Original Solution:

The solution is to add collection code (such as gson.tojson and write into a file) to each java class, not only the constructor in each class but also each method in every class if we would like to collect every java object whenever it is created or modified by any method. This work is kind of unachievable if I am gonna to do it manually, and I do not have a good way to solve it right now. The second solution is to modify the actual main function of defects4j and add code to it. Since there are no java programs in defects4j that contain a main function, defects4j is using another way to make the program run (like the main). Based on my research, the defects4j is using ant to compile /build/run, this could be the main function (here https://github.com/rjust/defects4j/blob/572b84df33138b924bebe98f6f6dc058a5257f60/framework/bin/defects4j). Then the main calls d4j -test, and d4j-test calls compile and test (here https://github.com/rjust/defects4j/blob/master/framework/bin/d4j/d4j-test#L127). It is also related to perl and junit, so I need to add the jar file of the junit to work with the main function.

Adding collection code to defects4j:

Adding my own code to defects4j:
First I tried to add some code to defects4j to see if we could create a new file and write some information to it. I checked a buggy source code version (Closure, bug 1, buggy version), then I add the following code to AliasString.java in /tmp/closure_1_buggy:

```
try {
      File myObj = new File("filename.txt");
      myObj.createNewFile();
      FileWriter myWriter = new FileWriter("filename.txt");
      myWriter.write("Lets go go go go!");
      myWriter.close();
      System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
```

```
    e.printStackTrace();
  }
```

After running “defects4j compile” and “defects4j test”, the console prints out “Successfully wrote to the file” and I got a new file in /tmp/closure_1_buggy, which is called “filename” with the message I wrote in it.

Update:
After switching to my new pc, it seems not working anymore, still trying to figure out why.

Including Gson to defects4j:

In order to import gson to defects4j database, we need to modify the pom.xml file in the folder, such as /tmp/closure_1_buggy, but there does not exist a pom.xml file in the bug version of closure project (there is one for lang project), and it is still not work after I created a new pom.xml in the closure file.

Yigit’s solution to include Gson library as a dependency to defects4j:
1) Downloade the jar file from
https://search.maven.org/artifact/com.google.code.gson/gson/2.8.6/jar

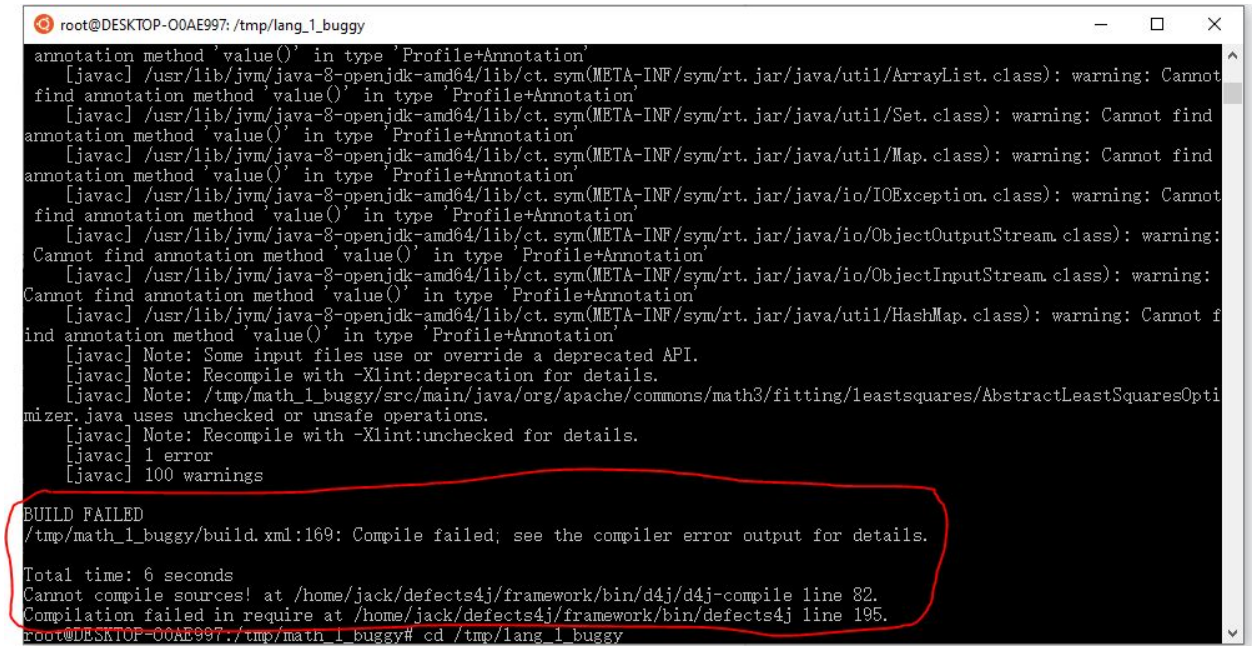
2) Move the Jar file to the root directory of the project you wanted to extract information from. For an example, /defects4j/framework/projects/Math/lib

3) Add the following lines to the build.xml on the individual project:
(this started at line 112 for me in Math1)

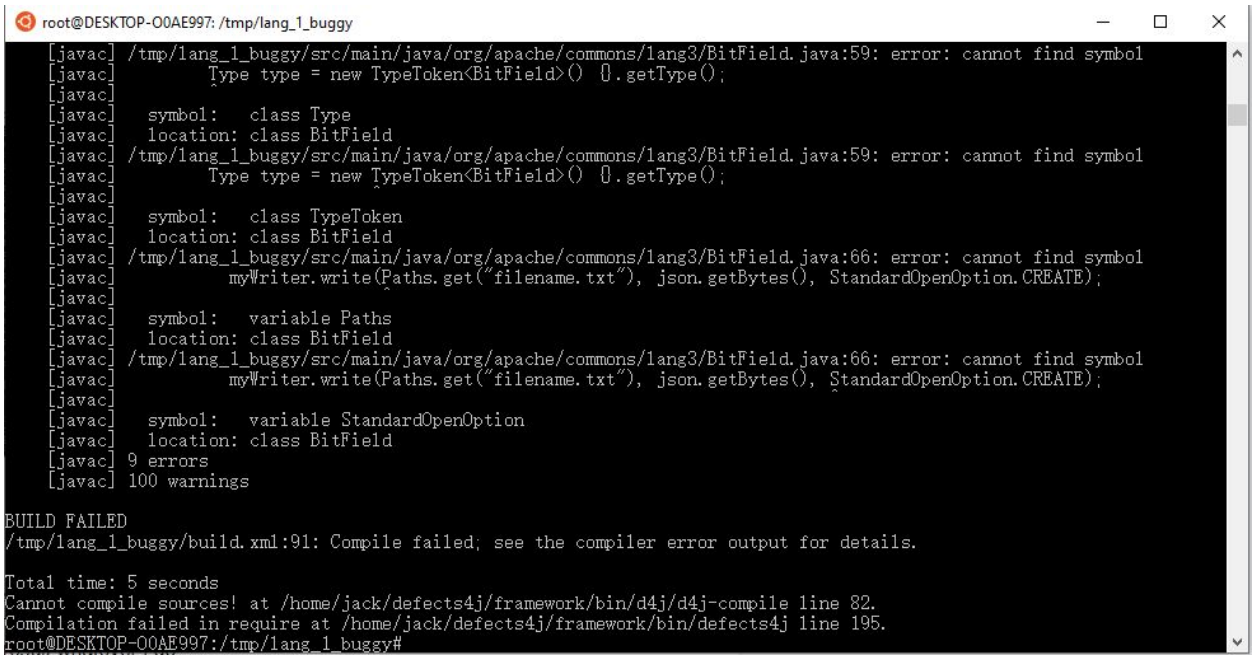
```
 <!-- External dependency classpath -->
  <path id="downloaded.lib.classpath">
    <pathelement location="${download.lib.dir}/junit-${junit.version}.jar"/>
    <pathelement
location="${d4j.home}/framework/projects/Math/lib/gson-2.8.6.jar" />
  </path>
```

4)Add the import statement to a test randomly chosen to output an object, "import com.google.gson.*"

Based on Yigit’s solution, I followed each step but I got a build failed:



I tried it with project Lang as well, still the same output:



I added the following code to the first bug version of project Lang in BitField.java, in order to use gson library to extract object information, we need to use the constructor to construct java objects first.
Gson gson = new GsonBuilder().setPrettyPrinting().create();
Type type = new TypeToken<BitField>() {}.getType();

```
BitField bit = new BitField(_mask);
String json = gson.toJson(bit, type);
try {
        File myObj = new File("filename.txt");
        myObj.createNewFile();
        FileWriter myWriter = new FileWriter("filename.txt");
        myWriter.write(Paths.get("filename.txt"), json.getBytes(),
        StandardOpenOption.CREATE);
        myWriter.close();
        System.out.println("Successfully wrote to the file.");
} catch (IOException e) {
        System.out.println("An error occurred.");
        e.printStackTrace();
}
```

One annoying problem so far:

The defects4j bug version can only be compiled and tested once, after the first try I will keep getting the same result. For example, if we get a compile error for the first run, we will still get a compile error even if we already fixed the problem. Similarly, if we successfully compiled the bug version first time by using “defects4j compile”, we will keep getting the same results even changing the code or deleting something from the folder. Same situation for “defects4j test. In addition, restarting the ubuntu would not solve this problem, the only way is to delete the folder and then regenerate another one using “defects4j checkout -p Closure -v 1b -w /tmp/closure_1_buggy”.

Currently Working on:

I am currently working on how to use java reflection in defects4j.

Since Gson library itself is using java reflection to function, we could also use java reflection for this project. Compared to Gson library, java reflection may not need to construct the object first then extract from it.

I tried to import “java.lang.reflect.Constructor;” directly into defects4j, and we can successfully compile and test, which means we do not need to add any jar file to use java reflection.