```
1   // Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas
2
3   package jminusminus;
4
5   /**
6    * The IDefn type is used to implement definitions of those things (local
7    * variables, formal arguments, types) that are named in some context (or
8    * scope).
9    */
10
11  interface IDefn {
12
13      /**
14       * The (local variable, formal parameter, or local or imported name)
15       * definition's type.
16       *
17       * @return the definition's type.
18       */
19
20      public Type type();
21
22  }
23
24  /**
25   * A definition of a type name. In the first instance, an identifier, but later
26   * resolved to a local name or an imported name.
27   */
28
29  class TypeNameDefn implements IDefn {
30
31      /** The definition's type. */
32      private Type type;
33
34      /**
35       * Construct a type name definition given its type.
36       *
37       * @param type
38       *            the definition's type.
39       */
40
41      public TypeNameDefn(Type type) {
42          this.type = type;
43      }
44
45      /**
46       * The type for this definition.
47       *
48       * @return the definition's type.
49       */
50
51      public Type type() {
52          return type;
53      }
54
55  }
56
57  /**
58   * The definition for a local variable (including formal parameters). All local
59   * variables are allocated on the stack at fixed offsets from the base of the
60   * stack frame, and all have types. Some local variables have initializations.
61   */
62
63  class LocalVariableDefn implements IDefn {
64
65      /** The local variable's type. */
66      private Type type;
```

```
67
68      /**
69       * The local variable's offset from the base of the current the stack frame.
70       */
71      private int offset;
72
73      /** Has this local variable been initialized? */
74      private boolean isInitialized;
75
76      /**
77       * Construct a local variable definition for a local variable.
78       *
79       * @param type
80       *            the variable's type.
81       * @param offset
82       *            the variable's offset from the base of the current stack frame
83       *            (allocated for each method invocation.)
84       */
85
86      public LocalVariableDefn(Type type, int offset) {
87          this.type = type;
88          this.offset = offset;
89      }
90
91      /**
92       * The type for this variable.
93       *
94       * @return the type.
95       */
96
97      public Type type() {
98          return type;
99      }
100
101     /**
102      * The offset of this variable on the stack frame.
103      *
104      * @return the offset.
105      */
106
107     public int offset() {
108         return offset;
109     }
110
111     /**
112      * Initialize this local variable.
113      */
114
115     public void initialize() {
116         this.isInitialized = true;
117     }
118
119     /**
120      * Has this local variable been initialized?
121      *
122      * @return true or false.
123      */
124
125     public boolean isInitialized() {
126         return isInitialized;
127     }
128
129 }
130
```