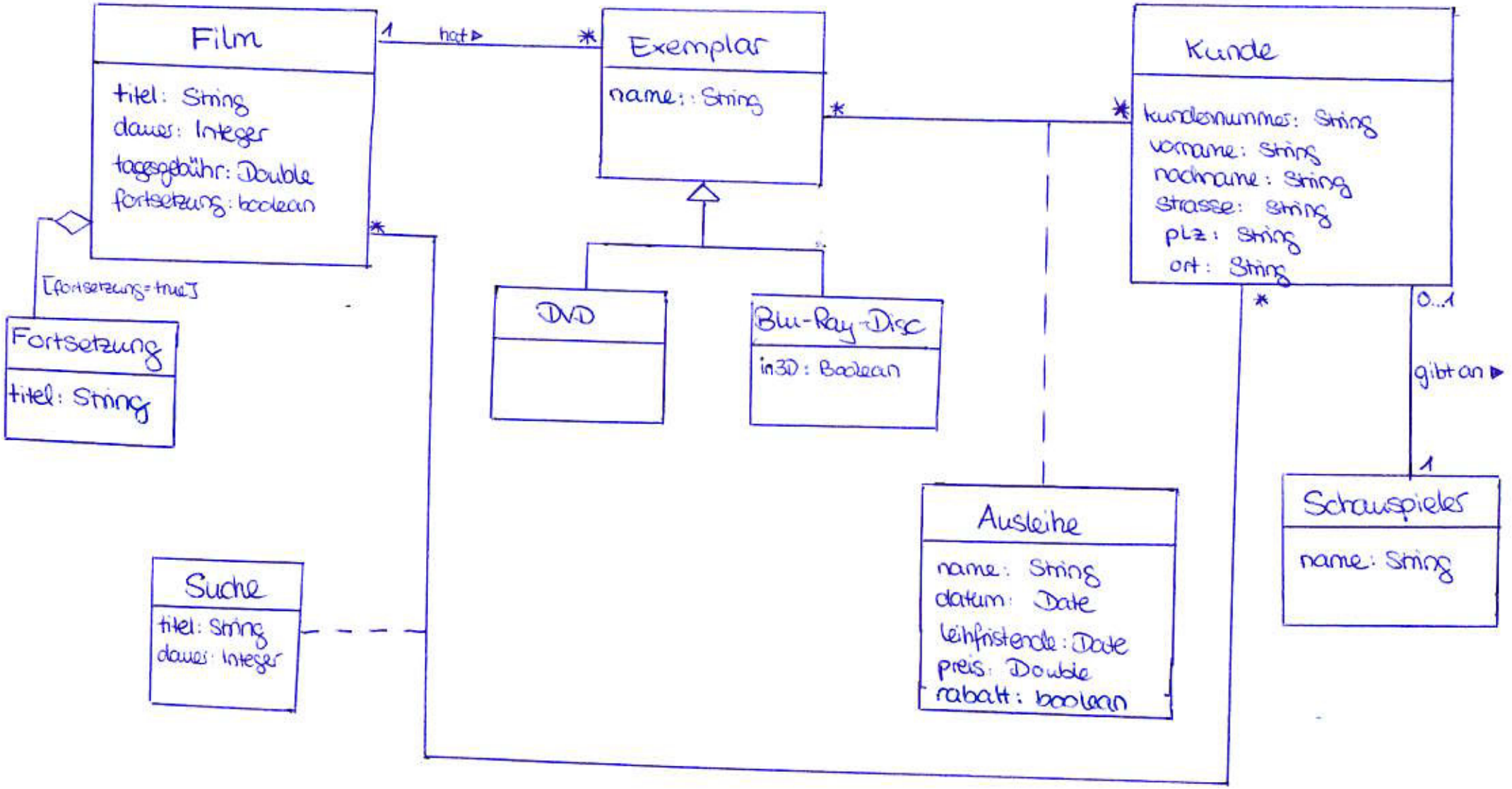
A)



B) Das Entwurfsmuster „Strategie" wurde hier verwendet.
Rechner ist dabei die Klasse Context und die Methode rechne() entspricht der Methode contextMethod(). Algorithmus entspricht der abstrakten Klasse Strategy. Die Subklassen von Algorithmus entsprechen den Subklassen von Strategy. Jede Klasse hat die Methode rechneEsAus(), die mit der algorithmicMethod() des Strategie Entwurfsmusters übereinstimmt. Durch dieses Entwurfsmuster können viele verwandte Klassen gekapselt werden, jede Subklasse von Algorithmus hat einen anderen Algorithmus zum Berechnen (intern), sie sind aber austauschbar, erweiterbar und leicht veränderbar.



(2.) konzeptuelles Klassendiagramm



Anmerkung: Umsetzung der Fortsetzung → unsicher