VariableEntry.java

```java
/****
 *
 * VariableEntry extends SymbolTableEntry by adding data fields to support
 * variables and parameters.  It has a boolean field indicating if this is a
 * reference-type symbol.  Reference-type symbols are definable in programming
 * languages with explicitly declared pointer types and/or call-by-reference
 * parameters.
 *                                                                         <p>
 * VariableEntry also has an integer memory location field.  This can be either
 * an absolute address, or a relative offset, e.g., in a stack frame.
 *
 */
public class VariableEntry extends SymbolTableEntry {

    /**
     * Construct this with null data fields.
     */
    public VariableEntry() {
    }

    /**
     * Construct this with the given data field values.
     */
    public VariableEntry(String name, TypeNode type, boolean isRef,
            int memoryLocation, int level) {
        super(name, type);
        this.isRef = isRef;
        this.memoryLocation = memoryLocation;
        this.level = level;
    }

    /**
     * Return the string rep of this, which consists of the return value of
     * super.toString, plus the values of this.isRef and this.memoryLocation.
     */
    public String toString(int level) {
        return super.toString(level) + ", is ref: " + isRef + ", mem loc: " +
            Integer.toString(memoryLocation);
    }

    /** True if this is a reference variable or parameter. */
    public boolean isRef;

    /** Memory location */
    public int memoryLocation;

    /** The lexical nesting level of this variable.  This is a convenience for
     * computing the runtime address of the variable.  If the level = 0, then
     * the address is relative to the top of the static pool.  If the level is
     * > 0, then the address is relative to the current top of the stack. */
    public int level;

}
```