Informatikklausur Nr. 2

erweiterte Programmiertechniken

Gestaltung und Abgabe deiner Klausur:
Du erstellst ein LibreOffice oder Word-Dokument und lädst es zur Abgabe im Aufgabenmodul von IServ hoch. Dieses Dokument wird dir auch digital zur Verfügung gestellt. Du darfst es daher gerne als Vorlage für deine Lösungen verwenden.

Hinweis:
Du darfst nur Funktionen und Schreibweisen verwenden, die im Unterricht eingeführt worden sind! Achte darauf bei Copy&Paste oder Nutzung von ChatBots. Wenn du externe Quellen oder ChatBots verwendest, MUSST du die Quelle angeben bzw. die Codeteile z.B. blau markieren, die nicht von dir stammen.

Aufgabe 1 – Programmierung (4 + 6 [+ 4] Punkte):

Du musst alle deine Programme mit sinnvollen, deutschen Kommentaren versehen. Denke daran, dass wir schon oft festgestellt haben, dass die Kommentare bei den Beispielprogrammen im Netz nicht immef stimmen ...

a) Schreibe ein Programm, welches die Zahlen eines Arrays aufsummiert. Das Array sollte aus mindestens 10 Zahlen bestehen. Die Bildung der Summe muss in einer Methode mit dem Namen *arraySum(array)* erfolgen.

b) Schreibe ein Programm, welches folgende zwei Arrays

1: [0,2,4,6,8,10]
2: [1,3,5,7,9,11]

so zusammenfügt, dass folgendes Array entsteht:

3: [0,1,2,3,4,5,6,7,8,9,10,11]

Bonusaufgabe (nur Zusatzpunkte):
Die Voraussetzungen bei b) sind sehr günstig, da beide Arrays die gleiche Anzahl an Elementen enthalten. Ändere dein Programm so, dass es auch mit folgenden Konstellationen funktioniert:

1: [0,2,4,6,8,10]
2: [1,3,5,7,9]

1: [2,4,6,8,10]
2: [1,3,5,7,9,11]

Tipp:
Denke daran, dass die Variablennamen für Arrays nur Zeiger auf Datenbereiche sind. Im Code *könnte* es darauf ankommen, ob du mit dem kleineren oder größeren Array beginnst. Du könntest das jeweils kleinere z.B. in *smaller* und das größere in *bigger* umbenennen. Vielleicht findest du aber auch eine andere Lösung!

Aufgabe 2 – Objektorientierung (4 + 4 + 6 Punkte):
Das Planetarium Hamburg möchte eine Sternendatenbank anlegen. Diese soll unbedingt objektorientiert sein und für den Anfang folgende fünf Sterne enthalten:

Attribut	star1	star2	star3	star4	star5
Name	Sirius	Alpha Centauri	Rigel	Almaaz	Luhmann 16
Id	TYC 5949-2777-1	TYC 9007-5849-1	TYC 5331-1752-1	TYC 2907-1275-1	WISE J1049-5319A
Distance	8.6	4.37	860	2000	6.589
apparentMagnitude	-1.46	-0.27	0.13	2.92	14.94
type	Main sequence	Main sequence	Blue supergiant	Yellow supergiant	Brown dwarf

Erläuterung:
Distance: Entfernung zur Erde in Lichtjahren (LY)
apparentMagnitude: Einheit für die scheinbare Helligkeit eines Steins

a) Schreibe eine Klasse *Star*, die die Informationen eines Sterns in geeigneter Weise repräsentiert. Du wirst zwei Datentypen (float + string) brauchen ...

b) Erweitere die Klasse *Star* um zwei Methoden:
- **isCloserThan(star)**, die genau dann **true** zurückgibt, wenn der aktuelle Stern näher an der Erde ist als der Vergleichsstern, sonst **false**.
- **getDistanceInPC()**, welche die Entfernung des Sterns in Parsec (Abkürzung PC) zurückgibt. Gespeichert ist die Entfernung in Lichtjahren (Abkürzung LY). Es gilt 1 PC ≈ 3.26 LY.

c) Schreibe eine weitere Klasse StarsDatabase, welche ein Array von Sternen verwaltet und Methoden zum

- Hinzufügen von Sternen – **add(star)**
- Löschen von Sternen – **remove(star)**
- Abfragen von Sternen, Ausgabe aller Daten – **get(star)**
- Abfragen der Größe der Datenbank – **size()**

bereitstellt. Wundere dich nicht. Diese Klasse kommt im Prinzip tatsächlich mit einem einzigen Attribut („Array of stars“) aus.

Tipp:
Wenn dein Array aus Sternen s*tars[]* heißt …

stars.remove(x) - entfernt einen Stern x
stars.append(x) – fügt einen neuen Stern x hinzu

viel Erfolg!