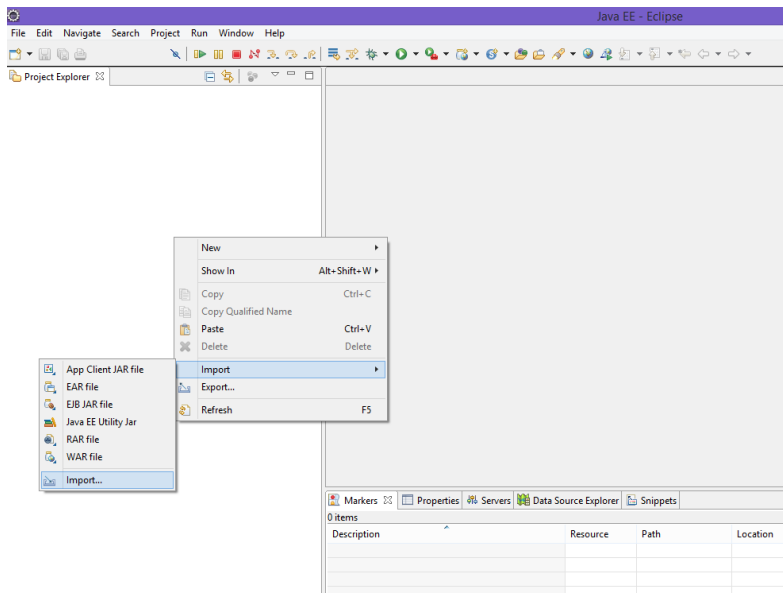


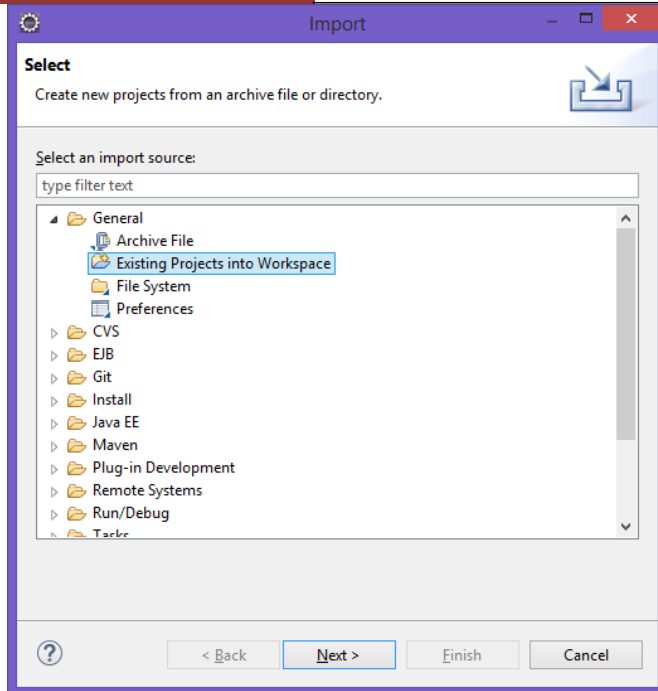
User manual to run SPLAnE through source code.

Installation:

1. Software required to run SPLAnE:
 - a. Java 1.6
 - b. Eclipse IDE
2. Download SPLAnE code from here:
http://www.cse.iitb.ac.in/~splane/splane/src/SPLAnE_Src.zip
3. Unzip the folder anywhere on the local drive.
4. Open Eclipse IDE.
 - a. In Package Explorer click import submenu, as shown below:



- b. Select existing projects into workspace category.



- c. Select the unzip folder where SPLAnE_Src code is unzipped.
- d. Select all projects and click Finish. This will import all the SPLAnE related projects in your workspace.

Running SPLAnE using Source Code:

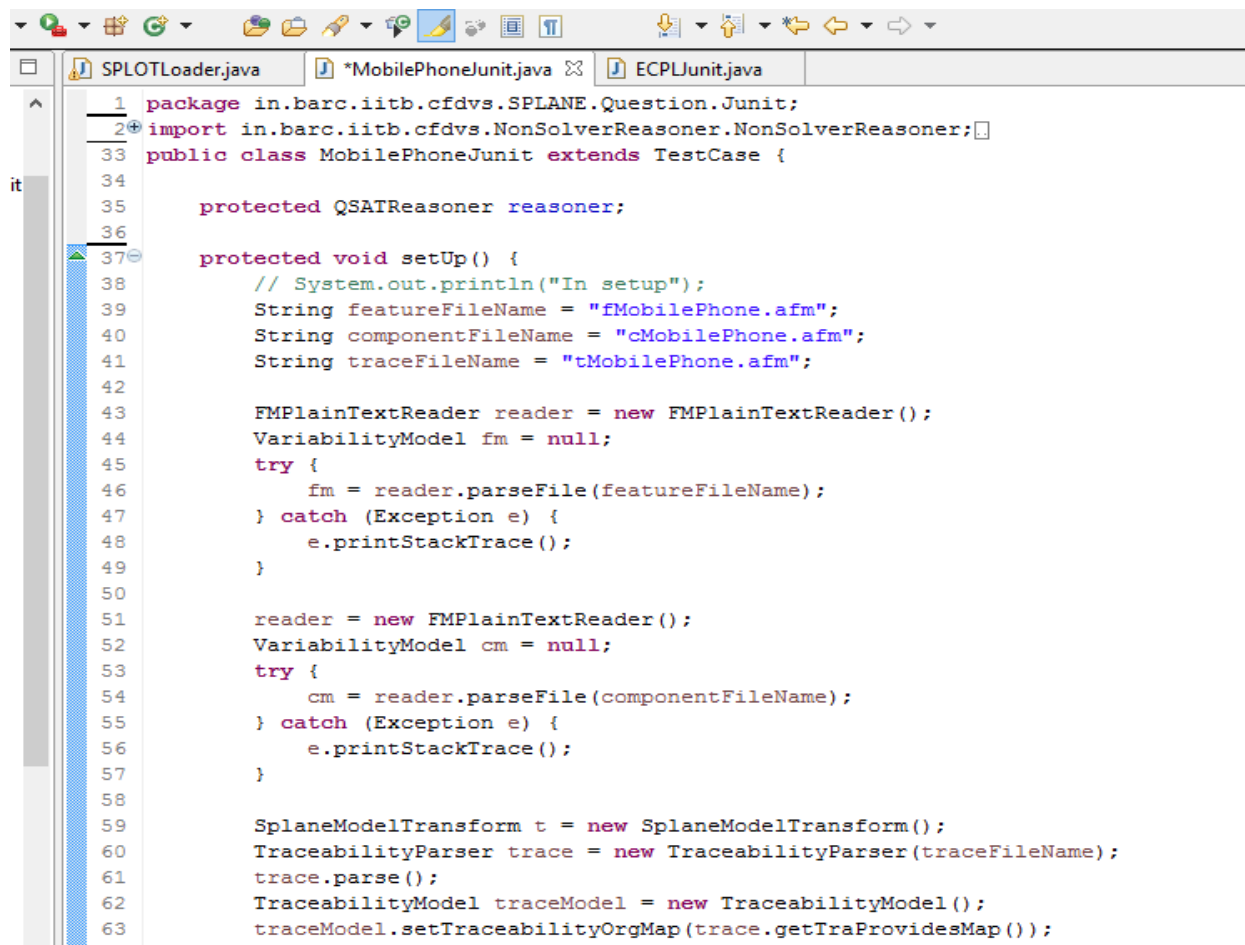
1. Navigate to **QSATReasoner** Project in Package Explorer. Open package **in.barc.iitb.cfdvs.SPLANE.Question.Junit**, in this package create the Junit class.
2. Copy the Feature Model, Component Model and traceability model files in **QSATReasoner** folder. In SetUp method of junit, set the model file name to respective variables as shown below:

```
String featureFileName = "fMobilePhone.afm";  
String componentFileName = "cMobilePhone.afm";  
String traceFileName = "tMobilePhone.afm";
```

3. According to file format load the Parser, example for plain text format do the following:

```
FMPlainTextReader reader = new FMPlainTextReader();
VariabilityModel fm = null;
try {
    fm = reader.parseFile(featureFileName);
} catch (Exception e) {
    e.printStackTrace();
}
```

This will load the feature model. Similarly, do this for Component model and traceability model. The complete code is given below and the sample Junit for MobilePhoneProductLine can be found in existing code.



```
1 package in.barc.iitb.cfdvs.SPLANE.Question.Junit;
2+ import in.barc.iitb.cfdvs.NonSolverReasoner.NonSolverReasoner;
33 public class MobilePhoneJUnit extends TestCase {
34
35     protected QSATReasoner reasoner;
36
37     protected void setUp() {
38         // System.out.println("In setup");
39         String featureFileName = "fMobilePhone.afm";
40         String componentFileName = "cMobilePhone.afm";
41         String traceFileName = "tMobilePhone.afm";
42
43         FMPlainTextReader reader = new FMPlainTextReader();
44         VariabilityModel fm = null;
45         try {
46             fm = reader.parseFile(featureFileName);
47         } catch (Exception e) {
48             e.printStackTrace();
49         }
50
51         reader = new FMPlainTextReader();
52         VariabilityModel cm = null;
53         try {
54             cm = reader.parseFile(componentFileName);
55         } catch (Exception e) {
56             e.printStackTrace();
57         }
58
59         SplaneModelTransform t = new SplaneModelTransform();
60         TraceabilityParser trace = new TraceabilityParser(traceFileName);
61         trace.parse();
62         TraceabilityModel traceModel = new TraceabilityModel();
63         traceModel.setTraceabilityOrgMap(trace.getTraProvidesMap());
64     }
65 }
```

4. Creating the test case:

```
// Test for checking the feature model is valid or invalid.
public void testValidFeatureModelQuestion() {
    ValidFeatureModelQuestion q = new ValidFeatureModelQuestion();
    long startTime = System.currentTimeMillis();
    QSATResult qsatResult = (QSATResult) reasoner.ask(q);
    long stopTime = System.currentTimeMillis();
    System.out.println("Execution Time:"+(stopTime - startTime));
    System.out.println("Number of Features:"+reasoner.getFeaturesMap().size());
    System.out.println("Number of Components:"+reasoner.getComponentsMap().size());
    System.out.println(qsatResult.getQsatDump());
}
```

For every question we can write the testcase and execute it. Sample testcase for each question is available in existing junit.

```
// Tested and working
public void testImplementationQuestionValid() {
    List<String> implementation = getMinimumComponents();
    implementation.add("SMSApps");
    implementation.add("SymbianOSSetup");
    implementation.add("MIDP20");
    String featureName = "SMS";
    ImplementsQuestion q = new ImplementsQuestion(implementation,featureName);
    long startTime = System.currentTimeMillis();
    QSATResult qsatResult = (QSATResult) reasoner.ask(q);
    long stopTime = System.currentTimeMillis();
    System.out.println("Execution Time:"+(stopTime - startTime));
    System.out.println(qsatResult.getQsatDump());
}
```