```
  1 //===========================================================================
  2 //      * Letsch Informatik *      www.LetsInfo.ch      CH-8636 Wald
  3 //      Beratung, Ausbildung und Realisation in Software-Engineering
  4 //===========================================================================
  5 // Project   : Master of Advanced Studies in Software-Engineering / 2008
  6 // Modul     : Java - Advanced Concepts
  7 // Title     : Übung 3: "Referenzen und AspectJ" / Aufgabe 2: "AspectJ"
  8 // Author    : Thomas Letsch
  9 // Tab-Width : 2
 10 /*///=========================================================================
 11 * Description: AspectJ und Logging.
 12 * History    : 23.03.07: Initial Version.
 13 *            : 24.01.08: Kosmetik.
 14 * CVS        : $Revision: 1.3 $  $Date: 2008/01/27 15:01:06 $
 15 /*///=========================================================================
 16 //     1         2         3         4         5         6         7         8
 17 //345678901234567890123456789012345678901234567890123456789012345678901234567890
 18 //===========================================================================
 19
 20 import java.util.logging.FileHandler;
 21 import java.util.logging.Formatter;
 22 import java.util.logging.Level;
 23 import java.util.logging.LogRecord;
 24 import java.util.logging.Logger;
 25 import java.util.logging.SimpleFormatter;
 26
 27
 28
 29 public aspect Logging {
 30
 31
 32   private static Logger mLogger;
 33   private static StringBuffer mBuffer;
 34
 35
 36   Logging() {
 37     mLogger = Logger.getLogger("LoggingDemo");
 38     mLogger.setUseParentHandlers(false);  // no Output anymore to Console.
 39     mLogger.setLevel(Level.ALL);
 40     try {
 41       FileHandler fileHandler = new FileHandler("Logging.txt");
 42       Formatter formatter = new MyFormatter();
 43       fileHandler.setFormatter(formatter);
 44       mLogger.addHandler(fileHandler);
 45     } catch (Exception pE) {
 46       pE.printStackTrace();
 47     }
 48     mBuffer = new StringBuffer();
 49   }
 50
 51
 52
 53   /**
 54    * Pointcut "allMethods": captures all Methods in BinarySearchTree
 55    * (or in Test-Class ;-)
 56    */
 57   pointcut allMethods(): execution (* BinarySearchTree.*(..)) || call (* Test.*(..)
    +);
 58
 59
 60
 61   /**
 62    * Advice "before(): allMethods()": generates the Log-Message.
 63    */
 64   before(): allMethods() {
 65     mBuffer.setLength(0);
 66     mBuffer.append(thisJoinPointStaticPart.getSignature().getName()+"(");
 67     Object[] args = thisJoinPoint.getArgs();
 68     for (int i = 0; i < args.length; i++) {
 69       mBuffer.append(args[i]);
 70       if (i < args.length-1) {
 71         mBuffer.append(", ");
 72       }
 73     }
 74     mBuffer.append(")");
 75     mLogger.log(Level.INFO, mBuffer.toString());
 76   }
 77
 78 } // End of aspect Logging
 79
 80
 81 /**
 82  * We want only one Line per Log-Event with only the Log-Message
 83  * (no TimeStamp etc.).
 84  * So we extend the SimpleFormatter and overwrite format().
 85  */
 86 class MyFormatter extends SimpleFormatter {
 87   private StringBuffer mBuffer = new StringBuffer();
 88   @Override
 89   public String format(LogRecord pRecord) {
 90     mBuffer.setLength(0);
 91     mBuffer.append(pRecord.getMessage());
 92     mBuffer.append('\n');
 93     return mBuffer.toString();
 94   }
 95 }
 96
 97
 98
 99
100
101 /* Log-File 'Logging.txt' of BinarySearchTree:
102
103 main([Ljava.lang.String;@750159)
104 maxDepth()
105 maxDepth(BinarySearchTree$Node@16cd7d5)
106 insert(5, Fünf:1)
107 search(5, BinarySearchTree$Node@16cd7d5)
108 printInorder()
109 traverseInorder(BinarySearchTree$Node@16cd7d5, [])
110 traverseInorder(BinarySearchTree$Node@1632c2d, [])
111 traverseInorder(BinarySearchTree$Node@1e97676, [Fünf:1])
112 hasNext()
113 next()
114 hasNext()
115 maxDepth()
116 maxDepth(BinarySearchTree$Node@16cd7d5)
117 maxDepth(BinarySearchTree$Node@1632c2d)
118 maxDepth(BinarySearchTree$Node@1e97676)
119 insert(3, Drei)
120 search(3, BinarySearchTree$Node@16cd7d5)
121 search(3, BinarySearchTree$Node@1632c2d)
122 printInorder()
123 traverseInorder(BinarySearchTree$Node@16cd7d5, [])
124 traverseInorder(BinarySearchTree$Node@1632c2d, [])
125 ...
126
127 */
```