COP5556 Assignment 1

Implement a scanner for a programming language with the following lexical structure:

InputCharacter ::= any 7-bit ASCII character
LineTerminator ::= LF | CR | CR LF

LF is the ASCII character also known as “newline”. The Java character literal is ‘\n’.
CR is the ASII character also known as “return”, The Java character literal is ‘\r’.
CR immediately followed by LF counts as one line terminator, not two.

Input ::= (WhiteSpace | Comment | Token)*

WhiteSpace ::= SP | HT | FF | LineTerminator

SP is the ASCII character also known as “space”. The Java char literal is ‘ ‘.
HT is the ASCII character also known as “horizontal tab”. The Java char literal is ‘\t’.
FF is the ASCII character also known as “form feed”. The Java char literal is ‘\f’.

Comment ::= / * ((* NOT (/)) | NOT(*))* *** /
UPDATED 1/25, can have at least one * before final /

Token ::= Identifier | Keyword | Literal | Separator | Operator
Identifier ::= IdentifierChars but not a Keyword or BooleanLiteral
IdentifierChars ::= IdentiferStart IdentifierPart*
IdentifierStart ::= A..Z | a..z
IdentifierPart ::= IdentifierStart | Digit | _ | $
Literal ::= IntegerLiteral | FloatingPointLiteral | BooleanLiteral
IntegerLiteral ::= 0 | NonZeroDigit Digit*
FloatingPointLiteral ::= NonZeroDigit Digit* **.** Digit* | **.** Digit Digit* | 0 . Digit*   UPDATED 1/23
NonZeroDigit ::= 1 .. 9
Digit ::= NonZeroDigit | 0
BooleanLiteral ::= true | false
Separators ::= (|) | [|] | ; | , | { | } | << | >> | .   UPDATED 1/22, add . to separators
Operators ::=   > | < | ! | ? | : | == | != | <= | >= |
& | | | + | - | * | / | % | ** | := | @
Keywords ::= Z | default_width | default_height | show | write | to | input | from
| cart_x | cart_y | polar_a | polar_r | abs | sin | cos | atan | log
| image | int | float | filename
| boolean | red | blue | green | alpha | while | if | width | height

- If an illegal character is encountered, your scanner should throw a LexicalException. The message should contain useful information about the error. The contents of the message will not be graded, but you will appreciate it later if it is descriptive.

- If a numeric literal is provided that is out of the range of the Java equivalent of that type, then your scanner should throw a Lexical exception. (Hint: You can use the Java method Float.isFinite to test the range of float values.) The contents of the error message will not be graded, but you will appreciate it later if it is descriptive.
- Use the provided Scanner.java and ScannerTest.java as starting points.

Turn in a jar file containing the source code Scanner.java and ScannerTest.java.

Your ScannerTest will not be graded, but may be looked at in case of academic honesty issues.

We will subject your scanner to our set of junit tests and your grade will be determined solely by how many tests are passed. Name your jar file in the following format:
`firstname_lastname_ufid_hw1.jar`

Additional requirements:

- This code must remain in package cop5556sp18 (case sensitive): do not create additional packages.
- Names (of classes, method, variables, etc.) in starter code must not be changed. You may, of course, add additional variables, methods, enums, etc.
- Your code should not import any classes other than those from the standard Java distribution.

Comments and suggestions:

- As given, Scanner.java and ScannerTest.java should compile correctly. When executed, only three tests will pass, but all should pass in your completed scanner. You will need to add additional JUnit tests as you go along.
- **Work incrementally: add a single capability along with a junit test to exercise it.**
- Plan your approach. Pay attention to things that are basically the same--for example, if you can handle a semi-colon you can handle a comma, and all other characters that only appear by themselves in Tokens the same way.
- The scanner will be part of all the subsequent assignments. Errors may cause failures in subsequent assignments. A careful job now, including a complete test suite developed now will help you later.

Submission Checklist

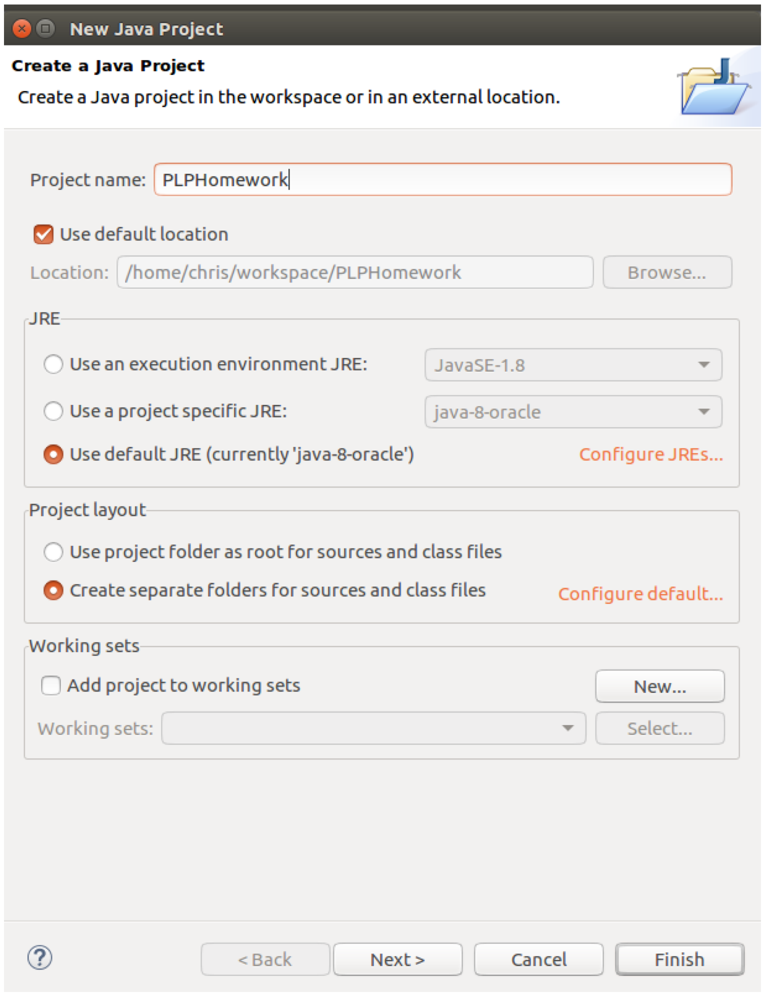
- **Make sure that sources are included in the jar file**. Many IDEs (including Eclipse) do not do this by default.
  - A quick reference for how to export a jar file from Eclipse
  - If you are not using Eclipse, check Creating a JAR file

- To ensure that we will be able to compile and run your submission: upload your jar file to one of the UF CISE servers, e.g. storm.cise.ufl.edu, uncompress it, and run it from the command line. (See https://www.cise.ufl.edu/help/access/remote for information about remote access to CISE machines) Instructions:
    - Copy/upload your file to a CISE server. Suppose your CISE ID is *username*, the following instruction will upload the *HW1.jar* to your home folder on cise server:
      ```
      scp /my/path/to/HW1.jar username@storm.cise.ufl.edu:~/
      ```
    - Uncompress file:
      ```
      jar xf HW1.jar
      ```
        - If you packaged everything correctly, your uncompressed project directory structure will looks like following:
          ```
          cop5556sp18
                  |--Scanner.java
                  |--ScannerTest.java
                  |-- *all the other files*
                  |-- ...
          ```
    - Compile:
      ```
      javac -cp .:/usr/share/java/junit4.jar:/usr/share/java/hamcrest-core.jar
      cop5556sp18/*.java
      ```
    - Run junit test from command line:
      ```
      java -cp .:/usr/share/java/junit4.jar:/usr/share/java/hamcrest-core.jar
      org.junit.runner.JUnitCore cop5556sp18.ScannerTest
      ```
- **Make sure that your jar file has the same directory structure as the original one that you downloaded from Canvas. If it doesn't, the grading script will fail resulting in a grade of 0.**
- Note that you can try this upload before you are finished with the assignment, giving you time to figure out what is wrong if you have problems.
- **No matter how your program runs on your own machine, if it fails to compile/run on the CISE server (storm or thunder) with the aforementioned instructions, your submission will get a zero grade, and there will be no regrade. So double check before your submission.**
- If you are non-CISE student who does not have a CISE account, see https://www.cise.ufl.edu/help/account for instructions to get one.
- If you use Eclipse, we suggest creating a project and importing the jar files (eg. HW1.jar) provided by each assignment into the project. After completing your work on the source files (keep all source files within the package cop5556sp18), you can export the package cop5556sp18 as a jar file for submission (remember to select the option of including source files in the jar package), so that it will have the same directory structure as the original jar file.
- You can submit multiple times and we will only grade your latest submission. The system will append a suffix to the file name for resubmissions—this is OK.
- **Important!!! It is YOUR RESPONSIBILITY to submit the assignment on time, where time is determined by Canvas. If you wait until the last minute, it is possible that your submission will fail due to server overload. Start early and submit early.**
- Double check your Canvas submissions has uploaded properly by navigating away from the submission page, then returning and downloading your submission. In order to

receive consideration for any Canvas problems, you must submit a report to the UF help desk and forward your trouble ticket to the TA.

A Quick Tutorial on How to Start Homework 1 in Eclipse: (replace fa17 with sp18 wherever it appears)

1. Create a project (e.g. PLPHomework)
   `File->New->Java Project`



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: PLPHomework

[✓] Use default location

Location: /home/chris/workspace/PLPHomework

JRE

( ) Use an execution environment JRE: JavaSE-1.8

( ) Use a project specific JRE: java-8-oracle

(●) Use default JRE (currently 'java-8-oracle')

Project layout

( ) Use project folder as root for sources and class files

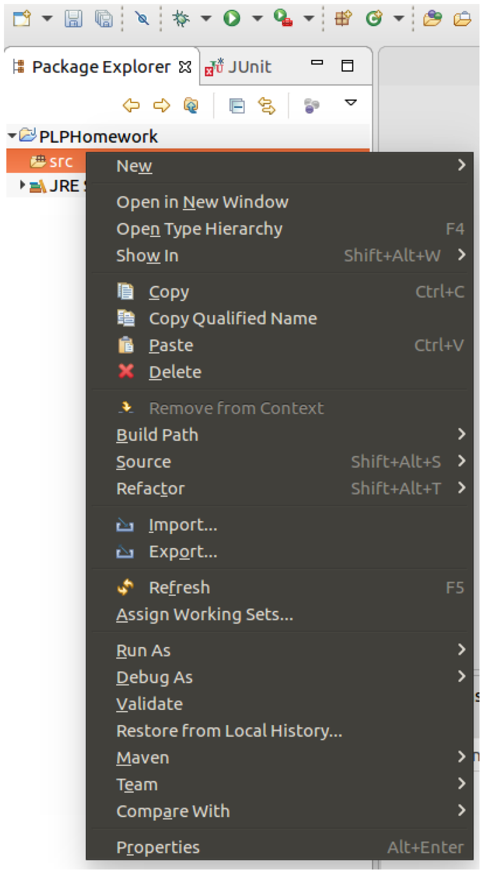
(●) Create separate folders for sources and class files

Working sets

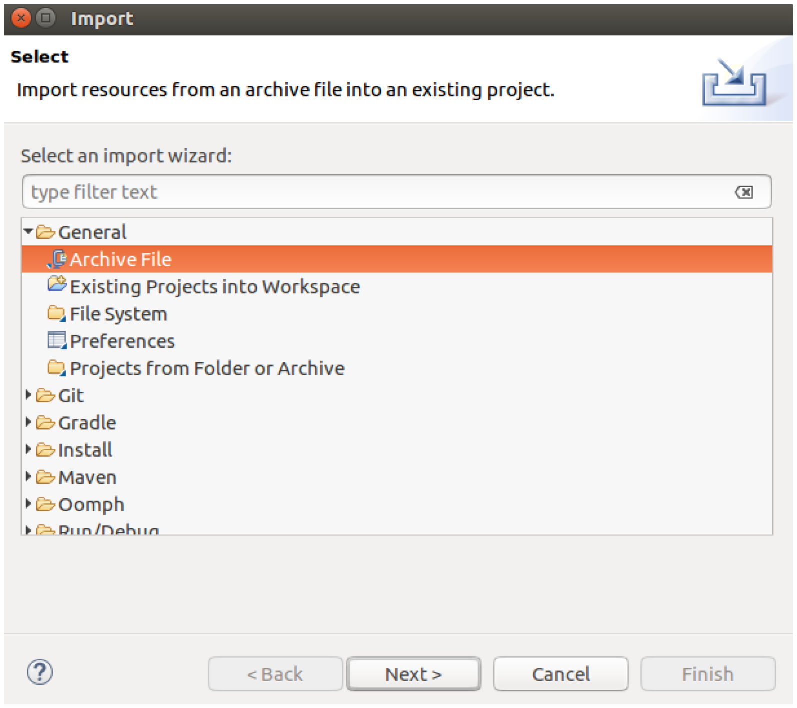
[ ] Add project to working sets

Working sets:

2. After project created, right click on the src folder in the left sidebar, choose `Import…`



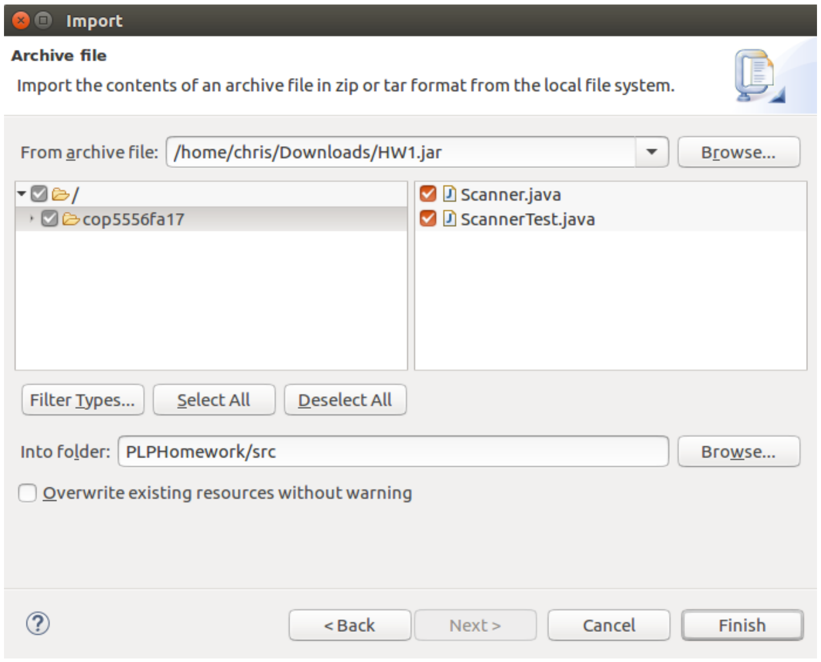
Select `General->Archive File`



Import resources from an archive file into an existing project.

Browse and choose your downloaded `HW1.jar`, make sure both `Scanner.java` and

`ScannerTest.java` have been checked



Import

Archive file

Import the contents of an archive file in zip or tar format from the local file system.

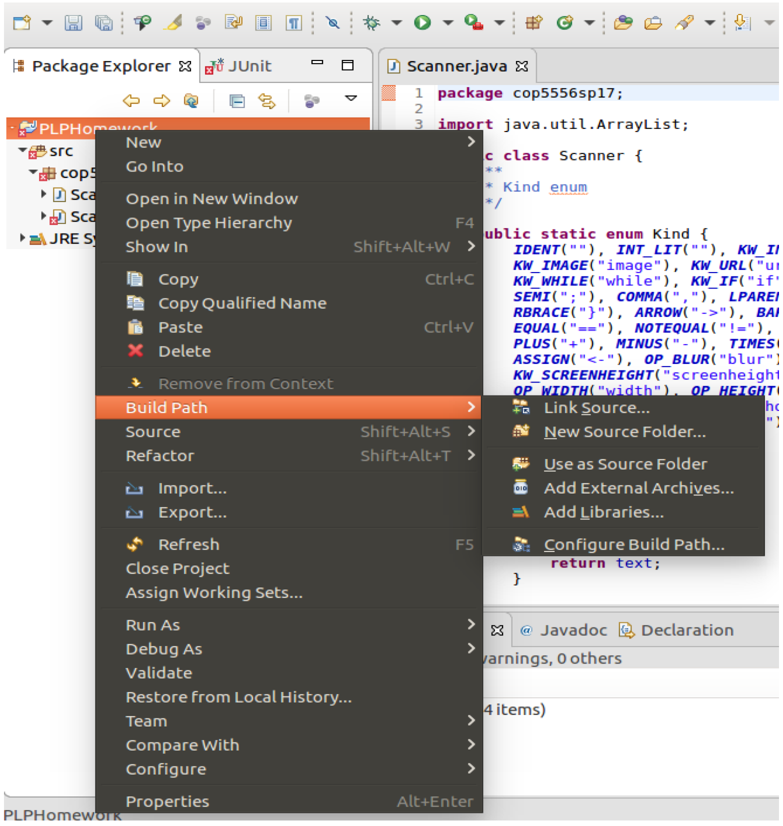
From archive file: /home/chris/Downloads/HW1.jar

- [✓] /
  - [✓] cop5556fa17
- [✓] Scanner.java
- [✓] ScannerTest.java

Into folder: PLPHomework/src

[ ] Overwrite existing resources without warning

3. Add Junit Library to Build Path.
   Right Click on project, select `Build Path->Add Libraries…`



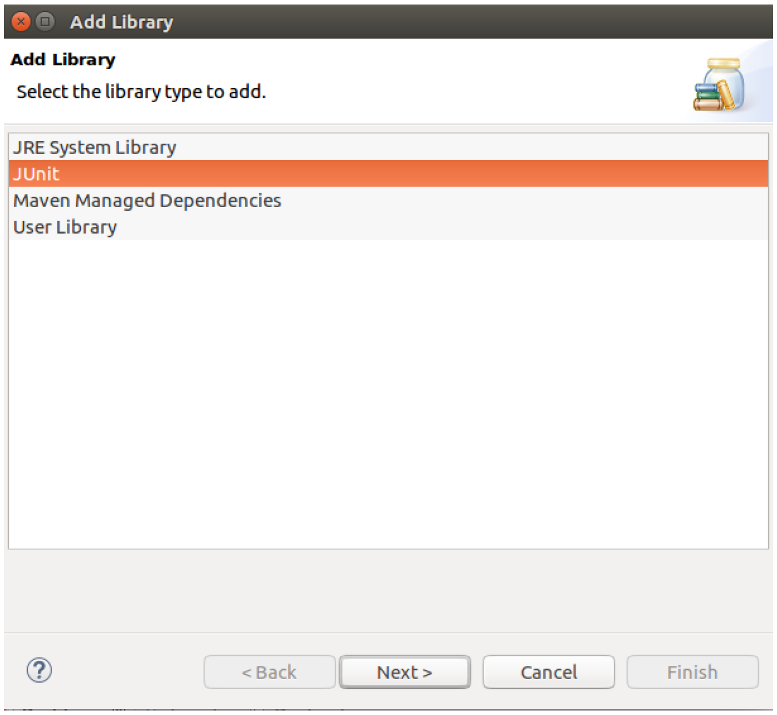
```
package cop5556sp17;

import java.util.ArrayList;

c class Scanner {
**
* Kind enum
*/

ublic static enum Kind {
    IDENT(""), INT_LIT(""), KW_I
    KW_IMAGE("image"), KW_URL("u
    KW_WHILE("while"), KW_IF("if"
    SEMI(";"), COMMA(","), LPARE
    RBRACE("}"), ARROW("->"), BA
    EQUAL("=="), NOTEQUAL("!="),
    PLUS("+"), MINUS("-"), TIMES
    ASSIGN("<-"), OP_BLUR("blur"
    KW_SCREENHEIGHT("screenheight
    OP_WIDTH("width"), OP_HEIGHT
        return text;
    }
```

In the list, choose JUnit



Add Library

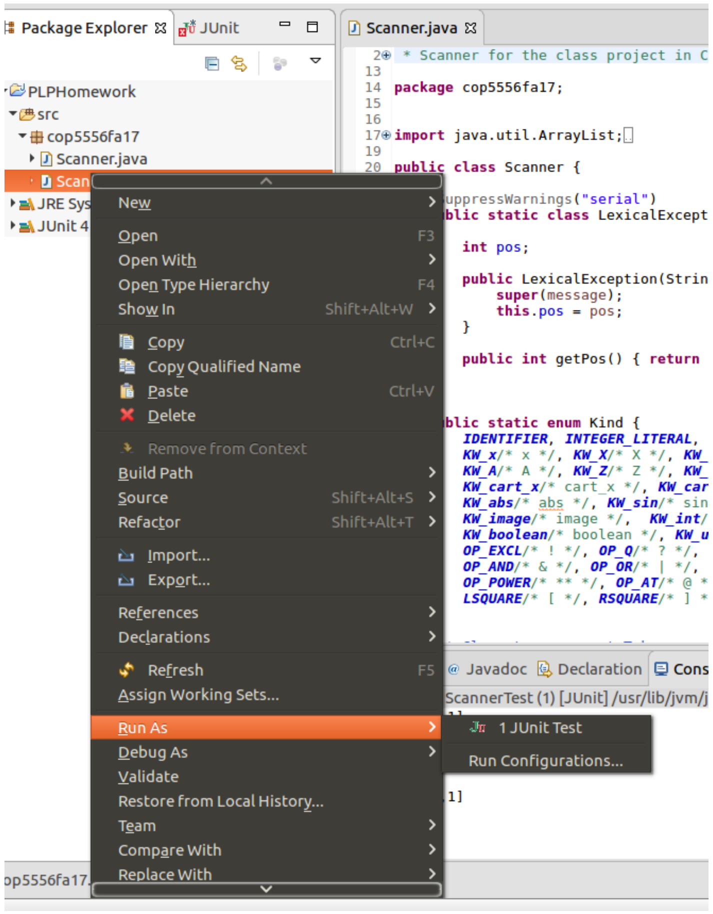
Add Library

Select the library type to add.

JRE System Library
JUnit
Maven Managed Dependencies
User Library

< Back | Next > | Cancel | Finish

4. To Run the unit tests



```
 * Scanner for the class project in C

package cop5556fa17;

import java.util.ArrayList;

public class Scanner {

uppressWarnings("serial")
blic static class LexicalExcept

    int pos;

    public LexicalException(Strin
        super(message);
        this.pos = pos;
    }

    public int getPos() { return


blic static enum Kind {
    IDENTIFIER, INTEGER_LITERAL,
    KW_x/* x */, KW_X/* X */, KW_
    KW_A/* A */, KW_Z/* Z */, KW_
    KW_cart_x/* cart_x */, KW_car
    KW_abs/* abs */, KW_sin/* sin
    KW_image/* image */,  KW_int/
    KW_boolean/* boolean */, KW_u
    OP_EXCL/* ! */, OP_Q/* ? */,
    OP_AND/* & */, OP_OR/* | */,
    OP_POWER/* ** */, OP_AT/* @ *
    LSQUARE/* [ */, RSQUARE/* ] *
```

```
ScannerTest (1) [JUnit] /usr/lib/jvm/j
1]
```