Programmierung für Mathematik HS12
Übung 8

12. November 2012

1 Aufgabe: Formen

1.1 Ziele

- Vererbung kennen lernen

1.2 Aufgabenstellung

Implementieren Sie die zwei Klassen `Square` und `Rectangle` . Diese zwei Klassen sollen die Methoden `perimeter()` und `area()` besitzen. Die Klassen sollen nebst den beiden Methoden auch einen passenden Konstruktor implementieren. Argumentieren Sie ob dabei die Klasse `Rectangle` von der Klasse `Square` erben soll oder `Square` von der `Rectangle` Klasse.

2 Aufgabe: Reihen

2.1 Ziele

- Vererbung vertiefen
- Abstrakte Klassen verstehen
- Mathematische Funktionen darstellen

2.2 Aufgabenstellung

2.2.1 Series

In dieser Aufgabe sollen Sie eine abstrakte Klasse `Series` erstellen. Diese Klasse soll eine Reihe darstellen, also eine Funktion der Art

$$f(x) = \sum_{n=0}^{\infty} a_n x^n$$

Ihr Programm soll die Summe nicht bis unendlich berechnen. Deshalb erwartet diese Klasse einen `int` Wert in ihrem Konstruktor der angibt bis zu welcher Grenze summiert wird. a_n sollen anhand von n berechnet werden können. Allerdings wird a_n erst in den Subklassen definiert. Deshalb besitzt die Klasse zusätzlich eine abstrakte Methode, die den Summand definiert:

```
protected abstract float summand(int n);
```

Desweiteren soll folgende Methode implementiert werden:

```
public float evaluate(float x)
```

die die Teilsumme, bis zum Wert der im Konstruktor übergeben wurde, im Punkt x returniert.

2.2.2 Subklassen

Schreiben Sie folgende Klassen die von der Klasse `Series` erben sollen und überschreiben Sie die Methode `summand` damit sie jeweils die entsprechenden Werte returniert. Falls Sie weitere Hilfsmethoden schreiben überlegen Sie sich gut in welcher Klasse Sie diese erstellen.

1. `Exp` die folgender Reihe entsprechen soll

$$\sum_{n=0}^{\infty} \frac{x^n}{n!}$$

2. `Sin` die folgender Reihe entsprechen soll

$$\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

3. `Log` die folgender Reihe entsprechen soll

$$\sum_{n=0}^{\infty} \frac{x^k}{k}$$

Hinweis Die so erstellte Klasse `Log` entspricht der Funktion $-ln(1-x)$

Zusatz: Erweitern Sie die Super-Klasse `Series` um die Methode `bisection` . Diese Methode erwartet die folgenden Parameter: zwei Anfangswerte als `float` , ein weiterer `float` ϵ der die Präzision vorgeben soll und ein `int` der die maximale Anzahl Iterationen innerhalb der Methode vorgibt. Die Methode soll eine Nullstelle der Serie, ermittelt nach der Bisektions- Methode, zurückliefern. Dabei wird eine Stelle die kleiner ist als ϵ als Nullstelle wahrgenommen. Die Methode beginnt mit der Überprüfung, ob die Auswertung der Teilsummen in den Anfangswerten verschiedene Vorzeichen haben. Ansonsten wird `null` zurückgeliefert und eine Meldung in der Konsole eingeblendet. Wird nach der angegebenen Anzahl Schritte immer noch keine Nullstelle gefunden, so wird wieder `null` zurückgeliefert und eine entsprechende Meldung erstellt.

Schreiben Sie ein TestDriver und testen Sie ihre Klassen sorgfältig.