Übung 10

Software-Tests (Modultests, Blackbox-Test, Akzeptanztest), Verschiedenes

1. Bauen Sie in die rekursive Implementierung des einfachen `int`-basierten Fakultät-Berechnungsprogramms aus der Vorlesung ein Testmodul ein, das für Eingabewerte von -1 bis +100 den berechneten Funktionswert auf Plausibilität überprüft, indem einmal mit dem „Vorgängerwert“ und einmal mit dem „korrekten Wert“ (unter Zuhilfenahme einer `BigInteger`-Variante) verglichen wird. Protokollieren Sie die Ausgaben (Erwartungswert, berechneter Wert, OK oder ERROR).

2. Versuchen Sie, anhand eines **Blackbox** Tests mit verschiedenen Eingabe-Zeichenketten herauszufinden, was die statische Klassenmethode `public static String geheimnis(String eingabe)` aus der nur als Compilat vorliegenden Datei **Geheim.class** eigentlich tut. Sie können hierzu, um nicht für jeden Test neu compilieren zu müssen, auch wieder die Klasse `Eingabe.java` zuhilfe nehmen, wie in diesem Code-Fragment gezeigt:

```
public class BlackboxTest {
 public static void main(String[] args) {
  for(;;) { // Endlosschleife
   String s1 = Eingabe.readString("Eingabe: ");
   String s2 = Geheim.geheimnis(s1);
   System.out.println("Ausgabe: " + s2);
  }
 }
}
```

3. Implementieren Sie den in der vorigen Aufgabe herausgefundenen Algorithmus neu als JAVA-Klasse mit einer statischen Methode, analog der Klasse **`Geheim`**. Welche Methode des Software Engineering haben Sie hiermit eingesetzt?

4. Wie kann man als Softwareentwickler sichergehen, dass ein Softwareprodukt nicht beim *Akzeptanztest* durchfällt?

Die folgenden Aufgaben sind etwas umfangreicher, teilweise Wiederholung bereits bekannter Themen, und sollen Ihnen bei der Klausurvorbereitung helfen. Die Bearbeitung ist (wie immer) freiwillig:

1. Zeichnen Sie eine UML-Grafik (suchen Sie sich die passende Diagrammform aus), die verdeutlichen soll, wie die Kernphasen eines Software-Projektes mit den begleitenden Prozessen zusammenhängen.

2. Dokumentieren Sie das Ihre eigene Implementation der Klasse **`Geheim`** für Entwickler und Anwender getrennt, und

3. Überlegen Sie sich eine Lizenz für Ihr Produkt, und verweisen Sie in der Dokumentation an geeigneter Stelle darauf, wie es bei den meisten Softwareprodukten üblich und notwendig ist.

4. Wissensfragen:

   (a) Mit welchen Tools lassen sich in Projekten, bei denen viele Entwickler am gleichen Quelltext arbeiten, die Fortschritte protokollieren und ggf. auch wieder rückgängig machen?

   (b) Was genau soll die Systemdokumentation darstellen? Geben Sie ein Beispiel für eine Systemdokumentation einer Java-Klasse an.

   (c) Warum sind Versionsnummern i.d.R. ungeeignet, um die „Marktreife“ bzw. Aktualität eines Software-Produktes feststellen zu können?

   (d) Wie lässt sich der „Erfolg“ eines Softwareprojektes, nach Software-Engineering-Methoden, feststellen bzw. messen?

   (e) Spielen Sie in Gedanken (ohne Implementation) die Entwicklung eines neuen Computerspiels mit Software-Engineering-Methoden von der Spielidee bis zur Markteinführung durch. Welche begleitenden Prozesse sind bei einem Computerspiel relevant, welche nicht?