# Compiling and Running

# Functionality

The program provides the capability to read a java class file and construct a call tree for a chosen method.

The class file is passed as an argument to the program and you will be prompted to choose a method from a list of methods that are available in that class. Once a method has been chosen, the call tree for the method will be shown as a command line output.

If the method being called belongs to another class, then the program will try to parse the corresponding class file to obtain the subsequent method calls.

If the program detects recursive calls then it will terminate the call tree branch and indicate recursion with a [RECURSIVE] tag next to the call tree branch.

# Design

## High Level Design

SmAssignment2.java contains the main method for the application. The main method will get the class file directory from the command line arguments and create a ClassFile object passing in the class file directory.

The implementation in ClassFile.java will take care of creating a DataInputStream and reading the byte code into a ClassFile object. Once the object has been constructed, the method information of the class files is taken using the getMethods() method that is available in the class file. This will return an array of method\_info. Subsequently the user will be prompted to choose a method from the list of methods shown.

Once the user selects the method, the method\_info object will be passed to the DrawMethodTree() method which will recursively draw the call tree starting with this method.

## Classes

The following files were provided to be used with the program.

* ClassFile.java - Modified
* ClassFileParser.java - Unused
* ClassFileParserException.java
* ConstantPool.java - Unmodified
* CPEntry.java – Modified
* Instruction.java – Unmodified
* Opcode.java - Unmodified

Extra classes added

* FieldInfo.java
* MethodInfo.java
* AttributeInfo.java
* CodeAttributeInfo.java
* SM Assignment2.java

### Class – ClassFile

Initial ClassFile provided had the implementation to take a class file as a parameter and creates a ClassFile object from it. But the implementation was not complete. The given file could only parse till the constant pool. To construct the call tree method information is needed. The rest of the implementation to parse the ClassFile was added later.

### Class - MethodInfo

MethodInfo class maps directly to the method\_info in the class file. The constructor accepts a DataInputStream which can be used to read the bytes and map it to a Methodnfo object.

### Class – AttributeInfo

AttributeInfo class can be used to map an attribute\_info type in the class file. Implementation of this class is like MethodInfo class. AttributeInfo has a class name parse that reads the attribute type using attribute\_name\_index. If it is a code attribute, then a CodeAttributeInfo object is created and returned. Else an AttributeInfo object is returned.

## Class – CodeAttributeInfo

CodeAttributeInfo is a type of AttributeInfo. Implementation is like AttributeInfo. Instead of the info array in AttributeInfo this class contains additional attributes. The most important of which is the code array. This array is a byte array and contains the JVM instructions which is needed to find out the method calls.

The method GetInstructionList will parse the code array and return a list of Instruction objects.

# Testing

# Quality

# Referencing