

Klausur AP2 – Beispielaufgaben

Das folgende sind zwei repräsentative Beispiele für Klausuraufgaben in AP2

Beispiel 1

Gegeben ist die folgende (unvollständige) Klassendeklaration:

```
/** Realisierung einer einfach verketteten Liste von
 * Textstrings
 */
public class StringList {

    private static class Node {
        private String value;
        private Node next;
    }
    private Node first;
    private int sizeVar;

    public StringList() { ...}

    /** gibt Anzahl der Listenelemente zurück */
    public int size() { ... }

    /** hängt value an das Ende der Liste */
    public void addLast(String value) { ... }

    /** true, wenn die Liste value enthält */
    public boolean contains(String value) { .. }

    /** sortiert die Liste nach aufsteigenden Strings
    */
    public void sort() { ... }
}
```

- a) Implementieren Sie den Konstruktor (erzeugt eine leere Liste) und die Funktion `size()`.
- b) Implementieren Sie die Funktion `addLast()`
- c) Implementieren Sie die Funktion `contains()`.
- d) Schreiben Sie die Funktion `sort()`. Gehen Sie dabei wie folgt vor: Wandeln Sie die Liste in ein Feld (Array) um. Rufen Sie für dieses Feld eine (vorhandene) Sortierfunktion auf:

```
java.util.Arrays.sort(array);
```

Achten Sie darauf, dass durch Ihren Algorithmus weder unnötige Einschränkungen noch eine Vergeudung von Speicherplatz verursacht werden.

- e) Erläutern Sie die Grundidee von Sortieren durch direkte Auswahl (selection sort). Geben Sie an, wie stark die Anzahl der Vergleiche, die Anzahl der Vertauschungen und die Gesamtrechenzeit mit der Anzahl N der zu sortierenden Elemente ansteigen (O-Notation).

Beispiel 2

Gegeben ist die folgende Klassendeklaration:

```
/** nützliche Berechnungen mit reellen Funktionen */
class Rechner {
    public static double berechneIntegral(
        ReelleFunktion f, double x0, double x1,
        int n) { ... }

    public static void zeichne(ReelleFunktion f,
        double x0, double x1) { ... }
}
```

Die angegebene Klassendeklaration ist Teil einer unfertigen Baustelle für ein Programm, das verschiedene Aktionen mit verschiedenen reellen Funktionen ausführen kann. Reelle Funktionen werden in dem Programm durch eine Schnittstelle beschrieben (s. 4b). Jede konkrete Funktionsklasse implementiert diese Schnittstelle.

4a) Was bedeutet das Wort **static** bei den Methodendeklarationen der Klasse `Rechner` und weshalb wird es hier verwendet?

4b) Schreiben Sie die Schnittstelle `ReelleFunktion` vollständig auf. Die Schnittstelle enthält die Methode `double f(double x)`.

4c) Schreiben Sie eine die Schnittstelle `ReelleFunktion` implementierende (konkrete) Klasse `Parabelfunktion`. Objekte der Klasse `Parabelfunktion` repräsentieren jeweils die Funktionsgleichung ax^2+bx+c (mit festen Werten für a, b, c).

4d) Programmieren Sie die Funktion `berechneIntegral()` aus der Klasse `Rechner`. Verwenden Sie zur numerischen Integralberechnung die Trapezregel, wonach angenähert für den Wert des Integrals gilt:

$$\int_{x_0}^{x_1} f(x) dx \approx \frac{1}{2} h (f(x_0) + f(x_1)) + h \sum_{i=1}^{n-1} f(x_0 + ih)$$

$$\text{mit } h = \frac{x_1 - x_0}{n}$$

Das Summenzeichen ist eine Abkürzung für $f(x_0+h)+f(x_0+2h)+\dots+f(x_0+(n-1)h)$

4e) Vervollständigen Sie das folgende Main-Programm:

```
public static void main(String[] args) {  
    // Anlegen einer Parabelfunktion  
    // (a = 1, b = 2, c = -1):  
  
    ReelleFunktion f = .....  
  
    // Berechnung des Integrals von x0 = -1 bis  
    // x1 = 1. Anzahl der Intervalle n = 20:  
  
    double integral = ...  
  
}
```