Wintersemester 2017/18 Beispielaufgaben

Karlsruher Institut für Technologie

Programmieren – Wintersemester 2017/2018

Architecture-driven Requirements Engineering (ARE)
Jun.-Prof. Dr.-Ing. Anne Koziolek

Beispielaufgaben Präsenzübung 1

Allgemeine Hinweise

Am 17.01.2018 findet die Präsenzübung statt. Informationen zur Hörsaaleinteilung und zum Zeitplan werden rechtzeitig in Ilias bekanntgegeben. Im Folgenden finden Sie Aufgaben, die beispielhaft für die Aufgaben in der Präsenzübung sind.

Hinweis: Die Aufgaben werden in der Präsenzübung nur auf Deutsch gestellt. Falls Sie ein Wörterbuch benötigen, geben Sie dieses **bis spätestens 10.01.2018** im Sekretariat von Prof. Reussner (Gebäude 50.34, Raum 328. Öffnungszeiten Montag bis Freitag, 11:00 Uhr bis 12:30 Uhr) zur Überprüfung ab. **Nicht durch uns geprüfte Wörterbücher können nicht verwendet werden.** Sie erhalten das Wörterbuch in der Präsenzübung von uns zurück. Falls Sie nicht an der Präsenzübung teilnehmen, können Sie es wieder im Sekretariat abholen.

All questions in the written test („Präsenzübung“) will be in German. If you need a dictionary, hand it in for checking at the secretary's office of Professor Reussner (building 50.34, room 328. Opening hours: Monday to Friday, 11:00 to 12:30) **at the latest on 10.01.2018**. **If we have not checked the dictionary, you will not be allowed to use it during the test.** We will return the dictionary to you immediately before the written test. If you don't participate in the written test you can pick the dictionary up at the secretary's office.

Hinweis: Für eine korrekte Lösung müssen nicht unbedingt alle Lücken ausgefüllt werden. Die Größe der Lücken steht weiterhin nicht unbedingt in Relation zur Länge des Texts, der eingefüllt werden muss.

Hinweis: Sie müssen bei der Beantwortung der Präsenzübungsaufgaben keine Vorgaben zum Code-Style einhalten wie sonst bei der Bearbeitung der Übungsblätter. Insbesondere können Sie nichtsprechende und kurze Methoden- und Variablennamen wählen.
Den Bezeichner `System.out.println` können Sie in Ihrer Lösung mit `sysout` abkürzen.

Aufgaben

1. Aufgabe: Java-Basics – Operanden und Operatoren (Beispiel)

(Lösung auf Seite 5)

Vervollständigen Sie Tabelle 2 für den Zustand nach Ausführung des folgenden Codes:

```
1 boolean x = true;
2 boolean y = false;
3 boolean z = true;
4 boolean w = (!(x || y) && z) | ((z && !y) ^ !x); // ^ ist der XOR-Operator
```

Expression	true	false
x	✗	
y		✗
z	✗	
!(x || y)		

Expression	true	false
!(x || y) && z		
(z && !y)		
(z && !y) ^ !x		
w		

Tabelle 1: Wahrheitswerte

2. Aufgabe: Objekt-Orientierung – Vererbung (Beispiel)

(Lösung auf Seite 5)

Ergänzen Sie den folgenden lückenhaften Code so, dass das Programm kompiliert und ein Aufruf der `main`-Methode den Text „correct message“ ausgibt.

```
public class ClassA {
  public String getMessage() {
    return "wrong message";
  }
}

public class ClassB                                          {
                                                              
                                                              
                                                              
                                                              
                                                              
                                                              
                                                              
  public static void main(String[] args) {
    ClassA messageProvider = new                                    ();
    System.out.println(messageProvider.getMessage());
  }
}
```

3. Aufgabe: Kontrollfluss (Beispiel)

(Lösung auf Seite 5)

Ergänzen Sie die folgenden Methode so, dass alle Werte des Felds `values` ausgegeben werden, die echt größer als `threshold` sind. Verwenden Sie zur Ausgabe die Methode `System.out.println`.

```
public static void printGreaterThan(int[] values, int threshold) {


}
```

4. Aufgabe: Objekt-Orientierung – Modellierung (Beispiel)

(Lösung auf Seite 6)

Ergänzen Sie die folgenden drei Klassen Person (`Person`), Studierende(r) (`Student`) und Universität (`University`) um die folgende Beschreibung zu erfüllen. **Sie müssen keinen Code zum Setzen der Attribute schreiben (keine Setter oder Konstruktoren). Ergänzen Sie falls nötig Attribute und Getter-Methoden. Greifen Sie falls möglich ohne Getter-Methoden auf Attribute zu.**

Ein(e) Studierende(r) ist eine Person, die zu genau einer Universität gehört. Die Email-Adresse einer studierenden Person setzt sich zusammen aus

`Vorname in Kleinbuchstaben` `.` `Nachname in Kleinbuchstaben` `@student.` `Universitätsdomäne`

Diese wird von der Methode `getStudentMailAddress()` der Klasse `Student` **berechnet** und ist kein Attribut von `Student`. Das Ergebnis wäre beispielsweise für eine Person mit dem Vornamen „Efemena“ und dem Nachnamen „Girard“, die an einer Universität mit der Domäne „kit.edu“ studiert: `efemena.girard@student.kit.edu`. Verwenden Sie zum Konvertieren in Kleinbuchstaben die Instanz-Methode `toLowerCase()` der Klasse `String`.

```
public class Student                                    {
  private University university;

  public String getStudentMailAddress() {


  }
}
```

```
public class Person                                                    {
  protected String firstName;
  protected String lastName;

}
```

```
public class University {
  private String domain;

}
```

Lösungen

1. Lösung: Java-Basics – Operanden und Operatoren (Beispiel)

(Aufgabe auf Seite 2)

Vervollständigen Sie Tabelle 2 für den Zustand nach Ausführung des folgenden Codes:

```
1 boolean x = true;
2 boolean y = false;
3 boolean z = true;
4 boolean w = (!(x || y) && z) | ((z && !y) ^ !x); // ^ ist der XOR-Operator
```

Expression	true	false
x	✗	
y		✗
z	✗	
!(x || y)		✗

Expression	true	false
!(x || y) & z		✗
(z && !y)	✗	
(z && !y) ^ !x	✗	
w	✗	

Tabelle 2: Wahrheitswerte

2. Lösung: Objekt-Orientierung – Vererbung (Beispiel)

(Aufgabe auf Seite 2)

Ergänzen Sie den folgenden lückenhaften Code so, dass das Programm kompiliert und ein Aufruf der `main`-Methode den Text „correct message“ ausgibt.

```
public class ClassA {
  public String getMessage() {
    return "wrong message";
  }
}

public class ClassB extends ClassA {
  @Override // Zeile optional fuer korrekte Loesung
  public String getMessage() {
    return "correct message";
  }

  public static void main(String[] args) {
    ClassA messageProvider = new ClassB();
    System.out.println(messageProvider.getMessage());
  }
}
```

3. Lösung: Kontrollfluss (Beispiel)

(Aufgabe auf Seite 3)

Ergänzen Sie die folgenden Methode so, dass alle Werte des Felds `values` ausgegeben werden, die echt größer als `threshold` sind. Verwenden Sie zur Ausgabe die Methode `System.out.println`.

```
public static void printGreaterThan(int[] values, int threshold) {
  // Antwortmoeglichkeit A
  for (int i = 0; i < values.length; i++) \{
   \ if (values[i] > threshold) \{
   \ \ \ System.out.println(values[i]);
   \ \}
  \}

  // Antwortmoeglichkeit B
  for (int value : values) \{
   \ if (value > threshold) \{
   \ \ \ System.out.println(value);
   \ \}
  \}
}
```

4. Lösung: Objekt-Orientierung – Modellierung (Beispiel)

(Aufgabe auf Seite 3)

Ergänzen Sie die folgenden drei Klassen Person (`Person`), Studierende(r) (`Student`) und Universität (`University`) um die folgende Beschreibung zu erfüllen. **Sie müssen keinen Code zum Setzen der Attribute schreiben (keine Setter oder Konstruktoren). Ergänzen Sie falls nötig Attribute und Getter-Methoden. Greifen Sie falls möglich ohne Getter-Methoden auf Attribute zu.**

Ein(e) Studierende(r) ist eine Person, die zu genau einer Universität gehört. Die Email-Adresse einer studierenden Person setzt sich zusammen aus

`Vorname in Kleinbuchstaben` `.` `Nachname in Kleinbuchstaben` `@student.` `Universitätsdomäne`

Diese wird von der Methode `getStudentMailAddress()` der Klasse `Student` **berechnet** und ist kein Attribut von `Student`. Das Ergebnis wäre beispielsweise für eine Person mit dem Vornamen „Efemena“ und dem Nachnamen „Girard“, die an einer Universität mit der Domäne „kit.edu“ studiert: `efemena.girard@student.kit.edu`. Verwenden Sie zum Konvertieren in Kleinbuchstaben die Instanz-Methode `toLowerCase()` der Klasse `String`.

```
public class Student extends Person {
  private University university;

  public String getStudentMailAddress() {
    return firstName.toLowerCase()
     \ + "."\ + lastName.toLowerCase()
     \ + "@student."\ + university.getDomain();
  }
}
```

```
public class Person {
  protected String firstName;
  protected String lastName;
}
```

```
public class University {
  private String domain;

  public String getDomain() \{
   \ return this.domain;
  \}
}
```