Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 – WS 2020/21

Prof. Dr. Michael Rohs, Tim Dünte, M.Sc.

Übungsblatt 1

Alle Übungen (bis auf die erste) müssen in Zweiergruppen bearbeitet werden. Beide Gruppenmitglieder müssen die Lösung der Zweiergruppe einzeln abgeben. Die Namen beider Gruppenmitglieder müssen sowohl in der PDF Abgabe als auch als Kommentar in jeglichen Quelltextabgaben genannt werden. Wenn Sie für Übungsblatt 1 noch keinen Gruppenpartner haben, geben Sie alleine ab und nutzen Sie das erste Online-Tutorium dazu, mit Hilfe des Tutors einen Partner zu finden. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 22.10. um 23:59 Uhr über https://assignments.hci.uni-hannover.de/WiSe2020/Programmieren1. Die Abgabe muss aus einer einzelnen zip-Datei bestehen, die den Quellcode, ein PDF bei Freitextaufgaben und alle weiteren nötigen Dateien (z.B. Eingabedaten oder Makefiles) enthält. Lösen Sie Umlaute in Dateinamen bitte auf.

Aufgabe 1: PostFix Entwicklungsumgebung

Die PostFix Entwicklungsumgebung (integrated development environment, IDE) ist verfügbar unter: https://postfix.hci.uni-hannover.de.

Öffnen Sie die PostFix IDE in einem Webbrowser Ihrer Wahl und führen Sie als Test folgendes Programm in der REPL und im Editor aus: `"Hello {Ihr Name}" println`

Lesen Sie das Tutorial zu PostFix (https://postfix.hci.uni-hannover.de/postfix-lang.html) bis einschließlich zum Abschnitt „Conditionals". Probieren Sie die Beispiele aus und machen Sie sich mit der Entwicklungsumgebung vertraut. Sie werden sie in den nächsten Übungen benötigen.

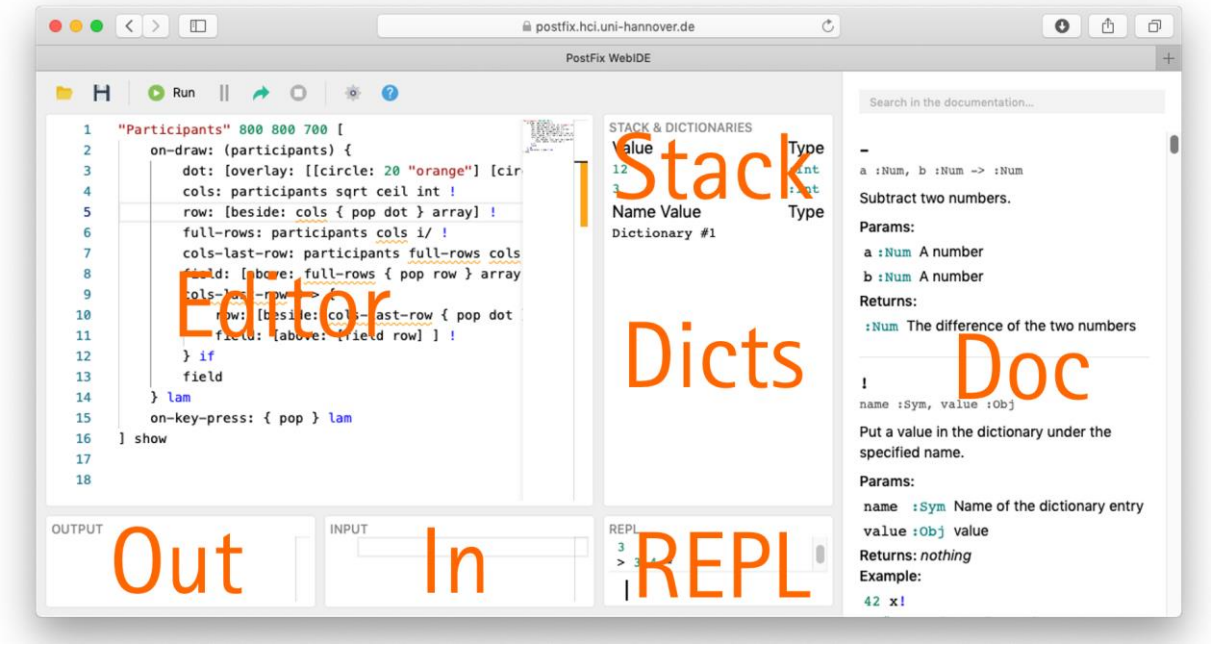

Abbildung 1: Die PostFix Entwicklungsumgebung.

a) Wie in Abbildung 1 zu sehen ist, hat die PostFix IDE sechs Bereiche: Editor, REPL, In, Out, Stack & Dicts und Doc. Geben Sie in eigenen Worten wieder, wofür die einzelnen Bereiche in der IDE nützlich sind. Geben Sie (wenn möglich) Beispielcodefragmente (nicht aus der Vorlesung) an, um die Funktionen zu erklären. Speichern Sie Ihre Codefragmente ab und fügen Sie sie der Abgabe bei.

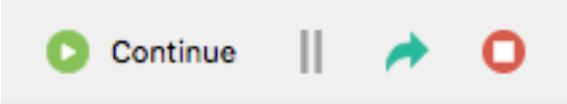

Abbildung 2: Schaltflächen zum Steuern der Programmausführung.

b) Im oberen Bereich der IDE finden sich vier Schaltflächen zum Steuern der Programmausführung (siehe Abbildung 2). Beschreiben Sie kurz, was diese Schaltflächen bewirken.

c) Ein Rechtsklick mit der Maus im Editor-Bereich öffnet ein Kontextmenü. Beschreiben Sie kurz, was die Menüpunkte „Add Conditional Breakpoint", „Toggle Breakpoint" und „Command Palette" bewirken.

Aufgabe 2: PostFix

Sollten Sie die Kommandos in der PostFix IDE testen, stellen Sie sicher, dass der Stack vor Ausführung jeder Teilaufgabe geleert ist. Rufen Sie dazu die Funktion `clear` in der REPL auf, um den Stack zu leeren.

a) Was passiert bei der Ausführung des folgenden Programms: `5 4 *`

b) Was steht auf dem Stack, nachdem das Programm `4 4 3 +` ausgeführt wurde und warum?

c) Wofür sind die Operatoren `read-int, read-flt, print, println` zuständig (in eigenen Worten)?

d) Beschreiben Sie die Operatoren `or` und `and`. Geben Sie jeweils ein Beispiel.

e) Welche der folgenden Ausdrücke sind äquivalent (hinsichtlich des Ergebnisses):

(1) `1 2 3 4 + + +`

(2) `1 2 + 3 4 + +`

(3) `1 2 + 3 + 4 +`

(4) `3 4 + 1 2 + +`

(5) `4 3 2 1 + + +`

Gilt dies auch für den Operator `-` (Subtraktion)? Begründen Sie kurz.

f) Parsen Sie das folgende Code Fragment per Hand und stellen Sie den Zustand des Stacks nach jedem gelesenen und verarbeiteten Token dar: `8 2 + -1 =`

Nutzen Sie als Hilfe das PostFix Execution Model aus der Vorlesung.

g) Was passiert, wenn Sie (nach `clear`) lediglich `1 /` in die REPL eingeben?

h) Führen Sie Schritt für Schritt die folgenden beiden Programme aus und beschreiben Sie kurz in eigenen Worten die Unterschiede in Ablauf und Ergebnis:
Programm 1: `{ 2 3 * }`
Programm 2: `[ 2 3 * ]`

i) Erläutern Sie inwiefern das Programm `0.1 0.2 + 0.3 =` zu einem unerwarteten Ergebnis führt.

Aufgabe 3: Intervalltest

a) Implementieren Sie ein Programm in PostFix, das bei Eingabe einer ganzen Zahl prüft, ob diese sich in dem Intervall zwischen -10 und 0 befindet (-10 und 0 gehören mit zum Intervall). Das Programm soll entsprechend true oder false ausgeben, falls die eingegebene Zahl im Intervall ist oder nicht. Folgende Befehle können hierbei hilfreich sein:
   - `read-int` : liest eine ganze Zahl ein
   - `println` : Ausgabe
   - `true, false` : Wahrheitswerte
   - `dup` : dupliziert das oberste Element auf dem Stack und legt es auf den Stack
   - `<, >, <=, >=` : relationale Operatoren
   - `and, or, not` : logische Operatoren
   - `Wahrheitswert {dies} {das} if` : Wenn Wahrheitswert dann mache dies ansonsten das

b) Ändern Sie das obige Programm so um, dass die Eingabe aus einer Variablen mit dem Namen „test“ gelesen wird und das Ergebnis in eine Variable mit dem Namen „result“ geschrieben wird. Weisen Sie der Variablen Test zu Beginn des Programms einen entsprechenden Wert zu.