FM1

Übung:

Besteht aus 3 Teilen:

- Tutorium
- Hausaufgaben
- Tests

Tutorium:

- Übung und Vertiefen des Stoffes aus der Vorlesung(Keine Wiederholung!)

Hausaufgaben:

- 12 Arbeitsblätter, davon 9 zu bestehen
- Arbeitsblatt kommt am Mittwoch raus auf ISIS
- Abgabe:

Für Montags-Tutorien: Donnerstag in 9 Tagen bis 23:55Uhr auf ISIS
Für Dienstags-Tutorien: Freitag in 9 Tagen bis 23:55Uhr auf ISIS

Tipps für die Hausaufgaben:

- Lass bitte keine Aufgabe aus, dass fällt auf, bearbeite sie notfalls teilweise und zeige, dass Du das Problem verstanden hast.
- Mach die Hausaufgaben in der betreuten Rechnerzeit, die Tutoren helfen sehr gut und sagen Dir, was falsch ist und wie man es richtig macht.

Termine der betreuten Rechnerzeit im TEL 206 (das ist das große Haus mit dem Telekom T:)

Do: 12-18Uhr und 16-19Uhr
Fr: 10-16Uhr

Tests:

- 2 Tests
- Dauer: ca. 20Min.
- bestanden mit insgesamt 50% der Punkte in beiden Tests

Termine:
18./19.11.2013
16./17.12.2013
29.01.2014 (Wiederholungstermin)

Hinweis:
Solltest Du wegen Krankheit nicht erscheinen können, benötigst Du einen ärztlichen Attest, sonst ist der/die Test/Klausur nicht bestanden.

Projektaufgabe:

- Kleines Softwareprojekt
- 3 aufeinander aufbauende Meilensteine

Klausurtermine:

18.02.2014
08.04.2014 (Nachklausur)

Aufgabe 1. Funktionsbegriff

a)Was ist bitte eine Funktion?

Lösung:

- rechtseindeutige (evtl. linkstotale) Abbildung

b)Was ist bitte der Unterschied zwischen einer partiellen und einer totalen Funktion?

Lösung:

- partielle Funktion: rechtseindeutige Abbildung
*- totale Funktion: rechtseindeutige **und** linkstotale Abbildung*

c)Was beinhaltet bitte die formale Definition einer Funktion?

Lösung:

*- die Angabe des Definitions- und Wertebereichs (Funktionsdeklaration) **und***
***-** die Abbildungsvorschrift (Funktionsdefinition)*

Aufgabe 2. Modularisierung

a)Was kann man bitte darunter im Zusammenhang mit Opal verstehen?

Lösung:

- die Unterteilung in Strukturen
- das Aufteilen in zwei Teile (Implementierungs- und Signaturteil)

b)Welcher Teil der formalen Definition darf bitte in welchem Abschnitt stehen?

Lösung:

- Funktionsdeklaration in Signatur- und Implementierungsteil
- Funktionsdefinition nur im Implementierungsteil

e)Welche Auswirkungen auf die Sichtbarkeit lassen sich bitte dabei erkennen?

Lösung:

- Steht die Funktionsdeklaration im Signaturteil ist die Funktion nach aussen hin sichtbar.
- Eine Funktionsdeklaration, die nur im Implementierungsteil steht bewirkt, dass die Funktion nach aussen hin nicht sichtbar ist.

Mein erstes Opalprogramm

Ein Beispiel aus der Mathematik:

Wir geben eine Funktion f an mit Definitions- und Wertebereich, die das doppelte einer übergebenen Zahl x berechnet.

f: R → R mit f(x) = 2*x

Wir können der Funktion auch einen anderen Namen geben, z.B. mul2:

mul2: R → R mit mul2(x) = 2*x

In Opal:

Wir brauchen 2 Dateien:

Math.sign	Math.impl
SIGNATURE Math	IMPLEMENTATION Math
IMPORT Real COMPLETELY	IMPORT Real COMPLETELY
FUN mul2: real → real	DEF mul2 == \\x. 2*x

Starten des Programms: (Die fettgedruckten Befehle sind Eingaben)

Hinweis: Gehe zuerst bitte in das Verzeichnis, wo Du Deine .impl- und .sign-Dateien gespeichert hast

pronto quoc-hung 21 (FM/Blatt1): ***oasys1***
oasys version 1.1e (ocs version 2.3n), (c) 1989-2001 The OPAL Group, TU Berlin
*>****a Math***
loading Math.sign
loading Math.impl
*>****f Math.sign***
*Math.sign>****e mul2(3)***
checking Math.sign
checking Math.impl
compiling Math.impl
starting evaluator process
6

mit **q** läßt sich Oasys wieder schließen

Aufgabe 3.

Erstelle bitte zwei Dateien mit den Namen HelloWorld.sign und HelloWorld.impl, und schreibe bitte eine Funktion *sayHello*, die “Hello World!” ausgeben kann.

Hinweis:

*Um mit Zeichenketten arbeiten zu können, muss man die Struktur **DENOTATION** importieren. Zeichenketten in Opal stehen in Anführungsstrichen (Bsp: "text"). Da Du keine Eingabeargumente hast, entfällt das \\x.*

Aufgabe 4.

Erstelle bitte zwei Dateien mit den Namen Cubic.sign und Cubic.impl, und schreibe bitte eine Funktion *cubic*, welche eine Nat-Zahl hoch drei nimmt.

Hinweis:

*Für die Aufgabe musst Du bitte die Struktur **Nat** importieren.*

Aufgabe 5.

Erstelle bitte zwei Dateien mit den Namen Addition.sign und Addition.impl, und schreibe bitte eine Funktion *sum*, die zwei natürliche Zahlen addiert. Die Funktion soll von außen nicht 'sichtbar' sein

Hinweis:

*Da Ihr zwei Eingabeargumente habt, lautet die Deklaration sum: nat ** nat → nat und die Definition beginnt mit sum == \\x,y.*
Eine Funktion ist nicht sichtbar, wenn die Deklaration FUN:... nur in der .impl steht.

Aufgabe 6.

Schreibe bitte in den vorgestellten Dateien Math.sign und Math.impl eine Funktion *spherePerimeter*, die den Umfang einer Kugel (Umfang = 2*PI*radius) berechnet. Die Funktion soll von aussen sichtbar sein. Definiere Dir bitte dazu die Zahl *PI*. Diese soll als Hilfskonstante nicht sichtbar sein und die Zahl Pi darstellen (Zwei Nachkommastellen genügen). Benutze bitte die Funktion *mul2* von oben.

Hinweis:

Beachte, dass Du bitte PI konvertieren musst, da es eine Kommazahl ist.
Bsp. : Statt 1,2 schreibt man "1.2"! (inkl. dem !)