Starting off the PAV'18 Class Project

Stanly Samuel and Dr. Rekha Pai

1 Bitbucket repository and Git

1.1 Bitbucket repository

We will be using Bitbucket to monitor your code. To set up using Bitbucket, do the following:

1. Create a Bitbucket account (Bitbucket.org).
2. The repository containing the basic project template is at https://stanlyjs@bitbucket.org/stanlyjs/e0227-null-dereference-analysis. **Do not clone this repository!**
3. Two members in each team is recommended. One member must fork (as shown in the project tutorial) the repo and give admin access to the second member.
4. While forking, please rename the forked repo to "E0227: Null dereference analysis - *teammember1, teammember2* ". For example, "E0227: Null dereference analysis - Stanly, Rekha". We will assume the number of teams (and members in each team) from the repositories received.
5. **Please ensure that the forked repo is private, failing which you will be penalized.**
6. Share the forked repo once again and give **admin** access to users **komondoor**, **stanlyjs** and **rekhapai**. We will monitor your commits through this access.
7. Now you can clone your forked repo into your local machine and keep pushing your local commits to this remote forked repository so that we can monitor your commits.

1.2 Git

Basic commands needed:

1. git status
2. git add .
3. git push origin master (origin is your forked repo and branch is master.)
4. git pull origin master
5. git commit -m "MESSAGE" (Please commit using sensible messages. Good examples "Added XYZ feature", "Updated XYZ file" etc. Bad examples: "Done", "Ok", "Updated", "Added commit", "Donald Trump")

Study git from here: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

1.3 Important points

1. The repository containing the basic project template is at https://stanlyjs@bitbucket.org/stanlyjs/e0227-null-dereference-analysis as mentioned before. We will be updating this repository with more test cases and other updates needed for the project. You need to be up to date with this repository. Whenever we make some changes to the repo, we will ask you to pull the changes. To do that add this repo as a remote in your local repository.

   ```
   git remote add upstream https://stanlyjs@bitbucket.org/stanlyjs/e0227-null-dereference-analysis.
   ```

   To update your local repo, run the following command:

   ```
   git pull upstream master
   ```

2. If you have never installed git, there will an authentication step in your local repo. Just follow the instructions as they come.

3. This repository has been tested on Windows 7 and Ubuntu 16.04 with Eclipse 2018-09 and Java SE 1.8. This repo should work on other platforms as well. However, it is **mandatory to use Java SE 1.8** only failing which the project will not build.

4. Please use Eclipse as the IDE; Wala is designed to work optimally with Eclipse.

2 Setting up the project locally

2.1 Eclipse

By now, you should have the local repository cloned on your machine as "e0227-null-dereference-analysis-stanly-rekha". The following steps correspond to setting up WALA in an Eclipse environment. Make sure that you have Java SDK and JRE 1.8 as well as Eclipse installed in your computer before you perform these steps.

1. Open Eclipse

2. Switch workspace to "e0227-null-dereference-analysis-stanly-rekha"

3. Import all the projects into eclipse by using **File → Import → General → Existing Projects into Workspace** and provide the path to "e0227-null-dereference-analysis-stanly-rekha"

4. Open **com.ibm.wala.core → dat → wala.properties.sample** and set the **java_runtime_dir** to point to your java runtime libraries (eg: /usr/lib/jvm/java-8-oracle/jre/lib).

5. Also update the **Default output directory** variable **output_dir** to where you want the Wala Output to be stored. This is a mandatory setting although we may never use it. Just update it to avoid any errors during runtime.

2.2 Running

You will observe that the sample testcase .jar file is present in the project folder as PAV18ProjectTests.jar. To run the analysis using this sample testcase and ensure that everything is set up correctly, follow these steps:

1. Set the arguments for PAV18Project:
   right click **PAV18Project → Run As → Run Configurations → Java Application.**

2. Under Main tab, select Project as **PAV18Project** and Main class as our analysis class **PAVNullDeref-Package.PAVNullDerefAnalysis**.

3. Under Arguments tab, provide the arguments in the following format:
   <path to jar> L<package name of main class>/<class name of main class> L<package name of class cointaining the method to be analysed>/<class name of that class> <method signature>
   E.g.:
   /home/stanly/Projects/e0227-null-dereference-analysis-stanly/PAV18ProjectTests.jar LTestCases/SampleTests LTestCases/SampleTests main

4. Run the project and you should be able to see the Call Graph for the project and the CFG and IR of the given method which looks something like this:

```
  Displaying Application's Call Graph nodes:
Node: < Application, LTestCases/SampleTests, main([Ljava/lang/String;)V > Context: Everywhere
Node: < Application, LTestCases/SampleTests, foo(I)V > Context: Everywhere
Node: < Application, LTestCases/SampleTests, bar(LTestCases/SampleTests;)V > Context: Everywhere
Node: < Application, LTestCases/SampleTests, <init>()V > Context: Everywhere
Node: synthetic < Primordial, Ljava/lang/Object, getClass()Ljava/lang/Class; > Context: JavaType



The IR of method TestCases.SampleTests.main([Ljava/lang/String;)V is:
< Application, LTestCases/SampleTests, main([Ljava/lang/String;)V >
CFG:
BB0[-1..-2]
```

```
    -> BB1
BB1[0..1]
    -> BB2
    -> BB4
BB2[2..3]
    -> BB3
    -> BB4
BB3[4..4]
    -> BB4
BB4[-1..-2]
Instructions:
BB0
BB1
1   invokestatic < Application, LTestCases/SampleTests, foo(I)V > v3:#10 @2 exception:v4(line 34
BB2
3   invokestatic < Application, LTestCases/SampleTests, bar(LTestCases/SampleTests;)V > v5:#null
BB3
4   return                                  (line 36)
BB4
```

Note: The projects **com.ibm.wala.core.tests**, **com.ibm.wala.ide** and **com.ibm.wala.ide.tests** might have errors in them. Ignore those as they will not affect the execution of PAV18Project.

2.3 Creating own test cases

1. Create a testcase source in a folder/package, compile it and compress the folder into a .jar. To run, ensure that the arguments are set appropriately as explained in section 2.2.
2. For example, you can export the PAV18ProjectTests project in Eclipse:
   right click **PAV18ProjectTests → Export → Java → JAR → click Next → provide destination → click Finish.**

2.4 Generating CFG dot file for a test case

1. Ensure that dot (in package graphviz) installed in your system.
2. Set the path for pdfviewer and dot in wala.examples.properties. Check location by typing 'which dot' in terminal. E.g. usr/bin/evince and usr/bin/dot.
3. Run PAV18ProjectTests folder using following configuration:

```
Run configuration:

  Project: com.ibm.wala.core.tests
  Main class: com.ibm.wala.examples.drivers.PDFWalaIR

  Parameters:

  -appJar ${workspace_loc}/PAV18ProjectTests.jar -sig TestCases.SampleTests.foo(I)V;
```

2.5 Possible errors while running

The following error is highly likely to be the **only** cause of error while setting up this project initially on your machine:

2.5.1 Error

"Unable to build scope" or "Unable to build Class Hierarchy"

2.5.2 Cause

Both occur due to the absence of Java SE 1.8 and/or the absence of an updated path to JRE 1.8.

2.5.3 Solution

1. Refer to Section 2.1 point 4.
2. Under PAV18Project folder in Eclipse, right click on JRE System Library and select Java SE 1.8 in Execution Environment.