CLFile.java

```java
// Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas

package jminusminus;

import java.io.IOException;
import java.util.ArrayList;
import static jminusminus.CLConstants.*;

/**
 * Representation of the ClassFile structure (JVM Spec Section 4.2). An instance
 * of CLFile is created when a class is read using CLAbsorber or constructed
 * using CLEmitter.
 *
 * We have our own representation and don't use java.lang.Class because Java
 * does not offer an interface to programmatically create a class file in memory
 * other than creating it in one shot from a byte stream.
 */

class CLFile {

    // The fields below represent the members of the ClassFile
    // structure. See JVM Spec Section 4.2 for details.

    /** ClassFile.magic item. */
    public long magic; // 0xCAFEBABE

    /** ClassFile.minor_version item. */
    public int minorVersion;

    /** ClassFile.major_version item. */
    public int majorVersion;

    /** ClassFile.constant_pool_count item. */
    public int constantPoolCount;

    /** ClassFile.constant_pool item. */
    public CLConstantPool constantPool;

    /** ClassFile.access_flags item. */
    public int accessFlags;

    /** ClassFile.this_class item. */
    public int thisClass;

    /** ClassFile.super_class item. */
    public int superClass;

    /** ClassFile.interfaces_count item. */
    public int interfacesCount;

    /** ClassFile.interfaces item. */
    public ArrayList<Integer> interfaces;

    /** ClassFile.fields_count item. */
    public int fieldsCount;

    /** ClassFile.fields item. */
    public ArrayList<CLFieldInfo> fields;

    /** ClassFile.methods_count item. */
    public int methodsCount;

    /** ClassFile.methods item. */
    public ArrayList<CLMethodInfo> methods;

    /** ClassFile.attributes_count item. */
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```java
    public int attributesCount;

    /** ClassFile.attributes item. */
    public ArrayList<CLAttributeInfo> attributes;

    /**
     * Write the contents of this class to the specified output stream.
     *
     * @param out
     *            output stream.
     * @throws IOException
     *             if an error occurs while writing.
     */

    public void write(CLOutputStream out) throws IOException {
        out.writeInt(magic);
        out.writeShort(minorVersion);
        out.writeShort(majorVersion);
        out.writeShort(constantPoolCount);
        constantPool.write(out);
        out.writeShort(accessFlags);
        out.writeShort(thisClass);
        out.writeShort(superClass);
        out.writeShort(interfacesCount);
        for (int i = 0; i < interfaces.size(); i++) {
            Integer index = interfaces.get(i);
            out.writeShort(index.intValue());
        }
        out.writeShort(fieldsCount);
        for (int i = 0; i < fields.size(); i++) {
            CLMemberInfo fieldInfo = fields.get(i);
            if (fieldInfo != null) {
                fieldInfo.write(out);
            }
        }
        out.writeShort(methodsCount);
        for (int i = 0; i < methods.size(); i++) {
            CLMemberInfo methodInfo = methods.get(i);
            if (methodInfo != null) {
                methodInfo.write(out);
            }
        }
        out.writeShort(attributesCount);
        for (int i = 0; i < attributes.size(); i++) {
            CLAttributeInfo attributeInfo = attributes.get(i);
            if (attributeInfo != null) {
                attributeInfo.write(out);
            }
        }
    }

    /**
     * Write the contents of the class file to STDOUT in a format similar to
     * that of javap.
     */

    public void writeToStdOut() {
        PrettyPrinter p = new PrettyPrinter();
        p.printf("Magic Number: %x\n", magic);
        p.printf("Minor Version: %d\n", minorVersion);
        p.printf("Major Version: %d\n", majorVersion);
        p.printf("Access Flags: %s\n", classAccessFlagsToString(accessFlags));
        p.println();
        constantPool.writeToStdOut(p);
        p.println();
        p.printf("This Class Index: %d\n", thisClass);
        p.printf("Super Class Index: %d\n", superClass);
        p.println();
        p.printf("// Fields (%d Items)\n", fieldsCount);
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```java
        for (int i = 0; i < fields.size(); i++) {
            CLMemberInfo fieldInfo = fields.get(i);
            if (fieldInfo != null) {
                fieldInfo.writeToStdOut(p);
            }
        }
        p.println();
        p.printf("// Methods (%d Items)\n", methodsCount);
        for (int i = 0; i < methods.size(); i++) {
            CLMemberInfo methodInfo = methods.get(i);
            if (methodInfo != null) {
                methodInfo.writeToStdOut(p);
            }
        }
        p.println();
        p.printf("// Attributes (%d Items)\n", attributesCount);
        for (int i = 0; i < attributes.size(); i++) {
            CLAttributeInfo attributeInfo = attributes.get(i);
            attributeInfo.writeToStdOut(p);
        }
    }

    /**
     * Return (as a string) the class access permissions and properties
     * contained in the specified mask of flags.
     * 
     * @param accessFlags
     *            mask of access flags.
     * @return a string identifying class access permissions and properties.
     */

    public static String classAccessFlagsToString(int accessFlags) {
        StringBuffer b = new StringBuffer();
        if ((accessFlags & ACC_PUBLIC) != 0) {
            b.append("public ");
        }
        if ((accessFlags & ACC_FINAL) != 0) {
            b.append("final ");
        }
        if ((accessFlags & ACC_SUPER) != 0) {
            b.append("super ");
        }
        if ((accessFlags & ACC_INTERFACE) != 0) {
            b.append("interface ");
        }
        if ((accessFlags & ACC_ABSTRACT) != 0) {
            b.append("abstract ");
        }
        if ((accessFlags & ACC_SYNTHETIC) != 0) {
            b.append("synthetic ");
        }
        if ((accessFlags & ACC_ANNOTATION) != 0) {
            b.append("annotation ");
        }
        if ((accessFlags & ACC_ENUM) != 0) {
            b.append("enum ");
        }
        return b.toString().trim();
    }

    /**
     * Return (as a string) the inner class access permissions and properties
     * contained in the specified mask of flags.
     * 
     * @param accessFlags
     *            mask of access flags.
     * @return a string identifying the inner class access permissions and
     *         properties.
     */
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```java

    public static String innerClassAccessFlagsToString(int accessFlags) {
        StringBuffer b = new StringBuffer();
        if ((accessFlags & ACC_PUBLIC) != 0) {
            b.append("public ");
        }
        if ((accessFlags & ACC_PRIVATE) != 0) {
            b.append("private ");
        }
        if ((accessFlags & ACC_PROTECTED) != 0) {
            b.append("protected ");
        }
        if ((accessFlags & ACC_STATIC) != 0) {
            b.append("static ");
        }
        if ((accessFlags & ACC_FINAL) != 0) {
            b.append("final ");
        }
        if ((accessFlags & ACC_INTERFACE) != 0) {
            b.append("interface ");
        }
        if ((accessFlags & ACC_ABSTRACT) != 0) {
            b.append("abstract ");
        }
        if ((accessFlags & ACC_SYNTHETIC) != 0) {
            b.append("synthetic ");
        }
        if ((accessFlags & ACC_ANNOTATION) != 0) {
            b.append("annotation ");
        }
        if ((accessFlags & ACC_ENUM) != 0) {
            b.append("enum ");
        }
        return b.toString().trim();
    }

    /**
     * Return (as a string) the field access permissions and properties
     * contained in the specified mask of flags.
     *
     * @param accessFlags
     *            mask of access flags.
     * @return a string identifying the field access permissions and properties.
     */

    public static String fieldAccessFlagsToString(int accessFlags) {
        StringBuffer b = new StringBuffer();
        if ((accessFlags & ACC_PUBLIC) != 0) {
            b.append("public ");
        }
        if ((accessFlags & ACC_PRIVATE) != 0) {
            b.append("private ");
        }
        if ((accessFlags & ACC_PROTECTED) != 0) {
            b.append("protected ");
        }
        if ((accessFlags & ACC_STATIC) != 0) {
            b.append("static ");
        }
        if ((accessFlags & ACC_FINAL) != 0) {
            b.append("final ");
        }
        if ((accessFlags & ACC_VOLATILE) != 0) {
            b.append("volatile ");
        }
        if ((accessFlags & ACC_TRANSIENT) != 0) {
            b.append("transient ");
        }
        if ((accessFlags & ACC_NATIVE) != 0) {
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
            b.append("native ");
        }
        if ((accessFlags & ACC_SYNTHETIC) != 0) {
            b.append("synthetic ");
        }
        if ((accessFlags & ACC_ENUM) != 0) {
            b.append("enum ");
        }
        return b.toString().trim();
    }

    /**
     * Return (as a string) the method access permissions and properties
     * contained in the specified mask of flags.
     *
     * @param accessFlags
     *            mask of access flags.
     * @return a string identifying the method access permissions and
     *         properties.
     */

    public static String methodAccessFlagsToString(int accessFlags) {
        StringBuffer b = new StringBuffer();
        if ((accessFlags & ACC_PUBLIC) != 0) {
            b.append("public ");
        }
        if ((accessFlags & ACC_PRIVATE) != 0) {
            b.append("private ");
        }
        if ((accessFlags & ACC_PROTECTED) != 0) {
            b.append("protected ");
        }
        if ((accessFlags & ACC_STATIC) != 0) {
            b.append("static ");
        }
        if ((accessFlags & ACC_FINAL) != 0) {
            b.append("final ");
        }
        if ((accessFlags & ACC_SYNCHRONIZED) != 0) {
            b.append("synchronized ");
        }
        if ((accessFlags & ACC_BRIDGE) != 0) {
            b.append("bridge ");
        }
        if ((accessFlags & ACC_VARARGS) != 0) {
            b.append("varargs ");
        }
        if ((accessFlags & ACC_NATIVE) != 0) {
            b.append("native ");
        }
        if ((accessFlags & ACC_ABSTRACT) != 0) {
            b.append("abstract ");
        }
        if ((accessFlags & ACC_STRICT) != 0) {
            b.append("strict ");
        }
        if ((accessFlags & ACC_SYNTHETIC) != 0) {
            b.append("synthetic ");
        }
        return b.toString().trim();
    }

    /**
     * Return the integer value (mask) corresponding to the specified access
     * flag.
     *
     * @param accessFlag
     *            access flag.
     * @return the integer mask.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```java
     */

    public static int accessFlagToInt(String accessFlag) {
        int flag = 0;
        if (accessFlag.equals("public")) {
            flag = ACC_PUBLIC;
        }
        if (accessFlag.equals("private")) {
            flag = ACC_PRIVATE;
        }
        if (accessFlag.equals("protected")) {
            flag = ACC_PROTECTED;
        }
        if (accessFlag.equals("static")) {
            flag = ACC_STATIC;
        }
        if (accessFlag.equals("final")) {
            flag = ACC_FINAL;
        }
        if (accessFlag.equals("super")) {
            flag = ACC_SUPER;
        }
        if (accessFlag.equals("synchronized")) {
            flag = ACC_SYNCHRONIZED;
        }
        if (accessFlag.equals("volatile")) {
            flag = ACC_VOLATILE;
        }
        if (accessFlag.equals("bridge")) {
            flag = ACC_BRIDGE;
        }
        if (accessFlag.equals("transient")) {
            flag = ACC_TRANSIENT;
        }
        if (accessFlag.equals("varargs")) {
            flag = ACC_VARARGS;
        }
        if (accessFlag.equals("native")) {
            flag = ACC_NATIVE;
        }
        if (accessFlag.equals("interface")) {
            flag = ACC_INTERFACE;
        }
        if (accessFlag.equals("abstract")) {
            flag = ACC_ABSTRACT;
        }
        if (accessFlag.equals("strict")) {
            flag = ACC_STRICT;
        }
        if (accessFlag.equals("synthetic")) {
            flag = ACC_SYNTHETIC;
        }
        if (accessFlag.equals("annotation")) {
            flag = ACC_ANNOTATION;
        }
        if (accessFlag.equals("enum")) {
            flag = ACC_ENUM;
        }
        return flag;
    }

}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder