Übungsblatt 5

Wichtig: Abgabe der Konsantenfaltung

Bitte denken Sie daran, Ihre Implementierung der Konstantenfaltung bis spätestens **26.11.** fertigzustellen und die entsprechend markierte Revision auf den `git`-Server des Lehrstuhls zu pushen.

Aufgabe 5.1: AST-Transformationen: Schleifen

Für bestimmte Programmoptimierungen ist es notwendig bzw. hilfreich, kopfgesteuerte Schleifen (`while`) in semantisch äquivalente fußgesteuerte Schleifen (`do/while`) zu übersetzen (mehr dazu in *Optimierungen in Übersetzern* im Sommer ☺). Überlegen Sie sich entsprechende AST-Transformationsregeln.

Welche beiden Möglichkeiten gibt es für die Erzeugung der neuen AST-Knoten? Was versteht man unter *semantischer Nachattributierung* und warum ist dies notwendig?

Aufgabe 5.2: AST-Transformationen: Variablendeklarationen mit Initialisierung

Die Sprache `e2` soll um Variablendeklarationen mit Initialisierung erweitert werden:

```
var a : int := 13;
var b : int := 3 * a;
```

Überlegen Sie sich, wie Sie den Übersetzer möglichst einfach erweitern können.

Achtung: Es gibt (mindestens) zwei Schwierigkeiten, an die man vielleicht nicht sofort denkt... ☺

Aufgabe 5.3: Transformation *Innerer Klassen* in Java

In der Vorlesung haben Sie kennengelernt, wie der Java-Compiler *innere Klassen* übersetzt. Im Rahmen dieser Aufgabe sollen Sie die dazu notwendigen Schritte anhand eines Beispiels nachvollziehen.

a) Laden Sie sich das Beispiel zu inneren Klassen von der Webseite zur Übung herunter (siehe auch unten).

b) Überlegen Sie sich, wie der Java-Übersetzer das Beispiel übersetzt: Wie werden innere Klassen transformiert, wie Zugriffe auf private Variablen?

c) Überprüfen Sie Ihre Überlegungen, indem Sie den Java-Disassembler `javap` oder einen Java-Decompiler (z.B. *Procyon*[1], `https://bitbucket.org/mstrobel/procyon`) verwenden.

Aufgabe 5.4: Transformation von *Generics* in Java

In der Vorlesung haben Sie kennengelernt, wie der Java-Compiler *Generics* übersetzt. Im Rahmen dieser Aufgabe sollen Sie die dazu notwendigen Schritte anhand eines Beispiels nachvollziehen.

a) Laden Sie sich die beiden Beispiele zu Generics von der Webseite zur Übung herunter (siehe auch unten).

b) Überlegen Sie sich, wie der Java-Übersetzer die beiden Beispiele übersetzt: Wie werden Klassen mit Typparametern in Klassen ohne Typparameter übersetzt? Wann sind Brückenmethoden notwendig und wie werden sie verwendet?

c) Überprüfen Sie Ihre Überlegungen, indem Sie den Java-Disassembler `javap` oder einen Java-Decompiler (z.B. *Procyon*[1], `https://bitbucket.org/mstrobel/procyon`) verwenden.

[1] Für Procyon benötigte Kommandozeilenargumente: `--exclude-nested --show-synthetic --retain-explicit-casts`.

```
public final class TransformationInnerClasses {

  public int a;
  private int b;

  private class InnerClass {

    private int x = a;
    private int y = b;

    public int foo() {
      return a + b;
    }

  }

  public void bar(final int p) {

    class InBar {

      public int buzz() {
        return p;
      }

    }

    (new InBar()).buzz();

    (new InnerClass()).foo();
  }

}
```

```
final class Pair<F, S> {

  private F first;
  private S second;

  public final void setFirst(final F first) {
    this.first = first;
  }

  public final F getFirst() {
    return this.first;
  }

  public final void setSecond(final S second) {
    this.second = second;
  }

  public final S getSecond() {
    return this.second;
  }

}

public final class TransformationGenerics {

  public static final void main(
      final String[] args) {
    final Pair<String, Integer> pair =
        new Pair<>();

    pair.setFirst("Sinn␣des␣Lebens");
    pair.setSecond(42);

    final String first = pair.getFirst();
    final Integer second = pair.getSecond();
  }

}
```

```
import java.util.Arrays;
import java.util.Comparator;

public final class IntComparator implements Comparator<Integer> {

  @Override
  public final int compare(final Integer a, final Integer b) {
    if (a < b) {
      return -1;
    }

    if (a > b) {
      return 1;
    }

    return 0;
  }

  public static void main(String[] args) {
    final Integer[] values = {13, 3, 12, 11};
    Arrays.sort(values, new IntComparator());
    System.out.println(Arrays.toString(values));
  }

}
```