Physikalisch–Technische Lehranstalt Wedel
Prof. Dr. Uwe Schmidt

5. Februar 1998

Aufgaben zur Klausur **Objektorientierte Programmierung** im WS 97/98 (IA42)

Zeit: 60 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 9 Seiten

---

Aufgabe 1:

Verarbeitung von arithmetischen Ausdrücken mit **Java**:

Es soll eine Klassenhierarchie zur Verarbeitung von arithmetischen Ausdrücken entwickelt werden. Der Einfachheit halber sollen hier nur Ausdrücke über den ganzen Zahlen betrachtet werden.

Die Wurzelklasse diese Klassenhierarchie sei die Klasse *Expr*:

```
//—————————

public
abstract
class Expr {

  // die eval Funktion

  public
  abstract
  int eval();

}

//—————————
```

Die einzige Verarbeitungsmethode ist hier das Auswerten eines Ausdrucks, die *eval*–Funktion.

Arithmetische Ausdrücke sind im einfachsten Fall Konstante, in dieser Aufgabe also ganze Zahlen. Für Konstante wird eine Klasse *Const* benötigt. Geben sie eine sinnvolle Definition für die *Const*–Klasse an:

Zusammengesetzte Ausdrücke sind z.B. einstellige und zweistellige Ausdrücke. In dieser Aufgabe sollen nur Ausdrücke der Form $e_1 + e_2$ und $e_1 * e_2$ behandelt werden. Entwickeln Sie sinnvolle Definitionen für allgemein verwendbare Klassen $AddExpr$ für Addition und $MultExpr$ für die Multiplikation. Gemeinsame Teile der Klassendefinition sind gegebenenfalls zusammenzufassen.

(*AddExpr* und *MultExpr* cont.)

Erweitern Sie die Klassenhierarchie so, daß auch Subtraktionen verarbeitet werden können.

Aufgabe 2:

Gegeben sei das folgende Testprogramm:

```
public
class ExcTest {
  public
  static
  void main(String[] argv) {
    int i = 0;
    try {
      i = Integer.parseInt(argv[0]);
    }
    catch ( Exception e ) { }
    test(i);
  }

  public
  static
  void test(int i) {
  tryblock:
    try {
      switch ( i ) {
      case 1:
        System.out.println("test:  call foo");
        foo();
        break;
      case 2:
        System.out.println("test:  call wow");
        wow();
        break;
      default:
        System.out.println("test:  normal exit");
      }
    }
    catch ( MyException e ) {
      System.out.println(
        "test:  caught MyException");
    }
    catch ( Exception e ) {
      System.out.println("test:  caught Exception");
    }
    finally {
      System.out.println("test:  exec finally");
    }
    System.out.println("test:  normal exit");
  }
```

```
  public
  static
  void foo()
    throws MyException {
      try {
        System.out.println(
          "foo :  throw MyException");
        throw
          new MyException("foo test");
      }
      finally {
        System.out.println("foo :  exec finally");
      }
  }

  public static
  void wow()
    throws MyException {
      int a, b=1, c=0;
      try {
        System.out.println(
          "wow :  division by 0");
        a = b/c;
      }
      catch (Exception e) {
        System.out.println(
          "wow :  caught Exception");
        System.out.println(
          "wow :  throw MyException");

        throw
          new MyException("wow test");
      }
  }
}

class MyException extends Exception {
  public
  MyException(String s) {
    super(s);
  }
}
```

Welche Ausgabe wird bei einem Aufruf von `java ExcTest 1` erzeugt:

.....

.....

.....

.....

.....

.....

.....

Welche Ausgabe wird bei einem Aufruf von `java ExcTest 2` erzeugt:

.....

.....

.....

.....

.....

.....

.....
