Bitte tragen Sie individuell bis 08:00 Uhr am Vortag des Seminars über den Checklistenbaustein in Opal ein, welche Aufgaben Sie präsentieren können. Für die Präsentation sollten Sie ein Dokument im Querformat vorbereitet haben, das Sie dann entsprechend im Seminar in BigBlueButton hochladen können. Auch müssen Sie den virtuellen Klassenraum mit Mikrofon betreten.

Aufgabe 1: Termüberdeckung

Betrachten Sie die folgenden Quelltextausschnitt

```
...
if ( (x<y) && ( (2*x >= y) || (y > 5*x) ) ) { ... }
...
```

Bestimmen Sie eine Menge an konkreten Testfalleingaben, für die der Ausdruck vollständig termüberdeckt ist.

Aufgabe 2: Refactoring

Betrachten Sie die folgende Methode:

```
public double compute(int[] numbers, int threshold) {
   int sum = threshold/numbers.length - 1;
   int num = threshold;
   /* Punkt A */
   for(int k=0; k < numbers.length; k++) {
      /* Punkt B */
      sum = sum + numbers[k];
      /* Punkt C */
   }
   /* Punkt D */
   if (sum > num)
      num = numbers.length;
   else
      num = (int)(Math.sqrt(numbers.length);
   double result = sum / num + threshold;
   return result;
}
```

a) Extrahieren Sie die Zeile zwischen B und C in eine eigene Methode.

b) Extrahieren Sie die Zeilen zwischen A und D in eine eigene Methode.

Aufgabe 3: Refactoring II

Betrachten Sie die folgende Java-Klasse:

```
class Mitarbeiter {
   private String name;
   private String vorwahl;
   private String telefonnummer;
   public Mitarbeiter (String name, String vorw, String nr) {
      this.name = name;
      vorwahl = vorw;
      telefonnummer = nr;
```

```
    }
    public String getName() {
        return name;
    }
    public String getTelefon() {
        return vorwahl + " / " + telefonnummer;
    }
}
```

Die beiden Attribute `vorwahl` und `telefonnummer` sollen in einer eigenen Klasse gekapselt werden.

a) Zeichnen Sie das zugehörige Klassendiagramm.

b) Schieben Sie zunächst nur die Attribute in die neue Klasse und stellen Sie Getter-Methoden zur Verfügung. Was ändert sich im Quelltext?

c) Verschieben Sie jetzt auch die Methode `getTelefon` und deligieren Sie den Aufruf in der Klasse `Mitarbeiter`. Geben Sie den resultierenden Java-Code beider Klassen an.

Aufgabe 4: Refactoring III (alte Prüfungsaufgabe)

Betrachten Sie die folgende Java-Methode.

```
1:  public String describeDetails(MyObject obj, boolean all) {
2:    boolean status = all;
3:    String[] items = obj.getAllItems();
4:    String result = obj.getIntro(status);
5:    result += status.getCount();
6:    status = status && obj.getStatus(); // getStatus() hat keine Seiteneffekte
7:    if (status) {
8:      for (int i=0; i<items.length; i++)
9:        if (items[i].length() > 10)
10:           result = result + "\\" + items[i];
11:   }
12:   return result + " - Ergebnis: " + status;
13: }
```

Es sollen die Zeilen 4–11 gemäß der folgenden Teilaufgaben in eine Methode `composeString` extrahiert werden.

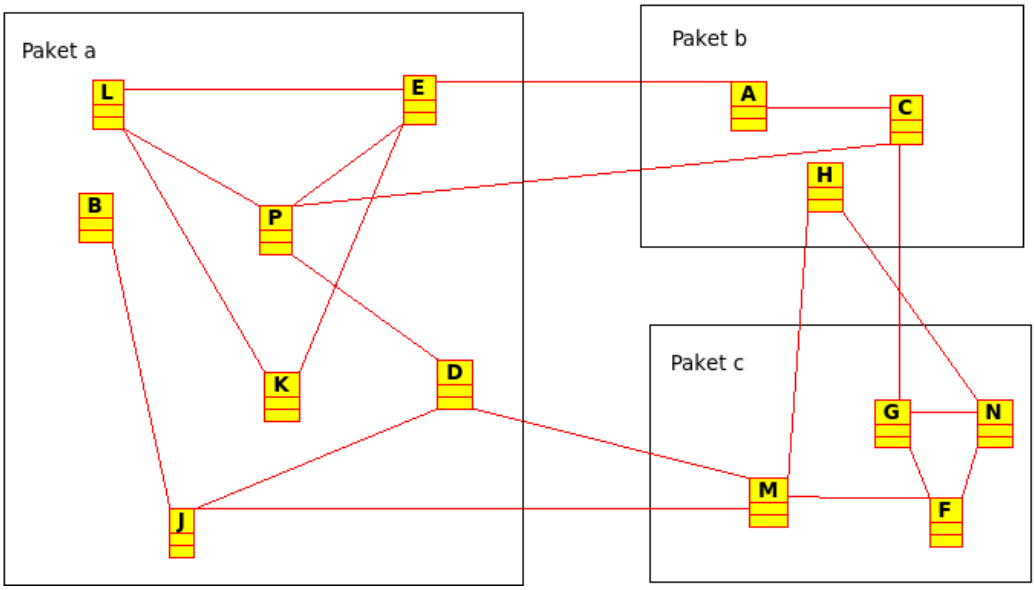
a) Argumentieren Sie, warum die Methode nicht direkt per Refactoring extrahiert werden kann.

b) Führen Sie „Temporäre Variable zerlegen“ auf der Variable `status` durch.

c) Führen Sie „Temporäre Variable durch eine Abfrage ersetzen“ auf einer aus `status` entstandenen Variablen durch.

d) Extrahieren Sie jetzt die ursprünglichen Zeilen 4–11 und geben Sie den resultierenden Quelltext von `describeDetails` an.

Aufgabe 5: Kopplung und Kohäsion



a) Bestimmen Sie die Metrik PCoh für jedes Paket sowie IIPU für die gesamte Architektur.

b) Teilen Sie die Klassen neu zu drei Paketen (mit jeweils 4–5 Klassen) zu, sodass die Kohäsion möglichst groß und die Kopplung zwischen den Paketen möglichst klein ist. Berechnen Sie erneut die Werte der Metriken.