

CLConstantPool.java

```
1  // Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas
2
3  package jminusminus;
4
5  import java.io.IOException;
6  import java.util.ArrayList;
7
8  /**
9   * Representation of a class' constant_pool table (JVM Spec Section 4.5). An
10   * instance of CLConstantPool is created when a class is read using CLAbsorber
11   * or constructed using CLEmitter.
12   */
13
14  class CLConstantPool {
15
16      // The fields below represent the members of the
17      // constant_pool structure. See JVM Spec Section 4.5 for details.
18
19      /** Index of the next item into the constant pool. */
20      private int cpIndex;
21
22      /** List of constant pool items. */
23      private ArrayList<CLCPInfo> cpItems;
24
25      /**
26       * Look for the specified item in the constant pool. If it exists, return
27       * its index. Otherwise, add the item to the constant pool and return its
28       * (new) index.
29       *
30       * @param cpInfo
31       *       the item to find or add
32       * @return index constant pool index of the item.
33       */
34
35      private int findOrAdd(CLCPInfo cpInfo) {
36          int index = find(cpInfo);
37          if (index == -1) {
38              index = addCPItem(cpInfo);
39          }
40          return index;
41      }
42
43      /**
44       * Construct a CLConstantPool object.
45       */
46
47      public CLConstantPool() {
48          cpIndex = 1;
49          cpItems = new ArrayList<CLCPInfo>();
50      }
51
52      /**
53       * Return the size of the constant pool.
54       *
55       * @return the size of the constant pool.
56       */
57
58      public int size() {
59          return cpItems.size();
60      }
61
62      /**
63       * Return the constant pool index of the specified item if it exists in the
64       * pool, -1 otherwise.
65       *
66       * @param cpInfo
```

```

67         *         item to find.
68         * @return the constant pool index or -1.
69         */
70
71     public int find(CLCPInfo cpInfo) {
72         int index = cpItems.indexOf(cpInfo);
73         if (index != -1) {
74             CLCPInfo c = cpItems.get(index);
75             index = c.cpIndex;
76         }
77         return index;
78     }
79
80     /**
81      * Return the constant pool item at the specified index, or null if the
82      * index is invalid.
83      *
84      * @param i
85      *         constant pool index.
86      * @return the constant pool item or null.
87      */
88
89     public CLCPInfo cpItem(int i) {
90         if (((i - 1) < 0) || ((i - 1) >= cpItems.size())) {
91             return null;
92         }
93         return cpItems.get(i - 1);
94     }
95
96     /**
97      * Add the specified (non null) item to the constant pool table and return
98      * its index.
99      *
100     * @param cpInfo
101     *        the item to add to the constant pool table.
102     * @return constant pool index of the item.
103     */
104
105     public int addCpItem(CLCPInfo cpInfo) {
106         int i = cpIndex++;
107         cpInfo.cpIndex = i;
108         cpItems.add(cpInfo);
109
110         // long and double, with their lower and higher words,
111         // are treated by JVM as two items in the constant pool. We
112         // have a single representation for each, so we add a null as
113         // a placeholder in the second slot.
114         if ((cpInfo instanceof CLConstantLongInfo)
115             || (cpInfo instanceof CLConstantDoubleInfo)) {
116             cpIndex++;
117             cpItems.add(null);
118         }
119         return i;
120     }
121
122     /**
123      * Write the contents of the constant_pool to the specified output stream.
124      *
125      * @param out
126      *         output stream.
127      * @throws IOException
128      *         if an error occurs while writing.
129      */
130
131     public void write(CLOutputStream out) throws IOException {
132         for (int i = 0; i < cpItems.size(); i++) {
133             CLCPInfo cpInfo = cpItems.get(i);
134             if (cpInfo != null) {
135                 cpInfo.write(out);

```

```

136     }
137 }
138 }
139
140 /**
141  * Write the contents of the constant pool to STDOUT in a format similar to
142  * that of javap.
143  *
144  * @param p
145  *         for pretty printing with indentation.
146  */
147
148 public void writeToStdOut(PrettyPrinter p) {
149     p.printf("// Constant Pool (%s Items)\n", size());
150     p.printf("%-10s%-20s%\n", "Index", "Item Type", "Content");
151     p.printf("%-10s%-20s%\n", "-----", "-----", "-----");
152     for (int i = 1; i <= size(); i++) {
153         CLCPInfo cpInfo = cpItem(i);
154         if (cpInfo != null) {
155             cpInfo.writeToStdOut(p);
156         }
157     }
158 }
159
160 // Following methods are helpers for creating singleton
161 // instances of the different constant pool items -- classes extending
162 // CLCPInfo objects in our representation. Each method
163 // accepts as arguments the information needed for creating a
164 // particular constant pool item, creates an instance
165 // of the item, checks if the item already exists in the
166 // constant pool, adds it if not, and returns the (possibly new) index
167 // of the item.
168
169 /**
170  * Return the constant pool index of a singleton instance of
171  * CLConstantClassInfo.
172  *
173  * @param s
174  *         class or interface name in internal form.
175  * @return constant pool index.
176  */
177
178 public int constantClassInfo(String s) {
179     CLCPInfo c = new CLConstantClassInfo(constantUtf8Info(s));
180     return findOrAdd(c);
181 }
182
183 /**
184  * Return the constant pool index of a singleton instance of
185  * CLConstantFieldRefInfo.
186  *
187  * @param className
188  *         class or interface name in internal form.
189  * @param name
190  *         name of the field.
191  * @param type
192  *         descriptor of the field.
193  * @return constant pool index.
194  */
195
196 public int constantFieldRefInfo(String className, String name, String type) {
197     CLCPInfo c = new CLConstantFieldRefInfo(constantClassInfo(className),
198         constantNameAndTypeInfo(name, type));
199     return findOrAdd(c);
200 }
201
202 /**
203  * Return the constant pool index of a singleton instance of
204  * CLConstantMethodRefInfo.

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

205 *
206 * @param className
207 *         class or interface name in internal form.
208 * @param name
209 *         name of the method.
210 * @param type
211 *         descriptor of the method.
212 * @return constant pool index.
213 */
214
215 public int constantMethodRefInfo(String className, String name, String type)
216 {
217     CLCPInfo c = new CLConstantMethodRefInfo(constantClassInfo(className),
218         constantNameAndTypeInfo(name, type));
219     return findOrAdd(c);
220 }
221 /**
222 * Return the constant pool index of a singleton instance of
223 * CLConstantInterfaceMethodRefInfo.
224 *
225 * @param className
226 *         class or interface name in internal form.
227 * @param name
228 *         name of the method.
229 * @param type
230 *         descriptor of the method.
231 * @return constant pool index.
232 */
233 public int constantInterfaceMethodRefInfo(String className, String name,
234     String type) {
235     CLCPInfo c = new CLConstantInterfaceMethodRefInfo(
236         constantClassInfo(className), constantNameAndTypeInfo(name,
237             type));
238     return findOrAdd(c);
239 }
240
241 /**
242 * Return the constant pool index of a singleton instance of
243 * CLConstantStringInfo.
244 *
245 * @param s
246 *         the constant string value.
247 * @return constant pool index.
248 */
249
250 public int constantStringInfo(String s) {
251     CLCPInfo c = new CLConstantStringInfo(constantUtf8Info(s));
252     return findOrAdd(c);
253 }
254
255 /**
256 * Return the constant pool index of a singleton instance of
257 * CLConstantIntegerInfo.
258 *
259 * @param i
260 *         the constant int value.
261 * @return constant pool index.
262 */
263
264 public int constantIntegerInfo(int i) {
265     CLCPInfo c = new CLConstantIntegerInfo(i);
266     return findOrAdd(c);
267 }
268
269 /**
270 * Return the constant pool index of a singleton instance of
271 * CLConstantFloatInfo.
272 */

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

273 *
274 * @param f
275 *         the constant floating-point value.
276 * @return constant pool index.
277 */
278
279 public int constantFloatInfo(float f) {
280     CLCPInfo c = new CLConstantFloatInfo(f);
281     return findOrAdd(c);
282 }
283
284 /**
285 * Return the constant pool index of a singleton instance of
286 * CLConstantLongInfo.
287 *
288 * @param l
289 *         the constant long value.
290 * @return constant pool index.
291 */
292
293 public int constantLongInfo(long l) {
294     CLCPInfo c = new CLConstantLongInfo(l);
295     return findOrAdd(c);
296 }
297
298 /**
299 * Return the constant pool index of a singleton instance of
300 * CLConstantDoubleInfo.
301 *
302 * @param d
303 *         the constant double value.
304 * @return constant pool index.
305 */
306
307 public int constantDoubleInfo(double d) {
308     CLCPInfo c = new CLConstantDoubleInfo(d);
309     return findOrAdd(c);
310 }
311
312 /**
313 * Return the constant pool index of a singleton instance of
314 * CLConstantNameAndTypeInfo.
315 *
316 * @param name
317 *         field or method name.
318 * @param type
319 *         field or method type descriptor.
320 * @return constant pool index.
321 */
322
323 public int constantNameAndTypeInfo(String name, String type) {
324     CLCPInfo c = new CLConstantNameAndTypeInfo(constantUtf8Info(name),
325         constantUtf8Info(type));
326     return findOrAdd(c);
327 }
328
329 /**
330 * Return the constant pool index of a singleton instance of
331 * CLConstantUtf8Info.
332 *
333 * @param s
334 *         the constant string value.
335 * @return constant pool index.
336 */
337
338 public int constantUtf8Info(String s) {
339     CLCPInfo c = new CLConstantUtf8Info(s.getBytes());
340     return findOrAdd(c);
341 }

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

342
343 }
344

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder