Professionelle Softwareentwicklung

Klausurvorbereitung

Anmerkungen

Die folgende Auswahl an Fragen soll Ihnen einen Eindruck geben, welche Art von Fragen Sie in der Klausur erwarten können. Es ist **keine** Aufzählung aller Themen, die in der Klausur vorkommen.

Inhalt der Klausur sind

1. Alle Vorlesungen
2. Alle Übungen
3. Alle Projekte und Blätter
4. Alle Videos (ausser, die Videos die explizit ausgeschlossen sind)

Der Fragenkatalog spiegelt **nicht** den Umfang einer Klausur wider. Die Fragen stammen zum überwiegenden Teil aus dem vergangenen Semester, es sind aber Fragen entfernt, geändert oder neu hinzugefügt worden. Neue Fragen finden Sie ganz unten.

Fragen

- Mit Hilfe welcher Technik kann man den gedanklichen Fokus vom Produkt auf den Prozess lenken um Prokrastination zu überwinden?
- Nennen Sie ein Beispiel für eine Verletzung des SRP
- Gibt es Fälle, in denen es egal ist, wie sauber oder ungetestet der Code ist? Wenn ja, geben Sie ein Beispiel.
- Eine Klasse hat ein statisches Attribut `foo`, ein nicht-statisches Attribut `bar` und eine Instanzmethode `compute`. Welche der folgenden Aussagen sind wahr?
  1. `compute` kann auf `foo` und `bar` zugreifen
  2. `compute` kann auf `foo`, aber nicht auf `bar` zugreifen
  3. `compute` kann auf `bar`, aber nicht auf `foo` zugreifen
- Für welche Art von Klassenbeziehung gilt das Liskov'sche Substitutionsprinzip?
- Welche der folgenden Aussagen ist korrekt?
  1. Die Klasse eines Objekts kann nicht geändert werden
  2. Der deklarierte Typ kann die Klasse des Objektes sein oder jeder Supertyp
  3. Deklarierte Typen kann man mit einem Typecast ändern
  4. Deklarierte Typen kann man ohne Typecast abschwächen
- Angenommen, wir haben eine Klasse `LoginService` mit einer Methode `createSession(User user)`. Was darf die Methode gemäß dem Gesetz von Demeter machen?
  1. void Methoden auf `user` aufrufen
  2. Methoden auf `user` aufrufen, die einen Rückgabewert haben
  3. Methoden auf `user` aufrufen, die einen Rückgabewert haben und dann auf dem Rückgabewert Methoden aufrufen

- Was ist das Interface Segregation Prinzip und warum ist es wichtig das Prinzip einzuhalten?
- Wie lautet die Aussage des Dependency Inversion Prinzips?
- Warum hilft Dependency Inversion beim Testen von Anwendungen?
- Was ist der Unterschied zwischen einer Java Bean und einer Spring Bean?
- Welche drei Formen von Dependency Injektion haben wir in der Vorlesung gesehen. Erläutern Sie kurz die Vor- und Nachteile der einzelnen Injektionsarten.
- Ein Entwickler sagt zu Ihnen: "Meine Software ist korrekt, denn ich habe Tests geschrieben. Ich habe sogar eine Codeabdeckung von 100% mit IntelliJ erreicht." Was halten Sie von dieser Aussage?
- Was ist bei Assertions mit Gleitkommazahlen zu beachten?
- Wir haben einen Satz von Tests. Um herauszufinden, ob die Tests durchlaufen schauen wir auf die Ausgabe, wenn dort in jeder Zeile "ok" steht, dann ist alles in Ordnung. Gegen welche der FIRST Prinzipien verstoßen wir damit?
- Welche besonderen Regeln gelten für Mocks, aber nicht für Stubs?
- Wir benötigen eine Instanz der Klasse `Foo`. Bei jedem Aufruf der Instanzmethode `bar` aus der Klasse `Foo` muss die Rückgabe 42 sein. Schreiben Sie die zwei Zeilen, die uns eine solche Instanz erzeugen. Verwenden Sie Mockito.
- Warum sollen Tests Verhalten und keine Implementierungsdetails überprüfen?
- Bei einem Refactoring geht es darum, den bestehenden Code zu verbessern ohne dabei das Verhalten zu ändern. Folgender Code wurde testgetrieben entwickelt, es gibt Tests für `n=1` und `n=2`. Handelt es sich um ein gültiges Refactoring, obwohl sich das Ergebnis für `n=3` geändert hat?

```
// vorher
public String toRoman(int n) {
  String result = "";
  if (n==2) result += "I";
  result += "I";
  return result;
}

// nachher
public String toRoman(int n) {
  String result = "";
  for (int i=0;i<n;i++) {
    result += "I";
  }
  return result;
}
```

- Gegeben seien folgende zwei Klassen. Was wurde hier beim Thema Polymorphismus nicht verstanden?

```
public class Tier {
  private int position;

  public void bewegen(int entfernung) {
    this.position += entfernung;
  }
}

public class Fahrzeug extends Tier {

  private boolean gasGegeben;

  public void gasGeben() {
    gasGegeben = true;
  }

  public void bremsen() {
    gasGegeben = false;
  }

  public void fahren() {
    while (gasGegeben) {
      super.bewegen(10);
    }
  }
}
```

- Schreiben Sie stichpunktartig auf, was bei der Benennung von Elementen in Programmen wichtig ist.
- Was halten Sie von folgendem Codeschnippsel? Verbessern Sie den Code.

```
public double pythagoras(double a, double b) {
  double aquadrat = a*a; // a quadrieren
  double bquadrat = b*b; // b quadrieren
  return Math.sqrt(aquadrat + bquadrat);
}
```

- Methoden sollten eine einheitliche Abstraktionsebene haben. Was ist damit gemeint und warum ist das wichtig?
- Wie lang darf eine Methode sein?
  1. 5 Zeilen
  2. 50 Zeilen
  3. 500 Zeilen
  4. Egal, aber sie darf nicht mehr als eine Sache machen
- Was ist an folgender Klasse nicht gut?

```
public class ASP {
   // egal
}
```

- Warum ist die Verwendung von globalen Variablen problematisch?
- Folgender Test ist nicht optimal strukturiert. Erklären Sie, was das Problem ist und wie man es lösen könnte

```
@Test
public void checkBalance() {
  Konto k1 = new Konto();
  k1.einzahlen(100);
  assertEquals(100, k1.kontostand());
  Konto k2 = new Konto();
  k1.ueberweisen(k2,50);
  assertEquals(50, k1.kontostand());
  assertEquals(50, k2.kontostand());
}
```

- Was ist Datenabstraktion? Geben Sie ein Beispiel dafür an.
- Welche Object Calisthenics Regel ist hier verletzt und in welcher Zeile ist die Verletzung?

```
public class Schulden {
   Betrag betrag;

   public Schulden(Betrag betrag) {
     this.betrag = betrag;
   }

   public void betragHinzufuegen(Betrag betrag) {
     this.betrag.add(betrag);
   }
}
```

- Warum verletzt `berechneBrutto` nicht die Regeln der Object Calisthenics?

```
public class Geld {

  private int netto;
  private int brutto;

  public Geld(int netto) {
     this.netto = netto;
     this.brutto = berechneBrutto(netto);
  }

  private int berechneBrutto(int netto) {
    return netto * 1.19;
  }

  // ...
}
```

- Wir haben eine Webanwendung geschrieben, mit deren Hilfe Versandaufkleber in der Versandabteilung gedruckt werden. Dazu senden wir einen Request an die URL http://versand/printqueue?name=Jens&adresse=Universitaetsstr. Warum sollten wir für den Request nicht das HTTP Verb GET verwenden?
- Angenommen, wir haben folgendes Formular, das per GET Request von der URL http://www.klausur.de geholt wurde.

```
<form>
 <input name="a">
 <input name="b">
 <button type="submit">OK</button>
</form>
```

Wir tragen im ersten Textfeld "Hurra" und im zweiten Textfeld "bestanden" ein. Welche URL steht nach dem Absenden im Browserfenster (vorausgesetzt der Server antwortet mit Statuscode 200)?

- Gegeben sei folgender Controller, der aus einer Spring Boot Anwendung auf dem Server www.klausur.de stammt:

```
@Controller
@RequestMapping("/kunden")
public class KundenController {

 @GetMapping("/show")
 public void showAll() { /* ... */ }

 @GetMapping("/show/{id}")
 public void showOne() { /* ... */ }

}
```

Wie sieht ein Aufruf aus, der auf die Methode showOne gemappt wird?

- Was bedeutet es, wenn eine Methode idempotent ist? Geben Sie ein Beispiel für eine idempotente Methode, die einen Seiteneffekt hat. Sie können eine eigene (sinnvolle) Methode schreiben oder eine Methode aus der Java Standardbibilothek angeben.

Neue Fragen

- In Projekt 4 mussten in den Test oft Instanzen der Klasse `Blatt` konstruiert werden, die Instanzen der Klasse `Abgabe` beinhalten. Implementieren Sie eine Klasse `BlattBuilder`, die das Builderpattern verwendet, um Blätter für Tests zu konstruieren. Die Klasse sollte erlauben einzelne Abgaben zu konfigurieren, aber auch eine konfigurierbare Anzahl von Abgaben zu generieren.

```
// Blatt mit zwei Abgaben
Blatt b1 = new BlattBuilder().blatt(0).with(abgabe1).with(abgabe2).build();
// Blatt mit insgesamt 13 Abgaben
Blatt b2 = new BlattBuilder().blatt(1).with(12).with(abgabe3).build();
```

- Schreiben Sie eine Methode `printFirstLines`, die Dateinamen als Strings übergeben bekommt und dann den jeweiligen Dateinamen und die erste Zeile der Datei ausgibt.

Beispielaufrufe könnten so aussehen:

```
printFirstLines("moby.txt");
> moby.txt: Call me Ishmael.

printFirstLines("README.adoc","moby.txt");
> README.adoc: = Professionelle Softwareentwicklung: Blatt 2
> moby.txt: Call me Ishmael.
```

Tipp: Die Klasse `java.nio.file.Files` hat eine statische Methode mit folgender Signatur:

```
public static List<String> readAllLines(Path path) throws IOException {
    // ...
}
```

- Wir wollen eine statische Methode `boolean isLeapYear(int year)` testgetrieben entwickeln, die `true` zurückgibt, genau dann, wenn `year` ein Schaltjahr ist. Es gelten folgende Regeln (Auszug aus Wikipedia):
  - Die durch 4 ganzzahlig teilbaren Jahre sind Schaltjahre.
  - Säkularjahre, also die Jahre, die ein Jahrhundert abschließen (z. B. 1800, 1900, 2100 und 2200) sind keine Schaltjahre.
  - Schließlich sind die durch 400 ganzzahlig teilbaren Säkularjahre doch Schaltjahre.

Schreiben Sie die Tests, die notwendig sind, um die Methode testgetrieben zu entwickeln. Nach jedem Test schreiben Sie den Code auf, der durch diesen Test getrieben wurde auf. Schreiben Sie nur das Ergebnis nach dem Refactoring auf, nicht jede Zwischenform.

- In der Helper Kasse `Collections` aus der Java Runtime gibt es eine Methode mit der Signatur `public static <T extends Comparable<? super T>> void sort(List<T> list)`
  1. Erklären Sie die verwendeten Generics.
  2. Was muss man beachten, wenn man diese Methode auf einer Liste aufruft? (Tipp: Schauen Sie sich den Rückgabetyp an)
- Annotationen können Parameter in Form von Schlüssel-Wert-Paaren übergeben bekommen, z.B. `@RequestMapping(path="/kunden/liste")`. Unter bestimmten Voraussetzungen, kann man den Schlüssel weglassen. Welche Voraussetzungen sind das?
- Angenommen, wir arbeiten an einem sehr spezifischen Teil der Fachlogik einer Software. Warum ist die Zerlegung in Komponenten wichtig?
- Erklären Sie das Information Hiding Prinzip nach Parnas.
- Eine Komponente unserer Software verwendet ein sehr langsames externes System. Wir wollen sicherstellen, dass das externe System korrekt benutzt wird. Wie können wir das in einem Test sicherstellen, ohne das externe System tatsächlich aufzurufen?
- Wir wollen eine Methode `twice` implementieren, die einen Lambda-Ausdruck `f` und ein Objekt `t` vom Typ `T` als Parameter entgegennimmt. Der Lambda Ausdruck bekommt einen Parameter eines Typs `T` und gibt einen Wert vom gleichen Typ zurück. Die Methode soll den Ausdruck zweimal auf t anwenden, es soll also `f(f(t))` berechnet werden. Schreiben Sie die Funktion `twice` sowie ggf. notwendige weitere Klassen oder Interfaces.

Folgende Assertions sollen wahr sein:

```
assertThat(twice(x->x*x,4), is(256));
assertThat(twice(x->x+x,"."), is("...."));
assertThat(twice(x->x+x,3), is(12));
```

- Verwenden Sie Java Streams um die Summe der ersten `n` geraden Quadratzahlen auszurechnen.

- Nennen Sie zwei Operationen, die auf einer `ArrayList` sehr bequem durchführbar, aber auf einem normalen Array mit mehr Aufwand verbunden sind.