11. Übungsblatt: Programmierpraktikum I (WS 2003/04)

Abgabe: 26. Januar 2004

Java: Klasse *Genstring*

(Genstring.java, 4 Punkte)

Als Vorbereitung für Aufgabe 2 soll eine Klasse `Genstring` implementiert werden wie folgt:

- Ein `Genstring`-Objekt ist 32 Bit lang, es wird durch ein anderes Genstring-Objekt oder einfach per Zufallsgenerator initialisiert.
- Ein `Genstring`-Objekt kann seinen 32-Bit-String als `long`-Wert zurückgeben.
- Ein `Genstring`-Objekt kann **mutieren**, indem 1-5 (zufällig ermittelte) Bits invertiert werden.
- Zwei `Genstring`-Objekte können per **Crossover** zu einem neuen Objekt verschmelzen. Dazu wählt man eine zufällige Position i im Genstring ($1 \leq i \leq 31$) und setzt dann die Bits 0 bis $i-1$ des ersten Genstrings und die Bits i bis 31 des zweiten Genstrings zu einem neuen `Genstring`-Objekt zusammen.
- Die **Güte** eines `Genstring`-Objektes bzgl. eines Eingabeparameters a wird berechnet als
$$\text{Güte} = \frac{1}{|x^2 - a|} \ ,$$
wobei x der Wert des `Genstring`-Objektes als `long` ist. Die Gütefunktion ist so gewählt, dass ein `Genstring`-Objekt dann besonders gut ist, wenn sein Wert x nahe am Wert $\sqrt{a}$ liegt.

Es dürfen beliebige weitere Hilfsmethoden implementiert werden, wo dies sinnvoll erscheint.

Java: Wurzel-Approximation

(Aufgabe2.java, 8 Punkte)

Es soll nun ein Approximationsalgorithmus für die Wurzel einer natürlichen Zahl a implementiert werden[1]. Dieser soll wie folgt vorgehen:

[1] Es handelt sich hierbei um einen sogenannten genetischen Suchalgorithmus. Die Vorgehensweise dieser Algorithmen folgt dem Vorgehen der Natur beim Prinzip der genetischen

1. Er beginnt mit zehn vollkommen zufälligen `Genstring`-Objekten.
2. Er erzeugt fünf weitere Objekte, indem er aus den ursprünglichen zehn Objekten jeweils ein Objekt zieht (mit Zurücklegen) und es mutiert.
3. Er erzeugt fünf weitere Objekte, indem er aus den ursprünglichen zehn Objekten jeweils zwei verschiedene Objekte zieht (danach werden sie zurückgelegt) und sie per Crossover zu einem neuen Objekt verschmilzt.
4. Er ordnet die zwanzig Objekte nach ihrer Güte (bzgl. a) und behält nur die zehn besten.
5. Falls die Güte des besten Objektes zum fünften Mal gleich ist, hält der Algorithmus an und gibt den Wert dieses Objektes als `long` aus. Andernfalls fährt der Algorithmus wieder in Zeile 2 mit der Berechnung fort.

Dokumentiere in der schriftlichen Abgabe verschiedene Testläufe deines Programmes, wobei jeweils angegeben werden soll, wie viele Runden der Algorithmus zum Finden der Lösung benötigt hat. Außerdem ist jeweils der Wert a, die gefundene Approximation für $\sqrt{a}$ und der tatsächliche Wert für $\sqrt{a}$ anzugeben.

Java: Mitarbeiterdatenbank

(Dateien: s.u., 8 Punkte)

Ein Unternehmen hat drei Arten von Mitarbeitern:

- Ein `Arbeiter` erhält pro Jahr 13 Monatsgehälter zu je 1400 Euro (bei 13 Gehältern) zuzüglich 20 Euro pro geleisteter Überstunde.
- Ein `Angestellter` hat ein individuell verschiedenes Monatsgehalt. Pro Jahr kostet er den Arbeitgeber $13 \cdot \mathit{Monatsgehalt}$ Euro.
- Ein `freier Mitarbeiter` erhält eine individuell unterschiedliche Prämie pro erledigtem Auftrag. Pro Jahr kostet er $\mathit{Auftragszahl} \cdot \mathit{Auftragspr\ddot{a}mie}$ Euro.

Leite die Klassen `Arbeiter`, `Angestellter` und `FreierMitarbeiter` von einer gemeinsamen Basisklasse `Mitarbeiter` ab. Dabei soll jede Klasse eine eigene Version der Methode `jahresgehalt()` besitzen, die am Jahresende die Kosten für den betreffenden Mitarbeiter zurückgibt. Darüber hinaus muss jede Klasse die zur Berechnung benötigten Variablen (z.B. `ueberstunden` für Arbeiter) und passende Konstruktoren besitzen.

Implementiere nun im Hauptprogramm der Klasse `Aufgabe3` eine Mitarbeiterdatenbank als ein Array von 100 Mitarbeitern. Nachdem die Datenbank initialisiert wurde (z.B. per Zufallsgenerator), sollen die Daten aller Mitarbeiter am Bildschirm ausgegeben werden. Außerdem soll berechnet werden, welche Personalkosten im abgelaufenen Jahr für die gesamte Belegschaft angefallen sind.

Auslese. Genetische Algorithmen können auf viele Approximations- und Optimierungsprobleme angewandt werden und sind unter anderem in der Künstlichen Intelligenz populär.

Bem.: Es genügt, wenn das Programm ganze Euro-Beträge und volle Überstunden verarbeitet, es muss nicht mit Hinterkommastellen gerechnet werden.