Fragen, Antworten, Kommentare und Hinweise

Frage: Welche Voraussetzungen muss ich zur Teilnahme erfüllen?

Antwort: Formal benötigen Sie die 40 Leistungspunkte um das Praktikum oder die Hausarbeit angerechnet zu bekommen. Generell benötigen Sie gute Programmierkenntnisse, Kenntnisse aus Datenbanken (Teil SQL-Anfragen) und Algorithmen und Datenstrukturen (u. a. Bearbeitung von Bäumen).

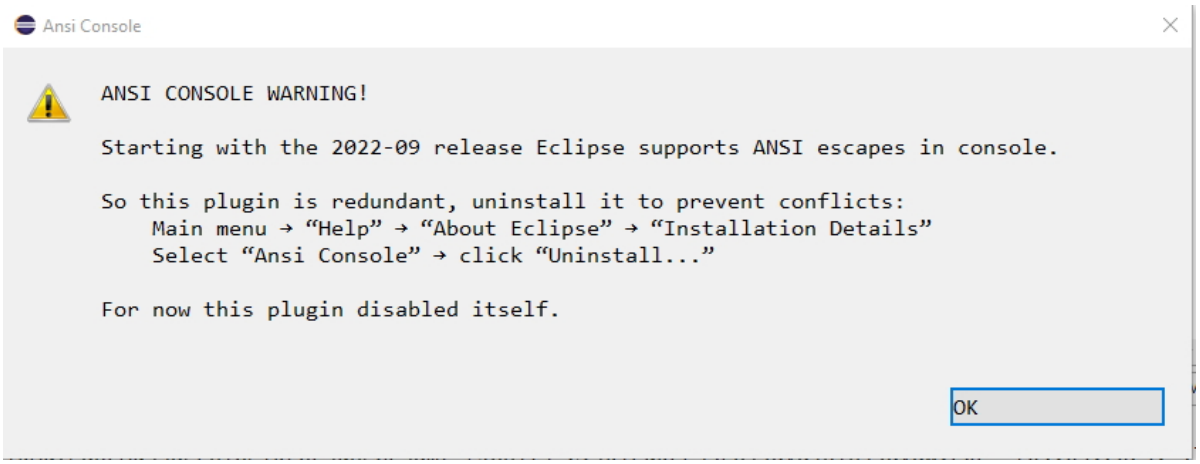
Frage: Im Veranstaltungsverzeichnis sind Räume für die Online-Veranstaltungen angegeben, warum?

Antwort: Die Räume werden generell für die Veranstaltung nicht genutzt, stehen aber als Backup-Lösungen, z. B. technischen Problemen zur Verfügung. Insofern die Räume offen sind, können sie natürlich auch für Zoom-Sessions genutzt werden. Ein Eintrag eines Links ist im Vorlesungsverzeichnis aktuell nicht möglich.

Frage: Muss ich genau diese KleukerSEU nutzen?

Antwort: Die SEU wurde erstellt, um möglichst wenig Konfigurationsprobleme zu haben, sie läuft auch in der Hochschule unter C:. Generell sind Sie aber für Ihre Ergebnisse verantwortlich und solange es läuft, ist es egal, wie Sie es gemacht haben. In der Hochschule sind die Ergebnisse mit dieser SEU vorzuführen. Beachten Sie, dass für auf Java/Drools oder Prolog basierenden Hausarbeiten ein in der KleukerSEU lauffähiges Projekt gefordert ist.

Frage: Ich bekomme bei Eclipse immer einmal das folgende Fenster angezeigt. Soll ich das Plugin deinstallieren?



Antwort: Besser nicht, da der erwähnte Support unsauber umgesetzt wurde und so z. B. Ergebnisse aus der Console nicht sauber mit Formatierung in ein Word-Dokument kopiert werden können.

Hinweis: Abhängig von Ihre bisherigen Java-Ausbildung, hier zwei Sprachkonstrukte, die Sie eventuell noch nicht gesehen haben, aber in dieser Veranstaltung vorkommen.

Bei Methoden kann beim letzten Parameter hinter dem Typen … geschrieben werden, wodurch beliebig viele Parameter dieses Typen beim Aufruf kommasepariert am Ende stehen können (ähnliche Ideen gibt es in C++ und anderen Sprachen). Formal entspricht dies einem Array dieses Typens, der wie üblich bearbeitet werden kann.

```
public class DynamischeParameteranzahl {
  public static int summ(int... pars) {
    int erg = 0;
    for(int i=0; i < pars.length; i++) { // "forEach" geht auch
      erg += pars[i];
    }
    return erg;
  }

  public static void main(String... s) {
    System.out.println("1: " + summ());
    System.out.println("2: " + summ(42));
    System.out.println("3: " + summ(43 ,44, 45, 46, 47));
  }
}
```

Ausgabe:

```
1: 0
2: 42
3: 225
```

Man kann statt `public static void main(String[] args1)` zum Start auch `public static void main(String… args2)` schreiben. Der formale Unterschied ist, dass args1 null sein kann, args2 bei einem Aufruf ohne Parameter ein leerer Array ist.

In Java gibt es die Möglichkeit in Interfaces default-Implementierungen der Methoden anzugeben, die genutzt werden können, wenn eine das Interface implementierende Klasse diese Methode nicht selber realisiert. Formal verwischt damit der Unterschied zu abstrakten Klassen. Der Vorteil ist aber die einfache Erweiterbarkeit. Sollte ein klassisches Interface um eine Methode ergänzt werden, bedeutete dies, dass alle Klassen, die dieses Interface realisieren bearbeitet werden mussten, um die neue Methode zu ergänzen. Diese Notwendigkeit fällt mit default-Methoden weg und die Erweiterung von Interfaces ist so problemlos möglich. Natürlich kann in diesen Implementierungen auf keine Objektvariablen zugegriffen werden, die Nutzung anderer Klassen und Methoden ist wie üblich erlaubt. Generell bleibt die dynamische Polymorphie das mit Abstand wichtigste Konzept der Objektorientierung.

```
public interface InterfaceMitDefaultmethode {
  int wasBinIch = 1;  // hat nichts mit default zu tun

  public default void zeigeZahl(int wert) {
    System.out.println("Zahl: " + wert);
  }
}

public class Bsp implements InterfaceMitDefaultmethode {
  public static void main(String[] args) {
    Bsp bsp = new Bsp();
```

```
    bsp.zeigeZahl(42);
  }
}
```

Die Ausgabe ist:

```
Zahl: 42
```

Im obigen Beispiel steht noch eine Variable `wasBinIch`, die zunächst für eine Objektvariable gehalten werden könnte. Das ist nicht der Fall. Da bei Interfaces redundante Informationen weggelassen werden können, sind hier die Schlüsselworte „public“ und „static“ und „final“ (Interfaces können niemals Objektvariablen haben) weggelassen. Meine Empfehlung ist, sie hinzuschreiben, da das Programm so für nicht-javaerfahrene Personen lesbarer wird. Das „public“ vor dem default kann nebenbei auch weggelassen werden ohne dass sich die Sichtbarkeit der Methode ändert, da ja das Interface schon public ist. Eine Nutzung kann wie folgt aussehen.

```
public class WildeKonstantennutzung {

  public static void main(String[] args) {
    System.out.println("Aha: " + InterfaceMitDefaultmethode.wasBinIch);
  }
}
```

Die Ausgabe ist:

```
Aha: 1
```