

CLAttributeInfo.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 import static jminusminus.CLConstants.*;
8
9 /**
10  * Representation of attribute_info structure (JVM Spec Section 4.8). Classes
11  * representing individual attributes inherit this class. This file has
12  * representations for all attributes specified in JVM Spec Second Edition,
13  * including the ones that were added for JDK 1.5.
14  *
15  * Attributes are used in the ClassFile (CLFile), field_info (CLFieldInfo),
16  * method_info (CLMethodInfo), and Code_attribute (CLCodeAttribute) structures
17  * of the class file format. While there are many kinds of attributes, only some
18  * are mandatory; these include:
19  *
20  * InnerClasses_attribute (class attribute) Synthetic_attribute (class, field,
21  * and method attribute) Code_attribute (method attribute) Exceptions_attribute
22  * (method attribute)
23  *
24  * CLAbsorber is capable of reading all attributes listed in this file. The ones
25  * which it does not recognize, it simply skips them and reports a warning to
26  * that extent.
27  *
28  * CLEmitter implicitly adds the required attributes to the appropriate
29  * structure. The optional attributes have to be added explicitly using the
30  * addClassAttribute(), addFieldAttribute(), addMethodAttribute(), and
31  * addCodeAttribute() methods in CLEmitter.
32  */
33
34 abstract class CLAttributeInfo {
35
36     // The fields below represent the members of the
37     // attribute_info
38     // structure and are thus inherited by the child classes of
39     // CLAttributeInfo. These classes define their own fields
40     // (if any) representing the members of the individual
41     // attribute_info structures they represent.
42
43     /** attribute_info.attribute_name_index item. */
44     public int attributeNameIndex;
45
46     /** attribute_info.attribute_length item. */
47     public long attributeLength;
48
49     /**
50      * Construct an CLAttributeInfo object.
51      *
52      * @param attributeNameIndex
53      *        attribute_info.attribute_name_index item.
54      * @param attributeLength
55      *        attribute_info.attribute_length item.
56      */
57
58     protected CLAttributeInfo(int attributeNameIndex, long attributeLength) {
59         this.attributeNameIndex = attributeNameIndex;
60         this.attributeLength = attributeLength;
61     }
62
63     /**
64      * Write the contents of this attribute to the specified output stream.
65      *
66      * @param out
```

```

67         *           output stream.
68         * @throws IOException
69         *           if an error occurs while writing.
70         */
71
72     public void write(CLOutputStream out) throws IOException {
73         out.writeShort(attributeNameIndex);
74         out.writeInt((long) attributeLength);
75     }
76
77     /**
78     * Write the contents of this attribute to STDOUT in a format similar to
79     * that of javap.
80     *
81     * @param p
82     *         for pretty printing with indentation.
83     */
84
85     public void writeToStdOut(PrettyPrinter p) {
86         p.printf("Attribute Name Index: %s\n", attributeNameIndex);
87         p.printf("Attribute Length: %s\n", attributeLength);
88     }
89 }
90
91 /**
92 * Representation of ConstantValue_attribute structure (JVM Spec Section 4.8.2).
93 * This is a required field attribute.
94 */
95
96 class CLConstantValueAttribute extends CLAttributeInfo {
97
98     /** ConstantValue_attribute.constantvalue_index item. */
99     public int constantValueIndex;
100
101     /**
102     * Construct a CLConstantValueAttribute object.
103     *
104     * @param attributeNameIndex
105     *         ConstantValue_attribute.attribute_name_index item.
106     * @param attributeLength
107     *         ConstantValue_attribute.attribute_length item.
108     * @param constantValueIndex
109     *         ConstantValue_attribute.constantvalue_index item.
110     */
111
112     public CLConstantValueAttribute(int attributeNameIndex,
113         long attributeLength, int constantValueIndex) {
114         super(attributeNameIndex, attributeLength);
115         this.constantValueIndex = constantValueIndex;
116     }
117
118     /**
119     * @inheritDoc
120     */
121
122     public void write(CLOutputStream out) throws IOException {
123         super.write(out);
124         out.writeShort(constantValueIndex);
125     }
126
127     /**
128     * @inheritDoc
129     */
130
131     public void writeToStdOut(PrettyPrinter p) {
132         p.printf("ConstantValue {\n");
133         p.indentRight();
134         super.writeToStdOut(p);

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

136         p.printf("Constant Value Index: %s\n", constantValueIndex);
137         p.indentLeft();
138         p.printf("}\n");
139     }
140 }
141 }
142
143 /**
144  * Representation of exception_table entry structure (JVM Spec Section 4.8.3).
145  */
146
147 class CLEntryInfo {
148
149     /** exception_table_entry.start_pc item. */
150     public int startPC;
151
152     /** exception_table_entry.end_pc item. */
153     public int endPC;
154
155     /** exception_table_entry.handler_pc item. */
156     public int handlerPC;
157
158     /** exception_table_entry.catch_type item. */
159     public int catchType;
160
161     /**
162      * Construct a CLEntryInfo object.
163      *
164      * @param startPC
165      *     exception_table_entry.start_pc item.
166      * @param endPC
167      *     exception_table_entry.end_pc item.
168      * @param handlerPC
169      *     exception_table_entry.handler_pc item.
170      * @param catchType
171      *     exception_table_entry.catch_type item.
172      */
173
174     public CLEntryInfo(int startPC, int endPC, int handlerPC, int catchType)
175     {
176         this.startPC = startPC;
177         this.endPC = endPC;
178         this.handlerPC = handlerPC;
179         this.catchType = catchType;
180     }
181
182     /**
183      * Write the contents of this object to the specified output stream.
184      *
185      * @param out
186      *     output stream.
187      * @throws IOException
188      *     if an error occurs while writing.
189      */
190
191     public void write(CLOutputStream out) throws IOException {
192         out.writeShort(startPC);
193         out.writeShort(endPC);
194         out.writeShort(handlerPC);
195         out.writeShort(catchType);
196     }
197
198     /**
199      * Write the contents of this object to STDOUT in a format similar to that
200      * of javap.
201      *
202      * @param p
203      *     for pretty printing with indentation.
204      */

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

204
205     public void writeToStdOut(PrettyPrinter p) {
206         p.printf("%-8s    %-6s    %-10s    %-10s\n", startPC, endPC, handlerPC,
207                 catchType);
208     }
209
210 }
211
212 /**
213  * Representation of Code_attribute structure (JVM Spec Section 4.8.2). This is
214  * a required method attribute.
215  */
216
217 class CLCodeAttribute extends CLAttributeInfo {
218
219     /** Code_attribute.max_stack item. */
220     public int maxStack;
221
222     /** Code_attribute.max_locals item. */
223     public int maxLocals;
224
225     /** Code_attribute.code_length item. */
226     public long codeLength;
227
228     /**
229      * Code_attribute.code item.
230      */
231     public ArrayList<Integer> code;
232
233     /** Code_attribute.exception_table_length item. */
234     public int exceptionTableLength;
235
236     /** Code_attribute.exception_table item. */
237     public ArrayList<CLExceptionInfo> exceptionTable;
238
239     /** Code_attribute.attributes_count item. */
240     public int attributesCount;
241
242     /** Code_attribute.attributes item. */
243     public ArrayList<CLAttributeInfo> attributes;
244
245     /**
246      * Construct and return an integer from the four unsigned bytes specified.
247      *
248      * @param a
249      *         unsigned byte.
250      * @param b
251      *         unsigned byte.
252      * @param c
253      *         unsigned byte.
254      * @param d
255      *         unsigned byte.
256      * @return an integer constructed from the four unsigned bytes specified.
257      */
258
259     private int intValue(int a, int b, int c, int d) {
260         return (a << 24) | (b << 16) | (c << 8) | d;
261     }
262
263     /**
264      * Construct a CLCodeAttribute object.
265      *
266      * @param attributeNameIndex
267      *         Code_attribute.attribute_name_index item.
268      * @param attributeLength
269      *         Code_attribute.attribute_length item.
270      * @param maxStack
271      *         Code_attribute.max_stack item.
272      * @param maxLocals

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

273     *         Code_attribute.max_locals item.
274     * @param codeLength
275     *         Code_attribute.code_length item.
276     * @param code
277     *         Code_attribute.code item.
278     * @param exceptionTableLength
279     *         Code_attribute.exception_table_length item.
280     * @param exceptionTable
281     *         Code_attribute.exception_table item.
282     * @param attributesCount
283     *         Code_attribute.attributes_count item.
284     * @param attributes
285     *         Code_attribute.attributes item.
286     */
287
288     public CLCodeAttribute(int attributeNameIndex, long attributeLength,
289         int maxStack, int maxLocals, long codeLength,
290         ArrayList<Integer> code, int exceptionTableLength,
291         ArrayList<CLExceptionInfo> exceptionTable, int attributesCount,
292         ArrayList<CLAttributeInfo> attributes) {
293         super(attributeNameIndex, attributeLength);
294         this.maxStack = maxStack;
295         this.maxLocals = maxLocals;
296         this.codeLength = codeLength;
297         this.code = code;
298         this.exceptionTableLength = exceptionTableLength;
299         this.exceptionTable = exceptionTable;
300         this.attributesCount = attributesCount;
301         this.attributes = attributes;
302     }
303
304     /**
305     * @inheritDoc
306     */
307
308     public void write(CLOutputStream out) throws IOException {
309         super.write(out);
310         out.writeShort(maxStack);
311         out.writeShort(maxLocals);
312         out.writeInt(codeLength);
313         for (int i = 0; i < code.size(); i++) {
314             out.writeByte(code.get(i));
315         }
316         out.writeShort(exceptionTableLength);
317         for (int i = 0; i < exceptionTable.size(); i++) {
318             exceptionTable.get(i).write(out);
319         }
320         out.writeShort(attributesCount);
321         for (int i = 0; i < attributes.size(); i++) {
322             attributes.get(i).write(out);
323         }
324     }
325
326     /**
327     * @inheritDoc
328     */
329
330     public void writeToStdOut(PrettyPrinter p) {
331         p.printf("Code {\n");
332         p.indentRight();
333         super.writeToStdOut(p);
334         p.printf("Max Stack: %s\n", maxStack);
335         p.printf("Max Locals: %s\n", maxLocals);
336         p.printf("Code Length: %s\n", codeLength);
337         p.printf("%-10s%-17s%s\n", "PC", "Opcode", "Operands");
338         p.printf("%-10s%-17s%s\n", "---", "-----", "-----");
339         for (int i = 0; i < code.size(); i++) {
340             int pc = i;
341             int opcode = code.get(i);

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

342         String mnemonic = CLInstruction.instructionInfo[opcode].mnemonic;
343         int operandBytes =
CLInstruction.instructionInfo[opcode].operandCount;
344         short operandByte1, operandByte2, operandByte3, operandByte4;
345         int pad, deflt;
346         switch (operandBytes) {
347         case 0:
348             p.printf("%-10s%-17s\n", pc, mnemonic);
349             break;
350         case 1:
351             operandByte1 = code.get(++i).shortValue();
352             p.printf("%-10s%-17s%-5s\n", pc, mnemonic, operandByte1);
353             break;
354         case 2:
355             operandByte1 = code.get(++i).shortValue();
356             operandByte2 = code.get(++i).shortValue();
357             p.printf("%-10s%-17s%-5s%-5s\n", pc, mnemonic, operandByte1,
358                 operandByte2);
359             break;
360         case 3:
361             operandByte1 = code.get(++i).shortValue();
362             operandByte2 = code.get(++i).shortValue();
363             operandByte3 = code.get(++i).shortValue();
364             p.printf("%-10s%-17s%-5s%-5s%-5s\n", pc, mnemonic,
365                 operandByte1, operandByte2, operandByte3);
366             break;
367         case 4:
368             operandByte1 = code.get(++i).shortValue();
369             operandByte2 = code.get(++i).shortValue();
370             operandByte3 = code.get(++i).shortValue();
371             operandByte4 = code.get(++i).shortValue();
372             p.printf("%-10s%-17s%-5s%-5s%-5s%-5s\n", pc, mnemonic,
373                 operandByte1, operandByte2, operandByte3, operandByte4);
374             break;
375         case DYNAMIC: // variable length instructions
376             if (opcode == TABLESWITCH) {
377                 int low, high;
378                 pad = 4 - ((i + 1) % 4);
379                 i = i + pad + 1;
380                 deflt = intValue(code.get(i++), code.get(i++), code
381                     .get(i++), code.get(i++));
382                 low = intValue(code.get(i++), code.get(i++), code.get(i++),
383                     code.get(i++));
384                 high = intValue(code.get(i++), code.get(i++),
385                     code.get(i++), code.get(i));
386                 p.printf("%-10s%5s { // %s to %s \n", pc, mnemonic, low,
387                     high);
388                 for (int idx = low; idx <= high; idx++) {
389                     int offset = intValue(code.get(++i), code.get(++i),
390                         code.get(++i), code.get(++i));
391                     p.printf("%-10s    %s:%s\n", "", idx, offset);
392                 }
393                 p.printf("%-10s    default: %s\n", "", deflt);
394                 p.printf("%-10s}\n", "");
395             } else { // LOOKUPSWITCH
396                 int nPairs;
397                 pad = 4 - ((i + 1) % 4);
398                 i = i + pad + 1;
399                 deflt = intValue(code.get(i++), code.get(i++), code
400                     .get(i++), code.get(i++));
401                 nPairs = intValue(code.get(i++), code.get(i++), code
402                     .get(i++), code.get(i));
403                 p.printf("%-10s%5s { \n", pc, mnemonic);
404                 for (int idx = 0; idx < nPairs; idx++) {
405                     int match = intValue(code.get(++i), code.get(++i), code
406                         .get(++i), code.get(++i));
407                     int offset = intValue(code.get(++i), code.get(++i),
408                         code.get(++i), code.get(++i));
409                     p.printf("%-10s    %s:%s\n", "", match, offset);

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

410         }
411         p.printf("%-10s    default: %s\n", "", deflt);
412         p.printf("%-10s}\n", "");
413     }
414 }
415 }
416 p.println();
417 p.printf("// Exception Table (%s Items)\n", exceptionTableLength);
418 p.printf("%s    %s    %s    %s\n", "Start PC", "End PC", "Handler PC",
419         "Catch Type");
420 p.printf("%s    %s    %s    %s\n", "-----", "-----", "-----",
421         "-----");
422 for (int i = 0; i < exceptionTable.size(); i++) {
423     exceptionTable.get(i).writeToStdOut(p);
424 }
425 p.println();
426 p.printf("// Attributes (%s Items)\n", attributesCount);
427 for (int i = 0; i < attributes.size(); i++) {
428     CLAttributeInfo attributeInfo = attributes.get(i);
429     attributeInfo.writeToStdOut(p);
430 }
431 p.indentLeft();
432 p.printf("}\n");
433 }
434 }
435 }
436
437 /**
438  * Representation of Exceptions attribute structure (JVM Spec Section 4.8.4).
439  * This is a required method attribute.
440  */
441
442 class CLEExceptionsAttribute extends CLAttributeInfo {
443
444     /** Exceptions_attribute.number_of_exceptions item. */
445     public int numberOfExceptions;
446
447     /** Exceptions_attribute.exception_index_table item. */
448     public ArrayList<Integer> exceptionIndexTable;
449
450     /**
451      * Construct a CLEExceptionsAttribute object.
452      *
453      * @param attributeNameIndex
454      *         Exceptions_attribute.attribute_name_index item.
455      * @param attributeLength
456      *         Exceptions_attribute.attribute_length item.
457      * @param numberOfExceptions
458      *         Exceptions_attribute.number_of_exceptions item.
459      * @param exceptionIndexTable
460      *         Exceptions_attribute.exception_index_table item.
461      */
462
463     public CLEExceptionsAttribute(int attributeNameIndex, long attributeLength,
464         int numberOfExceptions, ArrayList<Integer> exceptionIndexTable) {
465         super(attributeNameIndex, attributeLength);
466         this.numberOfExceptions = numberOfExceptions;
467         this.exceptionIndexTable = exceptionIndexTable;
468     }
469
470     /**
471      * @inheritDoc
472      */
473
474     public void write(CLOutputStream out) throws IOException {
475         super.write(out);
476         out.writeShort(numberOfExceptions);
477         for (int i = 0; i < exceptionIndexTable.size(); i++) {
478             out.writeShort(exceptionIndexTable.get(i));

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder


```

479     }
480 }
481
482 /**
483  * @inheritDoc
484  */
485
486 public void writeToStdOut(PrettyPrinter p) {
487     p.printf("Exceptions {\n");
488     p.indentRight();
489     super.writeToStdOut(p);
490     p.printf("Number of Exceptions: %s\n", numberOfExceptions);
491
492     // Get rid of the [] in the toString() value of the
493     // exceptionIndexTable ArrayList.
494     String exceptions = exceptionIndexTable.toString();
495     exceptions = exceptions.substring(1, exceptions.length() - 2);
496
497     p.printf("Exception Index Table: %s\n", exceptions);
498     p.indentLeft();
499     p.printf("}\n");
500 }
501
502 }
503
504 /**
505  * Representation of classes table entry structure (JVM Spec Section 4.8.5).
506  */
507
508 class CLInnerClassInfo {
509     /** classes_table_entry.inner_class_info_index item. */
510     public int innerClassInfoIndex;
511     /** classes_table_entry.outer_class_info_index item. */
512     public int outerClassInfoIndex;
513     /** classes_table_entry.inner_name_index item. */
514     public int innerNameIndex;
515     /** classes_table_entry.inner_class_access_flags item. */
516     public int innerClassAccessFlags;
517
518     /**
519      * Construct a CLInnerClassInfo object.
520      *
521      * @param innerClassInfoIndex
522      *         classes_table_entry.inner_class_info_index item.
523      * @param outerClassInfoIndex
524      *         classes_table_entry.outer_class_info_index item.
525      * @param innerNameIndex
526      *         classes_table_entry.inner_name_index item.
527      * @param innerClassAccessFlags
528      *         classes_table_entry.inner_class_access_flags item.
529      */
530
531     public CLInnerClassInfo(int innerClassInfoIndex, int outerClassInfoIndex,
532         int innerNameIndex, int innerClassAccessFlags) {
533         this.innerClassInfoIndex = innerClassInfoIndex;
534         this.outerClassInfoIndex = outerClassInfoIndex;
535         this.innerNameIndex = innerNameIndex;
536         this.innerClassAccessFlags = innerClassAccessFlags;
537     }
538
539     /**
540      * Write the contents of this object to the specified output stream.
541      *
542      * @param out
543      *         output stream.
544      */
545 }

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder


```

548     * @throws IOException
549     *         if an error occurs while writing.
550     */
551
552     public void write(CLOutputStream out) throws IOException {
553         out.writeShort(innerClassInfoIndex);
554         out.writeShort(outerClassInfoIndex);
555         out.writeShort(innerNameIndex);
556         out.writeShort(innerClassAccessFlags);
557     }
558
559     /**
560     * Write the contents of this object to STDOUT in a format similar to that
561     * of javap.
562     *
563     * @param p
564     *         for pretty printing with indentation.
565     */
566
567     public void writeToStdOut(PrettyPrinter p) {
568         p.printf("%-11s    %-17s    %-10s    %s\n", innerClassInfoIndex,
569             outerClassInfoIndex, innerNameIndex, CLFile
570                 .innerClassAccessFlagsToString(innerClassAccessFlags));
571     }
572 }
573
574
575 /**
576  * Representation of InnerClasses_attribute structure (JVM Spec Section 4.8.5).
577  * This is a required class attribute.
578  *
579  * Note that this is just to register the inner classes with its parent class,
580  * and does not create the classes, which can be done using CLEmitter.
581  */
582
583 class CLInnerClassesAttribute extends CLAttributeInfo {
584
585     /** InnerClasses_attribute.number_of_classes item. */
586     public int numberOfClasses;
587
588     /** InnerClasses_attribute.classes item. */
589     public ArrayList<CLInnerClassInfo> classes;
590
591     /**
592     * Construct a CLInnerClassesAttribute object.
593     *
594     * @param attributeNameIndex
595     *         InnerClasses_attribute.attribute_name_index item.
596     * @param attributeLength
597     *         InnerClasses_attribute.attribute_length item.
598     * @param numberOfClasses
599     *         InnerClasses_attribute.number_of_classes item.
600     * @param classes
601     *         InnerClasses_attribute.classes item.
602     */
603
604     public CLInnerClassesAttribute(int attributeNameIndex,
605         long attributeLength, int numberOfClasses,
606         ArrayList<CLInnerClassInfo> classes) {
607         super(attributeNameIndex, attributeLength);
608         this.numberOfClasses = numberOfClasses;
609         this.classes = classes;
610     }
611
612     /**
613     * @inheritDoc
614     */
615
616     public void write(CLOutputStream out) throws IOException {

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

617         super.write(out);
618         out.writeShort(numberOfClasses);
619         for (int i = 0; i < classes.size(); i++) {
620             classes.get(i).write(out);
621         }
622     }
623
624     /**
625      * @inheritDoc
626      */
627
628     public void writeToStdOut(PrettyPrinter p) {
629         p.printf("InnerClassesAttribute {\n");
630         p.indentRight();
631         super.writeToStdOut(p);
632         p.printf("Number of Classes: %s\n", numberOfClasses);
633         p.printf("%s    %s    %s    %s\n", "Class Index", "Outer Class Index",
634             "Name Index", "Access Flags");
635         p.printf("%s    %s    %s    %s\n", "-----", "-----",
636             "-----", "-----");
637         for (int i = 0; i < classes.size(); i++) {
638             classes.get(i).writeToStdOut(p);
639         }
640         p.indentLeft();
641         p.printf("}\n");
642     }
643
644 }
645
646 /**
647  * Representation of EnclosingMethodAttribute structure (JVM Spec Section
648  * 4.8.6).
649  */
650
651 class CLEnclosingMethodAttribute extends CAttributeInfo {
652
653     /** EnclosingMethod_attribute.class_index item. */
654     public int classIndex;
655
656     /** EnclosingMethod_attribute.method_index item. */
657     public int methodIndex;
658
659     /**
660      * Construct a CLEnclosingMethodAttribute object.
661      *
662      * @param attributeNameIndex
663      *         EnclosingMethod_attribute.attribute_name_index item.
664      * @param attributeLength
665      *         EnclosingMethod_attribute.attribute_length item.
666      * @param classIndex
667      *         EnclosingMethod_attribute.class_index item.
668      * @param methodIndex
669      *         EnclosingMethod_attribute.method_index item.
670      */
671
672     public CLEnclosingMethodAttribute(int attributeNameIndex,
673         long attributeLength, int classIndex, int methodIndex) {
674         super(attributeNameIndex, attributeLength);
675         this.classIndex = classIndex;
676         this.methodIndex = methodIndex;
677     }
678
679     /**
680      * @inheritDoc
681      */
682
683     public void write(CLOutputStream out) throws IOException {
684         super.write(out);
685         out.writeShort(classIndex);

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

686         out.writeShort(methodIndex);
687     }
688
689     /**
690     * @inheritDoc
691     */
692
693     public void writeToStdOut(PrettyPrinter p) {
694         p.printf("EnclosingMethod {\n");
695         p.indentRight();
696         super.writeToStdOut(p);
697         p.printf("Class Index: %s\n", classIndex);
698         p.printf("Method Index: %s\n", methodIndex);
699         p.indentLeft();
700         p.printf("}\n");
701     }
702
703 }
704
705 /**
706  * Representation of Synthetic_attribute structure (JVM Spec Section 4.8.7).
707  * This is a required class, field, and method attribute.
708  */
709
710 class CLSyntheticAttribute extends CLAttributeInfo {
711
712     /**
713     * Construct a CLSyntheticAttribute object.
714     * @param attribute_name_index Synthetic_attribute.attribute_name_index item
715     * @param attribute_length Synthetic_attribute.attribute_length item.
716     */
717     public CLSyntheticAttribute(int attributeNameIndex, long attributeLength) {
718         super(attributeNameIndex, attributeLength);
719     }
720
721     /**
722     * @inheritDoc
723     */
724
725     public void write(CLOutputStream out) throws IOException {
726         super.write(out);
727     }
728
729     /**
730     * @inheritDoc
731     */
732
733     public void writeToStdOut(PrettyPrinter p) {
734         p.printf("Synthetic {\n");
735         p.indentRight();
736         super.writeToStdOut(p);
737         p.indentLeft();
738         p.printf("}\n");
739     }
740
741 }
742
743 /**
744  * Representation of Signature_attribute structure (JVM Spec Section 4.8.8).
745  */
746
747 class CLSignatureAttribute extends CLAttributeInfo {
748
749     /** Signature_attribute.signature_index item. */
750     public int signatureIndex;

```

```

755
756 /**
757  * Construct a CLSignatureAttribute object.
758  *
759  * @param attributeNameIndex
760  *         Signature_attribute.attribute_name_index item.
761  * @param attributeLength
762  *         Signature_attribute.attribute_length item.
763  * @param signatureIndex
764  *         Signature_attribute.signature_index item.
765  */
766
767 public CLSignatureAttribute(int attributeNameIndex, long attributeLength,
768                             int signatureIndex) {
769     super(attributeNameIndex, attributeLength);
770     this.signatureIndex = signatureIndex;
771 }
772
773 /**
774  * @inheritDoc
775  */
776
777 public void write(CLOutputStream out) throws IOException {
778     super.write(out);
779     out.writeShort(signatureIndex);
780 }
781
782 /**
783  * @inheritDoc
784  */
785
786 public void writeToStdOut(PrettyPrinter p) {
787     p.printf("Signature {\n");
788     p.indentRight();
789     super.writeToStdOut(p);
790     p.printf("Signature Index: %s\n", signatureIndex);
791     p.indentLeft();
792     p.printf("}\n");
793 }
794
795 }
796
797 /**
798  * Representation of SourceFile_attribute structure (JVM Spec Section 4.8.9).
799  */
800
801 class CLSourceFileAttribute extends CLAttributeInfo {
802
803     /** SourceFile_attribute.sourcefile_index item. */
804     public int sourceFileIndex;
805
806     /**
807      * Construct a CLSourceFileAttribute object.
808      *
809      * @param attributeNameIndex
810      *         SourceFile_attribute.attribute_name_index item.
811      * @param attributeLength
812      *         SourceFile_attribute.attribute_length item.
813      * @param sourceFileIndex
814      *         SourceFile_attribute.sourcefile_index item.
815      */
816
817     public CLSourceFileAttribute(int attributeNameIndex, long attributeLength,
818                                 int sourceFileIndex) {
819         super(attributeNameIndex, attributeLength);
820         this.sourceFileIndex = sourceFileIndex;
821     }
822
823     /**

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

824     * @inheritDoc
825     */
826
827     public void write(CLOutputStream out) throws IOException {
828         super.write(out);
829         out.writeShort(sourceFileIndex);
830     }
831
832     /**
833     * @inheritDoc
834     */
835
836     public void writeToStdOut(PrettyPrinter p) {
837         p.printf("SourceFile {\n");
838         p.indentRight();
839         super.writeToStdOut(p);
840         p.printf("Source File Index: %s\n", sourceFileIndex);
841         p.indentLeft();
842         p.printf("}\n");
843     }
844 }
845
846 /**
847  * Representation of SourceDebugExtension_attribute structure (JVM Spec Section
848  * 4.8.10).
849  */
850
851
852 class CLSourceDebugExtensionAttribute extends CLAttributeInfo {
853     /** SourceDebugExtension.debug_extension item. */
854     public byte[] debugExtension;
855
856     /**
857     * Construct a CLSourceDebugExtensionAttribute object.
858     *
859     * @param attributeNameIndex
860     *         SourceDebugExtension_attribute.attribute_name_index item.
861     * @param attributeLength
862     *         SourceDebugExtension_attribute.attribute_length item.
863     * @param debugExtension
864     *         SourceDebugExtension_attribute.debug_extension item.
865     */
866
867     public CLSourceDebugExtensionAttribute(int attributeNameIndex,
868         long attributeLength, byte[] debugExtension) {
869         super(attributeNameIndex, attributeLength);
870         this.debugExtension = debugExtension;
871     }
872
873     /**
874     * @inheritDoc
875     */
876
877     public void write(CLOutputStream out) throws IOException {
878         super.write(out);
879         for (int i = 0; i < debugExtension.length; i++) {
880             out.writeByte(debugExtension[i]);
881         }
882     }
883
884     /**
885     * @inheritDoc
886     */
887
888     public void writeToStdOut(PrettyPrinter p) {
889         p.printf("SourceDebugExtension {\n");
890         p.indentRight();
891         super.writeToStdOut(p);
892     }

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

893         p.printf("Debug Extension: %s\n", new String(debugExtension));
894         p.indentLeft();
895         p.printf("}\n");
896     }
897 }
898 }
899
900 /**
901  * Representation of line_number_table entry structure (JVM Spec Section
902  * 4.8.11).
903  */
904
905 class CLLineNumberInfo {
906
907     /** line_number_table_entry.start_pc item. */
908     public int startPC;
909
910     /** line_number_table_entry.line_number item. */
911     public int lineNumber;
912
913     /**
914      * Construct a CLLineNumberInfo object.
915      *
916      * @param startPC
917      *         line_number_table_entry.start_pc item.
918      * @param lineNumber
919      *         line_number_table_entry.line_number item.
920      */
921
922     public CLLineNumberInfo(int startPC, int lineNumber) {
923         this.startPC = startPC;
924         this.lineNumber = lineNumber;
925     }
926
927     /**
928      * Write the contents of this object to the specified output stream.
929      *
930      * @param out
931      *         output stream.
932      * @throws IOException
933      *         if an error occurs while writing.
934      */
935
936     public void write(CLOutputStream out) throws IOException {
937         out.writeShort(startPC);
938         out.writeShort(lineNumber);
939     }
940
941     /**
942      * Return true if this LineNumber_info object is "equal to" the specified
943      * LineNumber_info object, false otherwise.
944      *
945      * @param obj
946      *         the reference LineNumber_info object with which to compare.
947      * @return true if this LineNumber_info object is "equal to" the specified
948      *         LineNumber_info object, false otherwise.
949      */
950
951     public boolean equals(Object obj) {
952         if (obj instanceof CLLineNumberInfo) {
953             CLLineNumberInfo c = (CLLineNumberInfo) obj;
954             if (c.lineNumber == lineNumber) {
955                 return true;
956             }
957         }
958         return false;
959     }
960 }
961
962 /**

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

962     * Write the contents of this object to STDOUT in a format similar to that
963     * of javap.
964     *
965     * @param p
966     *         for pretty printing with indentation.
967     */
968
969     public void writeToStdOut(PrettyPrinter p) {
970         p.printf("%-8s    %-11s\n", startPC, lineNumber);
971     }
972
973 }
974
975 /**
976  * Representation of LineNumberTable_attribute structure (JVM Spec Section
977  * 4.8.11).
978  */
979
980 class CLLineNumberTableAttribute extends CLAttributeInfo {
981
982     /** LineNumberTable_attribute.line_number_table_length item. */
983     public int lineNumberTableLength;
984
985     /** LineNumberTable_attribute.line_number_table item. */
986     public ArrayList<CLLineNumberOfInfo> lineNumberTable;
987
988     /**
989      * Construct a CLLineNumberTableAttribute object.
990      *
991      * @param attribute_name_index
992      *         LineNumberTable_attribute.attribute_name_index item.
993      * @param attributeLength
994      *         LineNumberTable_attribute.attribute_length item.
995      * @param lineNumberTableLength
996      *         LineNumberTable_attribute.line_number_table_length item.
997      * @param lineNumberTable
998      *         LineNumberTable_attribute.line_number_table item.
999      */
1000
1001     public CLLineNumberTableAttribute(int attribute_name_index,
1002         long attributeLength, int lineNumberTableLength,
1003         ArrayList<CLLineNumberOfInfo> lineNumberTable) {
1004         super(attribute_name_index, attributeLength);
1005         this.lineNumberTableLength = lineNumberTableLength;
1006         this.lineNumberTable = lineNumberTable;
1007     }
1008
1009     /**
1010      * @inheritDoc
1011      */
1012
1013     public void write(CLOutputStream out) throws IOException {
1014         super.write(out);
1015         out.writeShort(lineNumberTableLength);
1016         for (int i = 0; i < lineNumberTable.size(); i++) {
1017             lineNumberTable.get(i).write(out);
1018         }
1019     }
1020
1021     /**
1022      * @inheritDoc
1023      */
1024
1025     public void writeToStdOut(PrettyPrinter p) {
1026         p.printf("LineNumberTable {\n");
1027         p.indentRight();
1028         super.writeToStdOut(p);
1029         p.printf("Line Number Table Length: %s\n", lineNumberTableLength);
1030         p.printf("%s    %s\n", "Start PC", "Line Number");

```



```

1031         p.printf("%s    %s\n", "-----", "-----");
1032         for (int i = 0; i < lineNumberTable.size(); i++) {
1033             lineNumberTable.get(i).writeToStdOut(p);
1034         }
1035         p.indentLeft();
1036         p.printf("}\n");
1037     }
1038 }
1039 }
1040
1041 /**
1042  * Representation of local_variable_table entry structure (JVM Spec Section
1043  * 4.8.12).
1044  */
1045
1046 class CLLocalVariableInfo {
1047
1048     /** local_variable_table_entry.start_pc item. */
1049     public int startPC;
1050
1051     /** local_variable_table_entry.length item. */
1052     public int length;
1053
1054     /** local_variable_table_entry.name_index item. */
1055     public int nameIndex;
1056
1057     /** local_variable_table_entry.descriptor_index item. */
1058     public int descriptorIndex;
1059
1060     /** local_variable_table_entry.index item. */
1061     public int index;
1062
1063     /**
1064      * Construct a CLLocalVariableInfo object.
1065      *
1066      * @param startPC
1067      *         local_variable_table_entry.start_pc item.
1068      * @param length
1069      *         local_variable_table_entry.length item.
1070      * @param nameIndex
1071      *         local_variable_table_entry.name_index item.
1072      * @param descriptorIndex
1073      *         local_variable_table_entry.descriptor_index item.
1074      * @param index
1075      *         local_variable_table_entry.index item.
1076      */
1077
1078     public CLLocalVariableInfo(int startPC, int length, int nameIndex,
1079                               int descriptorIndex, int index) {
1080         this.startPC = startPC;
1081         this.length = length;
1082         this.nameIndex = nameIndex;
1083         this.descriptorIndex = descriptorIndex;
1084         this.index = index;
1085     }
1086
1087     /**
1088      * Write the contents of this object to the specified output stream.
1089      *
1090      * @param out
1091      *         output stream.
1092      * @throws IOException
1093      *         if an error occurs while writing.
1094      */
1095
1096     public void write(CLOutputStream out) throws IOException {
1097         out.writeShort(startPC);
1098         out.writeShort(length);
1099         out.writeShort(nameIndex);

```

```

1100         out.writeShort(descriptorIndex);
1101         out.writeShort(index);
1102     }
1103
1104     /**
1105      * Write the contents of this object to STDOUT in a format similar to that
1106      * of javap.
1107      *
1108      * @param p
1109      *         for pretty printing with indentation.
1110      */
1111
1112     public void writeToStdOut(PrettyPrinter p) {
1113         p.printf("%-8s    %-6s    %-10s    %-16s    %-5s\n", startPC, length,
1114                 nameIndex, descriptorIndex, index);
1115     }
1116
1117 }
1118
1119 /**
1120  * Representation of LocalVariableTable_attribute structure (JVM Spec Section
1121  * 4.8.12).
1122  */
1123
1124 class CLLocalVariableTableAttribute extends CLAttributeInfo {
1125
1126     /**
1127      * LocalVariableTable_attribute.local_variable_table_length item.
1128      */
1129     public int localVariableTableLength;
1130
1131     /** LocalVariableTable_attribute.local_variable_table item. */
1132     public ArrayList<CLLocalVariableInfo> localVariableTable;
1133
1134     /**
1135      * Construct a CLLocalVariableTableAttribute object.
1136      *
1137      * @param attributeNameIndex
1138      *        LocalVariableTable_attribute.attribute_name_index item.
1139      * @param attributeLength
1140      *        LocalVariableTable_attribute.attribute_length item.
1141      * @param localVariableTableLength
1142      *        LocalVariableTable_attribute.local_variable_table_length item.
1143      * @param localVariableTable
1144      *        LocalVariableTable_attribute.local_variable_table item.
1145      */
1146
1147     public CLLocalVariableTableAttribute(int attributeNameIndex,
1148                                         long attributeLength, int localVariableTableLength,
1149                                         ArrayList<CLLocalVariableInfo> localVariableTable) {
1150         super(attributeNameIndex, attributeLength);
1151         this.localVariableTableLength = localVariableTableLength;
1152         this.localVariableTable = localVariableTable;
1153     }
1154
1155     /**
1156      * @inheritDoc
1157      */
1158
1159     public void write(CLOutputStream out) throws IOException {
1160         super.write(out);
1161         out.writeShort(localVariableTableLength);
1162         for (int i = 0; i < localVariableTable.size(); i++) {
1163             localVariableTable.get(i).write(out);
1164         }
1165     }
1166
1167     /**
1168      * @inheritDoc

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1169     */
1170
1171     public void writeToStdOut(PrettyPrinter p) {
1172         p.printf("LocalVariableTable {\n");
1173         p.indentRight();
1174         super.writeToStdOut(p);
1175         p.printf("Local Variable Table Length: %s\n", localVariableTableLength);
1176         p.printf("%s    %s    %s    %s    %s\n", "Start PC", "Length",
1177             "Name Index", "Descriptor Index", "Index");
1178         p.printf("%s    %s    %s    %s    %s\n", "-----", "-----",
1179             "-----", "-----", "-----");
1180         for (int i = 0; i < localVariableTable.size(); i++) {
1181             localVariableTable.get(i).writeToStdOut(p);
1182         }
1183         p.indentLeft();
1184         p.printf("}\n");
1185     }
1186
1187 }
1188
1189 /**
1190  * Representation of local_variable_type_table entry structure (JVM Spec Section
1191  * 4.8.13).
1192  */
1193
1194 class CLLocalVariableTypeInfo {
1195
1196     /** local_variable_type_table_entry.start_pc item. */
1197     public int startPC;
1198
1199     /** local_variable_type_table_entry.length item. */
1200     public int length;
1201
1202     /** local_variable_type_table_entry.name_index item. */
1203     public int nameIndex;
1204
1205     /** local_variable_type_table_entry.descriptor_index item. */
1206     public int signatureIndex;
1207
1208     /** local_variable_type_table_entry.index item. */
1209     public int index;
1210
1211     /**
1212      * Construct a CLLocalVariableTypeInfo object.
1213      *
1214      * @param startPC
1215      *         local_variable_type_table_entry.start_pc item.
1216      * @param length
1217      *         local_variable_type_table_entry.length item.
1218      * @param nameIndex
1219      *         local_variable_type_table_entry.name_index item.
1220      * @param signatureIndex
1221      *         local_variable_type_table_entry.signature_index item.
1222      * @param index
1223      *         local_variable_type_table_entry.index item.
1224      */
1225
1226     public CLLocalVariableTypeInfo(int startPC, int length, int nameIndex,
1227         int signatureIndex, int index) {
1228         this.startPC = startPC;
1229         this.length = length;
1230         this.nameIndex = nameIndex;
1231         this.signatureIndex = signatureIndex;
1232         this.index = index;
1233     }
1234
1235     /**
1236      * Write the content of this object to the specified output stream.
1237      */

```

```

1238     * @param out
1239     *         output stream.
1240     * @throws IOException
1241     *         if an error occurs while writing.
1242     */
1243
1244     public void write(CLOutputStream out) throws IOException {
1245         out.writeShort(startPC);
1246         out.writeShort(length);
1247         out.writeShort(nameIndex);
1248         out.writeShort(signatureIndex);
1249         out.writeShort(index);
1250     }
1251
1252     /**
1253     * Write the contents of this object to STDOUT in a format similar to that
1254     * of javap.
1255     *
1256     * @param p
1257     *         for pretty printing with indentation.
1258     */
1259
1260     public void writeToStdOut(PrettyPrinter p) {
1261         p.printf("%-8s    %-6s    %-10s    %-16s    %-5s\n", startPC, length,
1262             nameIndex, signatureIndex, index);
1263     }
1264
1265 }
1266
1267 /**
1268  * Representation of LocalVariableTypeTable_attribute structure (JVM Spec
1269  * Section 4.8.12).
1270  */
1271
1272 class CCLocalVariableTypeTableAttribute extends CCLAttributeInfo {
1273
1274     /**
1275     * LocalVariableTypeTable_attribute.local_variable_type_table_length item.
1276     */
1277     public int localVariableTypeTableLength;
1278
1279     /**
1280     * LocalVariableTypeTable_attribute.local_variable_type_table item.
1281     */
1282     public ArrayList<CCLocalVariableTypeInfo> localVariableTypeTable;
1283
1284     /**
1285     * Construct a CCLocalVariableTypeTableAttribute object.
1286     *
1287     * @param attributeNameIndex
1288     *         LocalVariableTypeTable_attribute.attribute_name_index item.
1289     * @param attributeLength
1290     *         LocalVariableTypeTable_attribute.attribute_length item.
1291     * @param localVariableTypeTableLength
1292     *         LocalVariableTypeTable_attribute.
1293     *         local_variable_type_table_length item.
1294     * @param localVariableTypeTable
1295     *         LocalVariableTypeTable_attribute.local_variable_type_table
1296     *         item.
1297     */
1298
1299     public CCLocalVariableTypeTableAttribute(int attributeNameIndex,
1300         long attributeLength, int localVariableTypeTableLength,
1301         ArrayList<CCLocalVariableTypeInfo> localVariableTypeTable) {
1302         super(attributeNameIndex, attributeLength);
1303         this.localVariableTypeTableLength = localVariableTypeTableLength;
1304         this.localVariableTypeTable = localVariableTypeTable;
1305     }
1306

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1307  /**
1308   * @inheritDoc
1309   */
1310
1311  public void write(CLOutputStream out) throws IOException {
1312      super.write(out);
1313      out.writeShort(localVariableTypeTableLength);
1314      for (int i = 0; i < localVariableTypeTable.size(); i++) {
1315          localVariableTypeTable.get(i).write(out);
1316      }
1317  }
1318
1319  /**
1320   * @inheritDoc
1321   */
1322
1323  public void writeToStdOut(PrettyPrinter p) {
1324      p.printf("LocalVariableTypeTable {\n");
1325      p.indentRight();
1326      super.writeToStdOut(p);
1327      p.printf("Local Variable Type Table Length: %s\n",
1328              localVariableTypeTableLength);
1329      p.printf("%s    %s    %s    %s    %s\n", "Start PC", "Length",
1330              "Name Index", "Signature Index", "Index");
1331      p.printf("%s    %s    %s    %s    %s\n", "-----", "-----",
1332              "-----", "-----", "-----");
1333      for (int i = 0; i < localVariableTypeTable.size(); i++) {
1334          localVariableTypeTable.get(i).writeToStdOut(p);
1335      }
1336      p.indentLeft();
1337      p.printf("}\n");
1338  }
1339
1340}
1341
1342/**
1343 * Representation of Deprecated_attribute structure (JVM Spec Section 4.8.14).
1344 */
1345
1346class CLDeprecatedAttribute extends CLAttributeInfo {
1347
1348    /**
1349     * Construct a CLDeprecatedAttribute object.
1350     *
1351     * @param attributeNameIndex
1352     *         Deprecated_attribute.attribute_name_index item.
1353     * @param attributeLength
1354     *         Deprecated_attribute.attribute_length item.
1355     */
1356
1357    public CLDeprecatedAttribute(int attributeNameIndex, long attributeLength) {
1358        super(attributeNameIndex, attributeLength);
1359    }
1360
1361    /**
1362     * @inheritDoc
1363     */
1364
1365    public void write(CLOutputStream out) throws IOException {
1366        super.write(out);
1367    }
1368
1369    /**
1370     * @inheritDoc
1371     */
1372
1373    public void writeToStdOut(PrettyPrinter p) {
1374        p.printf("Deprecated {\n");
1375        p.indentRight();

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1376         super.writeToStdOut(p);
1377         p.indentLeft();
1378         p.printf("}\n");
1379     }
1380 }
1381 }
1382
1383 /**
1384  * Representation of annotation structure (JVM Spec Section 4.8.15).
1385  */
1386
1387 class CLAnnotation {
1388
1389     /** annotation.type_index item. */
1390     public int typeIndex;
1391
1392     /** annotation.num_element_value_pairs item. */
1393     public int numElementValuePairs;
1394
1395     /** annotation.element_value_pairs item. */
1396     public ArrayList<CLElementValuePair> elementValuePairs;
1397
1398     /**
1399      * Construct a CLAnnotation object.
1400      *
1401      * @param typeIndex
1402      *         annotation.type_index item.
1403      * @param numElementValuePairs
1404      *         annotation.num_element_value_pairs item.
1405      * @param elementValuePairs
1406      *         annotation.element_value_pairs item.
1407      */
1408
1409     public CLAnnotation(int typeIndex, int numElementValuePairs,
1410                       ArrayList<CLElementValuePair> elementValuePairs) {
1411         this.typeIndex = typeIndex;
1412         this.numElementValuePairs = numElementValuePairs;
1413         this.elementValuePairs = elementValuePairs;
1414     }
1415
1416     /**
1417      * Write the contents of this object to the specified output stream.
1418      *
1419      * @param out
1420      *         output stream.
1421      * @throws IOException
1422      *         if an error occurs while writing.
1423      */
1424
1425     public void write(CLOutputStream out) throws IOException {
1426         out.writeShort(typeIndex);
1427         out.writeShort(numElementValuePairs);
1428         for (int i = 0; i < elementValuePairs.size(); i++) {
1429             elementValuePairs.get(i).write(out);
1430         }
1431     }
1432
1433     /**
1434      * Write the content of this object to STDOUT in a format similar to that of
1435      * javap.
1436      *
1437      * @param p
1438      *         for pretty printing with indentation.
1439      */
1440
1441     public void writeToStdOut(PrettyPrinter p) {
1442         p.printf("Annotation {\n");
1443         p.indentRight();
1444         p.printf("Type Index: %s\n", typeIndex);

```

```

1445         p.printf("Number of Element-Value Pairs: %s\n", numElementValuePairs);
1446         for (int i = 0; i < elementValuePairs.size(); i++) {
1447             elementValuePairs.get(i).writeToStdOut(p);
1448         }
1449         p.indentLeft();
1450         p.printf("}\n");
1451     }
1452 }
1453 }
1454
1455 /**
1456  * Representation of element_value union (JVM Spec Section 4.8.15.1).
1457  */
1458
1459 class CLElementValue {
1460
1461     /** element_value.tag item. */
1462     public short tag;
1463
1464     /** element_value.const_value_index item. */
1465     public int constValueIndex;
1466
1467     /** element_value.enum_const_value.type_name_index item. */
1468     public int typeNameIndex;
1469
1470     /** element_value.enum_const_value.const_name_index item. */
1471     public int constNameIndex;
1472
1473     /** element_value.class_info_index item. */
1474     public int classInfoIndex;
1475
1476     /** element_value.annotation_value item. */
1477     public CLAnnotation annotationValue;
1478
1479     /** element_value.array_value.numvalues item. */
1480     public int numValues;
1481
1482     /** element_value.array_value.values item. */
1483     public ArrayList<CLElementValue> values;
1484
1485     /**
1486      * Construct a CLElementValue object.
1487      *
1488      * @param tag
1489      *         element_value.tag item.
1490      * @param constValueIndex
1491      *         element_value.const_value_index item.
1492      */
1493
1494     public CLElementValue(short tag, int constValueIndex) {
1495         this.tag = tag;
1496         this.constValueIndex = constValueIndex;
1497     }
1498
1499     /**
1500      * Construct a CLElementValue object.
1501      *
1502      * @param typeNameIndex
1503      *         element_value.type_name_index item.
1504      * @param constNameIndex
1505      *         element_value.const_name_index item.
1506      */
1507
1508     public CLElementValue(int typeNameIndex, int constNameIndex) {
1509         this.tag = ELT_e;
1510         this.typeNameIndex = typeNameIndex;
1511         this.constNameIndex = constNameIndex;
1512     }
1513

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder


```

1514 /**
1515  * Construct a CLElementValue object.
1516  *
1517  * @param classInfoIndex
1518  *         element_value.class_info_index item.
1519  */
1520
1521 public CLElementValue(int classInfoIndex) {
1522     this.tag = ELT_c;
1523     this.classInfoIndex = classInfoIndex;
1524 }
1525
1526 /**
1527  * Construct a CLElementValue object.
1528  *
1529  * @param annotationValue
1530  *         element_value.annotation_value item.
1531  */
1532
1533 public CLElementValue(CAnnotation annotationValue) {
1534     this.tag = ELT_ANNOTATION;
1535     this.annotationValue = annotationValue;
1536 }
1537
1538 /**
1539  * Construct a CLElementValue object.
1540  *
1541  * @param numValues
1542  *         element_value.num_values.
1543  * @param values
1544  *         element_value.values.
1545  */
1546
1547 public CLElementValue(int numValues, ArrayList<CLElementValue> values) {
1548     this.tag = ELT_ARRAY;
1549     this.numValues = numValues;
1550     this.values = values;
1551 }
1552
1553 /**
1554  * Write the contents of this object to the specified output stream.
1555  *
1556  * @param out
1557  *         output stream.
1558  * @throws IOException
1559  *         if an error occurs while writing.
1560  */
1561
1562 public void write(CLOutputStream out) throws IOException {
1563     out.writeByte(tag);
1564     switch (tag) {
1565         case ELT_B:
1566         case ELT_C:
1567         case ELT_D:
1568         case ELT_F:
1569         case ELT_I:
1570         case ELT_J:
1571         case ELT_S:
1572         case ELT_Z:
1573         case ELT_s:
1574         out.writeInt(constValueIndex);
1575         break;
1576         case ELT_e:
1577         out.writeInt(typeNameIndex);
1578         out.writeInt(constNameIndex);
1579         break;
1580         case ELT_c:
1581         out.writeInt(classInfoIndex);
1582         break;

```

```

1583     case ELT_ANNOTATION:
1584         annotationValue.write(out);
1585         break;
1586     case ELT_ARRAY:
1587         out.writeInt(numValues);
1588         for (int i = 0; i < numValues; i++) {
1589             values.get(i).write(out);
1590         }
1591     }
1592 }
1593
1594 /**
1595  * Write the content of this object to STDOUT in a format similar to that of
1596  * javap.
1597  *
1598  * @param p
1599  *         for pretty printing with indentation.
1600  */
1601
1602 public void writeToStdOut(PrettyPrinter p) {
1603     p.printf("ElementValue {\n");
1604     p.indentRight();
1605     p.printf("Tag: %c\n", tag);
1606     switch (tag) {
1607         case ELT_B:
1608         case ELT_C:
1609         case ELT_D:
1610         case ELT_F:
1611         case ELT_I:
1612         case ELT_J:
1613         case ELT_S:
1614         case ELT_Z:
1615         case ELT_s:
1616             p.printf("Constant Value Index: %s\n", constValueIndex);
1617             break;
1618         case ELT_e:
1619             p.printf("Type Name Index: %s\n", typeNameIndex);
1620             p.printf("Constant Name Index: %s\n", constNameIndex);
1621             break;
1622         case ELT_c:
1623             p.printf("Class Info Index: %s\n", classInfoIndex);
1624             break;
1625         case ELT_ANNOTATION:
1626             annotationValue.writeToStdOut(p);
1627             break;
1628         case ELT_ARRAY:
1629             p.printf("Number of Values: %s\n", numValues);
1630             for (int i = 0; i < numValues; i++) {
1631                 values.get(i).writeToStdOut(p);
1632             }
1633     }
1634     p.indentLeft();
1635     p.printf("}\n");
1636 }
1637
1638 }
1639
1640 /**
1641  * Representation of the element_value_pairs table entry (JVM Spec Section
1642  * 4.8.15).
1643  */
1644
1645 class CLElementValuePair {
1646
1647     /** element_value_pairs_table_entry.element_name_index item. */
1648     public int elementNameIndex;
1649
1650     /** element_value_pairs_table_entry.value item. */
1651     public CLElementValue value;

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1652
1653 /**
1654  * Construct a CLElementValuePair object.
1655  *
1656  * @param elementNameIndex
1657  *         element_value_pairs_table_entry.element_name_index item.
1658  * @param value
1659  *         element_value_pairs_table_entry.value item.
1660  */
1661
1662 public CLElementValuePair(int elementNameIndex, CLElementValue value) {
1663     this.elementNameIndex = elementNameIndex;
1664     this.value = value;
1665 }
1666
1667 /**
1668  * Write the contents of this object to the specified output stream.
1669  *
1670  * @param out
1671  *         output stream.
1672  * @throws IOException
1673  *         if an error occurs while writing.
1674  */
1675
1676 public void write(CLOutputStream out) throws IOException {
1677     out.writeShort(elementNameIndex);
1678     value.write(out);
1679 }
1680
1681 /**
1682  * Write the content of this object to STDOUT in a format similar to that of
1683  * javap.
1684  *
1685  * @param p
1686  *         for pretty printing with indentation.
1687  */
1688
1689 public void writeToStdOut(PrettyPrinter p) {
1690     p.print("ElementValuePair\n");
1691     p.indentRight();
1692     p.printf("Element Name Index: %s\n", elementNameIndex);
1693     value.writeToStdOut(p);
1694     p.indentLeft();
1695     p.printf("}\n");
1696 }
1697
1698 }
1699
1700 /**
1701  * Representation of RuntimeVisibleAnnotations_attribute structure (JVM Spec
1702  * Section 4.8.15).
1703  */
1704
1705 class CLRuntimeVisibleAnnotationsAttribute extends CLAttributeInfo {
1706
1707     /** RuntimeVisibleAnnotations_attribute.num_annotations item. */
1708     public int numAnnotations;
1709
1710     /** RuntimeVisibleAnnotations_attribute.annotations item. */
1711     public ArrayList<CLAnnotation> annotations;
1712
1713     /**
1714      * Construct a CLRuntimeVisibleAnnotationsAttribute object.
1715      *
1716      * @param attributeNameIndex
1717      *         RuntimeVisibleAnnotations_attribute.attribute_name_index item.
1718      * @param attributeLength
1719      *         RuntimeVisibleAnnotations_attribute.attribute_length item.
1720      * @param numAnnotations

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1721     * RuntimeVisibleAnnotations_attribute.num_annotations item.
1722     * @param annotations
1723     * RuntimeVisibleAnnotations_attribute.annotations item.
1724     */
1725
1726     public CLRuntimeVisibleAnnotationsAttribute(int attributeNameIndex,
1727         long attributeLength, int numAnnotations,
1728         ArrayList<CLAnnotation> annotations) {
1729         super(attributeNameIndex, attributeLength);
1730         this.numAnnotations = numAnnotations;
1731         this.annotations = annotations;
1732     }
1733
1734     /**
1735     * @inheritDoc
1736     */
1737
1738     public void write(CLOutputStream out) throws IOException {
1739         super.write(out);
1740         out.writeShort(numAnnotations);
1741         for (int i = 0; i < annotations.size(); i++) {
1742             annotations.get(i).write(out);
1743         }
1744     }
1745
1746     /**
1747     * @inheritDoc
1748     */
1749
1750     public void write(PrintStream p) {
1751         p.print("RuntimeVisibleAnnotations {\n");
1752         p.indentRight();
1753         super.writeToStdOut(p);
1754         p.printf("Number of Annotations: %s\n", numAnnotations);
1755         for (int i = 0; i < annotations.size(); i++) {
1756             annotations.get(i).writeToStdOut(p);
1757         }
1758         p.indentLeft();
1759         p.print("}\n");
1760     }
1761 }
1762 }
1763
1764 /**
1765  * Representation of RuntimeInvisibleAnnotations_attribute structure (JVM Spec
1766  * Section 4.8.16).
1767  */
1768
1769 class CLRuntimeInvisibleAnnotationsAttribute extends CLAttributeInfo {
1770
1771     /**
1772     * RuntimeInvisibleAnnotations_attribute.num_annotations item.
1773     */
1774     public int numAnnotations;
1775
1776     /** RuntimeInvisibleAnnotations_attribute.annotations item. */
1777     public ArrayList<CLAnnotation> annotations;
1778
1779     /**
1780     * Construct a CLRuntimeInvisibleAnnotationsAttribute object.
1781     *
1782     * @param attributeNameIndex
1783     *         RuntimeInvisibleAnnotations_attribute.attribute_name_index
1784     *         item.
1785     * @param attributeLength
1786     *         RuntimeInvisibleAnnotations_attribute.attribute_length item.
1787     * @param numAnnotations
1788     *         RuntimeVisibleAnnotations_attribute.num_annotations item.
1789     * @param annotations

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1790         * RuntimeInvisibleAnnotations_attribute.annotations item.
1791     */
1792
1793     public CLRuntimeInvisibleAnnotationsAttribute(int attributeNameIndex,
1794         long attributeLength, int numAnnotations,
1795         ArrayList<CLAnnotation> annotations) {
1796         super(attributeNameIndex, attributeLength);
1797         this.numAnnotations = numAnnotations;
1798         this.annotations = annotations;
1799     }
1800
1801     /**
1802     * @inheritDoc
1803     */
1804
1805     public void write(CLOutputStream out) throws IOException {
1806         super.write(out);
1807         out.writeShort(numAnnotations);
1808         for (int i = 0; i < annotations.size(); i++) {
1809             annotations.get(i).write(out);
1810         }
1811     }
1812
1813     /**
1814     * @inheritDoc
1815     */
1816
1817     public void writeToStdout(PrettyPrinter p) {
1818         p.printf("RuntimeInvisibleAnnotations {\n");
1819         p.indentRight();
1820         super.writeToStdout(p);
1821         p.printf("Number of Annotations: %s\n", numAnnotations);
1822         for (int i = 0; i < annotations.size(); i++) {
1823             annotations.get(i).writeToStdout(p);
1824         }
1825         p.indentLeft();
1826         p.printf("}\n");
1827     }
1828
1829 }
1830
1831 /**
1832  * Representation of parameter_annotations_table entry structure (JVM Spec
1833  * Section 4.8.17).
1834  */
1835
1836 class CLParameterAnnotationInfo {
1837
1838     /** parameter_annotations_table_entry.num_annotations item. */
1839     public int numAnnotations;
1840
1841     /** parameter_annotations_table_entry.annotations item. */
1842     public ArrayList<CLAnnotation> annotations;
1843
1844     /**
1845     * Construct a ParameterAnnotationInfo object.
1846     *
1847     * @param numAnnotations
1848     *         parameter_annotations_table_entry.num_annotations item.
1849     * @param annotations
1850     *         parameter_annotations_table_entry.annotations item.
1851     */
1852
1853     public CLParameterAnnotationInfo(int numAnnotations,
1854         ArrayList<CLAnnotation> annotations) {
1855         this.numAnnotations = numAnnotations;
1856         this.annotations = annotations;
1857     }
1858

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1859 /**
1860  * Write the contents of this object to the specified output stream.
1861  *
1862  * @param out
1863  *         output stream.
1864  * @throws IOException
1865  *         if an error occurs while writing.
1866  */
1867
1868 public void write(CLOutputStream out) throws IOException {
1869     out.writeShort(numAnnotations);
1870     for (int i = 0; i < annotations.size(); i++) {
1871         annotations.get(i).write(out);
1872     }
1873 }
1874
1875 /**
1876  * Write the content of this object to STDOUT in a format similar to that of
1877  * javap.
1878  *
1879  * @param p
1880  *         for pretty printing with indentation.
1881  */
1882
1883 public void writeToStdOut(PrettyPrinter p) {
1884     p.printf("ParameterAnnotationInfo {\n");
1885     p.indentRight();
1886     p.printf("Number of Annotations: %s\n", numAnnotations);
1887     for (int i = 0; i < annotations.size(); i++) {
1888         annotations.get(i).writeToStdOut(p);
1889     }
1890     p.indentLeft();
1891     p.printf("}\n");
1892 }
1893
1894 }
1895
1896 /**
1897  * Representation of RuntimeVisibleParameterAnnotationsAttribute structure (JVM
1898  * Spec Section 4.8.17).
1899  */
1900
1901 class CLRuntimeVisibleParameterAnnotationsAttribute extends CLAttributeInfo {
1902
1903     /**
1904      * RuntimeVisibleParameterAnnotations_attribute.num_parameters item.
1905      */
1906     public short numParameters;
1907
1908     /**
1909      * RuntimeVisibleParameterAnnotations_attribute.parameter_annotations item.
1910      */
1911     public ArrayList<CLParameterAnnotationInfo> parameterAnnotations;
1912
1913     /**
1914      * Construct a CLRuntimeVisibleParameterAnnotationsAttribute object.
1915      *
1916      * @param attributeNameIndex
1917      *         RuntimeVisibleParameterAnnotations_attribute.
1918      *         attribute_name_index item.
1919      * @param attributeLength
1920      *         RuntimeVisibleParameterAnnotations_attribute.attribute_length
1921      *         item.
1922      * @param numParameters
1923      *         RuntimeVisibleParameterAnnotations_attribute.num_parameters
1924      *         item.
1925      * @param parameterAnnotations
1926      *         RuntimeVisibleParameterAnnotations_attribute.
1927      *         parameter_annotations item.

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

1928 */
1929
1930 public CLRuntimeVisibleParameterAnnotationsAttribute(
1931     int attributeNameIndex, long attributeLength, short numParameters,
1932     ArrayList<CLParameterAnnotationInfo> parameterAnnotations) {
1933     super(attributeNameIndex, attributeLength);
1934     this.numParameters = numParameters;
1935     this.parameterAnnotations = parameterAnnotations;
1936 }
1937
1938 /**
1939  * @inheritDoc
1940  */
1941
1942 public void write(CLOutputStream out) throws IOException {
1943     super.write(out);
1944     out.writeByte(numParameters);
1945     for (int i = 0; i < parameterAnnotations.size(); i++) {
1946         parameterAnnotations.get(i).write(out);
1947     }
1948 }
1949
1950 /**
1951  * @inheritDoc
1952  */
1953
1954 public void writeToStdOut(PrettyPrinter p) {
1955     p.printf("RuntimeVisibleParameterAnnotations {\n");
1956     p.indentRight();
1957     super.writeToStdOut(p);
1958     p.printf("Number of Parameters: %s\n", numParameters);
1959     for (int i = 0; i < parameterAnnotations.size(); i++) {
1960         parameterAnnotations.get(i).writeToStdOut(p);
1961     }
1962     p.indentLeft();
1963     p.printf("}\n");
1964 }
1965
1966 }
1967
1968 /**
1969  * Representation of RuntimeInvisibleParameterAnnotations_attribute structure
1970  * (JVM Spec Section 4.8.18).
1971  */
1972
1973 class CLRuntimeInvisibleParameterAnnotationsAttribute extends CLAttributeInfo {
1974
1975     /**
1976      * RuntimeInvisibleParameterAnnotations_attribute.num_parameters item.
1977      */
1978     public short numParameters;
1979
1980     /**
1981      * RuntimeInvisibleParameterAnnotations_attribute. parameter_annotations
1982      * item.
1983      */
1984     public ArrayList<CLParameterAnnotationInfo> parameterAnnotations;
1985
1986     /**
1987      * Construct a CLRuntimeInvisibleParameterAnnotationsAttribute object.
1988      *
1989      * @param attributeNameIndex
1990      *         RuntimeInvisibleParameterAnnotations_attribute.
1991      *         attribute_name_index item.
1992      * @param attributeLength
1993      *         RuntimeInvisibleParameterAnnotations_attribute.
1994      *         attribute_length item.
1995      * @param numParameters
1996      *         RuntimeInvisibleParameterAnnotations_attribute.num_parameters

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder


```

1997     *           item.
1998     * @param parameterAnnotations
1999     *           RuntimeInvisibleParameterAnnotations_attribute.
2000     *           parameter_annotations item.
2001     */
2002
2003     public CLRuntimeInvisibleParameterAnnotationsAttribute(
2004         int attributeNameIndex, long attributeLength, short numParameters,
2005         ArrayList<CLParameterAnnotationInfo> parameterAnnotations) {
2006         super(attributeNameIndex, attributeLength);
2007         this.numParameters = numParameters;
2008         this.parameterAnnotations = parameterAnnotations;
2009     }
2010
2011     /**
2012     * @inheritDoc
2013     */
2014
2015     public void write(CLOutputStream out) throws IOException {
2016         super.write(out);
2017         out.writeByte(numParameters);
2018         for (int i = 0; i < parameterAnnotations.size(); i++) {
2019             parameterAnnotations.get(i).write(out);
2020         }
2021     }
2022
2023     /**
2024     * @inheritDoc
2025     */
2026
2027     public void writeToStdout(PrettyPrinter p) {
2028         p.printf("RuntimeInvisibleParameterAnnotations {\n");
2029         p.indentRight();
2030         super.writeToStdout(p);
2031         p.printf("Number of Parameters: %d\n", numParameters);
2032         for (int i = 0; i < parameterAnnotations.size(); i++) {
2033             parameterAnnotations.get(i).writeToStdout(p);
2034         }
2035         p.indentLeft();
2036         p.printf("}\n");
2037     }
2038
2039 }
2040
2041 /**
2042  * Representation of AnnotationDefault_attribute structure (JVM Spec Section
2043  * 4.8.2).
2044  */
2045
2046 class CLAnnotationDefaultAttribute extends CLAttributeInfo {
2047
2048     /** AnnotationDefault_attribute.defaultValue item. */
2049     public CLElementValue defaultValue;
2050
2051     /**
2052     * Construct a CLAnnotationDefaultAttribute object.
2053     *
2054     * @param attributeNameIndex
2055     *           AnnotationDefault_attribute.attribute_name_index item.
2056     * @param attributeLength
2057     *           AnnotationDefault_attribute.attribute_length item.
2058     * @param defaultValue
2059     *           AnnotationDefault_attribute.defaultValue item.
2060     */
2061
2062     public CLAnnotationDefaultAttribute(int attributeNameIndex,
2063         long attributeLength, CLElementValue defaultValue) {
2064         super(attributeNameIndex, attributeLength);
2065         this.defaultValue = defaultValue;

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
2066     }
2067
2068     /**
2069     * @inheritDoc
2070     */
2071
2072     public void write(CLOutputStream out) throws IOException {
2073         super.write(out);
2074         defaultValue.write(out);
2075     }
2076
2077     /**
2078     * @inheritDoc
2079     */
2080
2081     public void writeToStdOut(PrettyPrinter p) {
2082         p.printf("AnnotationDefault {\n");
2083         p.indentRight();
2084         super.writeToStdOut(p);
2085         defaultValue.writeToStdOut(p);
2086         p.indentLeft();
2087         p.printf("}\n");
2088     }
2089
2090 }
2091
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder