COP5556 Assignment 1

Due: 11 Sept 2017 at 11:59pm

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

RawInputCharacter ::= any ASCII character
LineTerminator ::= LF | CR | CR LF

LF is the ASCII character also known as "newline", in java \n
CR is the ASII character also known as "return", in Java, the char \r
CR immediately followed by LF counts as one line terminator, not two

InputCharacter ::= RawInputCharacter, but not CR or LF
Input ::= (WhiteSpace | Comment | Token)*
Token ::= Identifier | Keyword | Literal | Separator | Operator
WhiteSpace ::= SP | HT | FF | LineTerminator

SP is the ASCII character also known as "space"
HT is the ASCII character also known as "horizontal tab"
FF is the ASCII character known also known as "form feed"

Comment ::= / / InputCharacter*
Identifier ::= IdentifierChars but not a Keyword or BooleanLiteral
IdentifierChars ::= IdentiferStart IdentifierPart*
IdentifierStart ::= A..Z | a..z | _ | $
IdentifierPart ::= IdentifierStart | Digit
Literal ::= IntegerLiteral | BooleanLiteral | StringLiteral
IntegerLiteral ::= 0 | NonZeroDigit Digit*
NonZeroDigit ::= 1 .. 9
Digit ::= NonZeroDigit | 0
BooleanLiteral ::= true | false
StringLiteral ::= " StringCharacter* "
StringCharacter::= InputCharacter but not " or \
          | EscapeSequence
EscapeSequence ::=
          \ b | \ t | \ n | \ f | \r | \" | \' | \ \
Separators ::= (|) | [|] | ; | ,
Operators ::= = | > | < | ! | ? | : | == | != | <= | >= |
          & | | | + | - | * | / | % | ** | -> | <- | @
Keywords ::= x | X | y | Y | r | R | a | A | Z | DEF_X | DEF_Y | SCREEN |
          cart_x | cart_y | polar_a | polar_r | abs | sin | cos | atan | log | image | int |
          boolean | url | file

- 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 helpful.
- If an integer literal is provided that is out of the range of a Java int, then your scanner should throw a LexicalException. The contents of the error message will not be graded, but you will appreciate it later if it is helpful.
- 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 cop5556fa17(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:

- The given Scanner.java and ScannerTest.java should compile correctly. When executed, only one test will pass, but all should pass in your completed scanner.
- **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. Similarly, for = (which may be the first char in an = Token or an == Token) and !, which may be the first token in a ! Token or a != Token.

- If you use Integer.parseInt to get the value of a numeric literal, it will throw a NumberFormatException if the value is too large. This is useful functionality, but the exception is not the same one as specified. You need to catch it and throw a Scanner.LexicalException instead with a useful message.
- Character.isJavaIdentifierPart mysteriously also returns true when it is given a value of 0, the sentinel value we are using to mark the end of the input, so you will probably need to check for this case.
- Potentially useful methods from the java.lang.Character class include isDigit, isWhiteSpace, isJavaIdentifierStart, isJavaIdentifierPart, along with java.lang.Integer.parseInt. In my solution, I used java.util.HashMap to map keywords to their Token.kind.

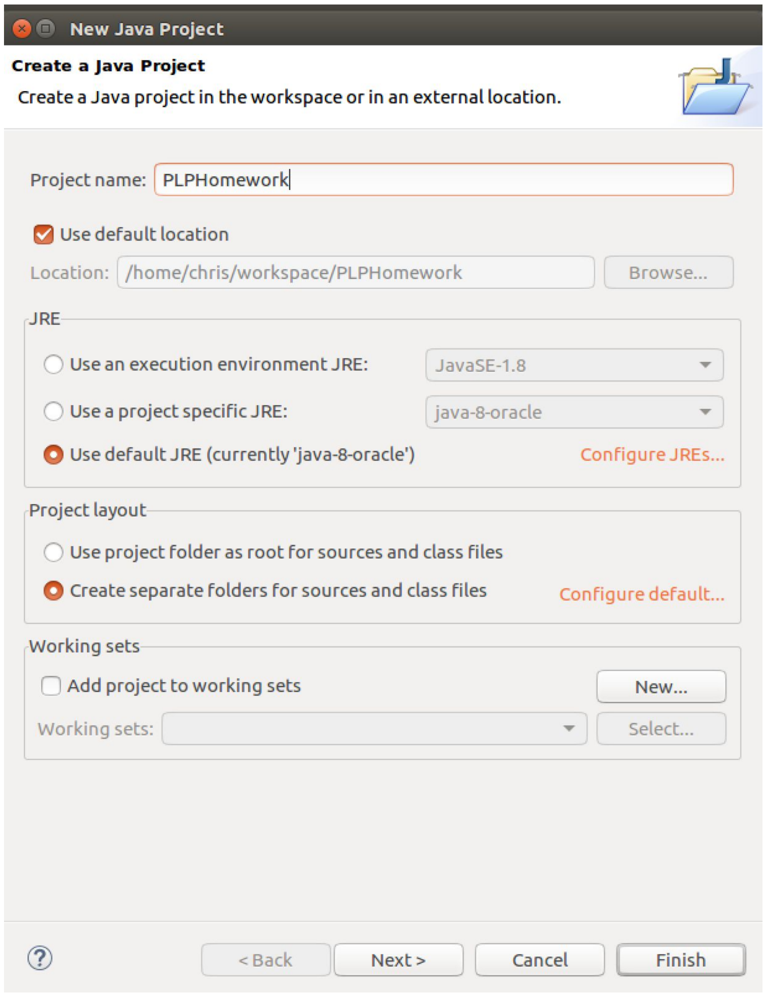
Submission Checklist

- **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 server, e.g. storm.cise.ufl.edu, uncompress it and run from the command line. Instructions:
  - Copy/upload your file to cise server. If your OS is windows, install some ssh client like putty for this step. Or you can use some ftp client(e.g. Filezilla) and skip this step. 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:
      ```
      cop5556fa17
              |--Scanner.java
              |--ScannerTest.java
              |-- *all the other files*
              |-- ...
      ```
  - Compile:
    ```
    javac -cp .:/usr/share/java/junit4.jar:/usr/share/java/hamcrest-core.jar
    cop5556fa17/*.java
    ```
  - Run junit test from command line:
    ```
    java -cp .:/usr/share/java/junit4.jar:/usr/share/java/hamcrest-core.jar
    org.junit.runner.JUnitCore cop5556fa17.ScannerTest
    ```
- **Make sure that your jar file has the same directory structure as the original one that you downloaded from Canvas(otherwise, you will fail grading and get 0 grade).**

- Note that you can do this upload in the process so if you have problems, you will have time to figure out what is wrong.
- **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 is no regrade. So double check before your submission.**
    - If you are non-CISE student who does not have a CISE account, google and consult the ithelp of CISE to get one account.
- 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 cop5556fa17), you can export the package cop5556fa17 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.
- **Important!!! It is YOUR RESPONSIBILITY to submit the assignment in time. If you keep failing submission on Canvas, contact TAs and ithelp for help. DO NOT jammed submitting at last 10 min before deadline, it's possible to fail submission at that time due to the server overloading. Start early, submit early. You can submit multiple times(not recommended), and we will only grade your last submission.**

A Quick Tutorial on How to Start Homework 1 in Eclipse:

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

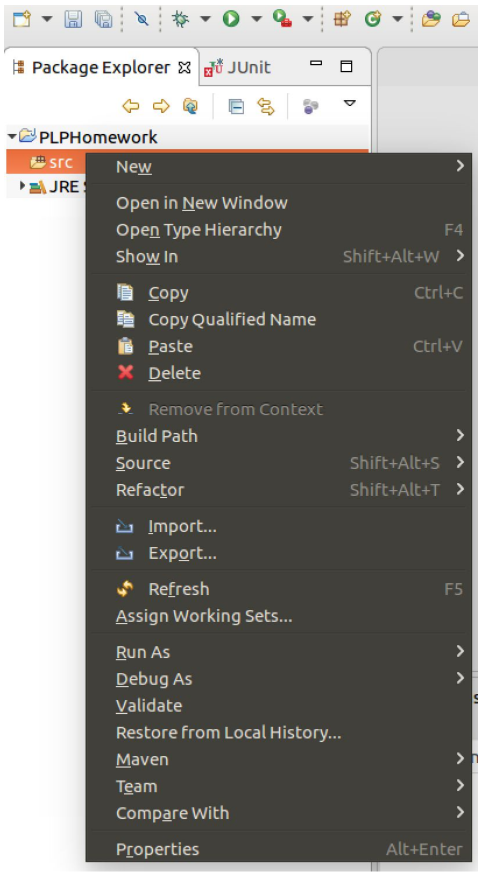
(●) Create separate folders for sources and class files

Working sets

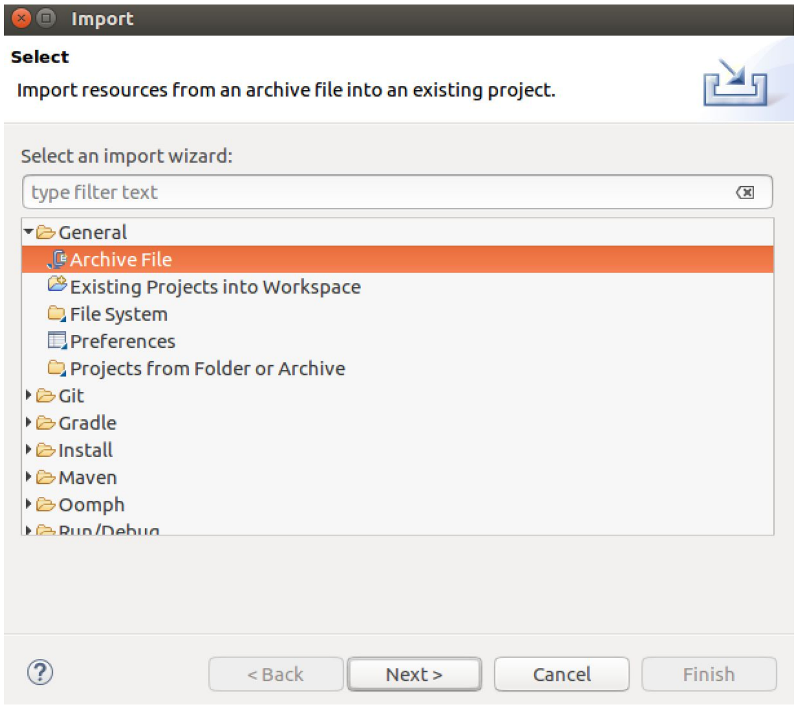
[ ] Add project to working sets

Working sets:

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



Select `General->Archive File`


Import

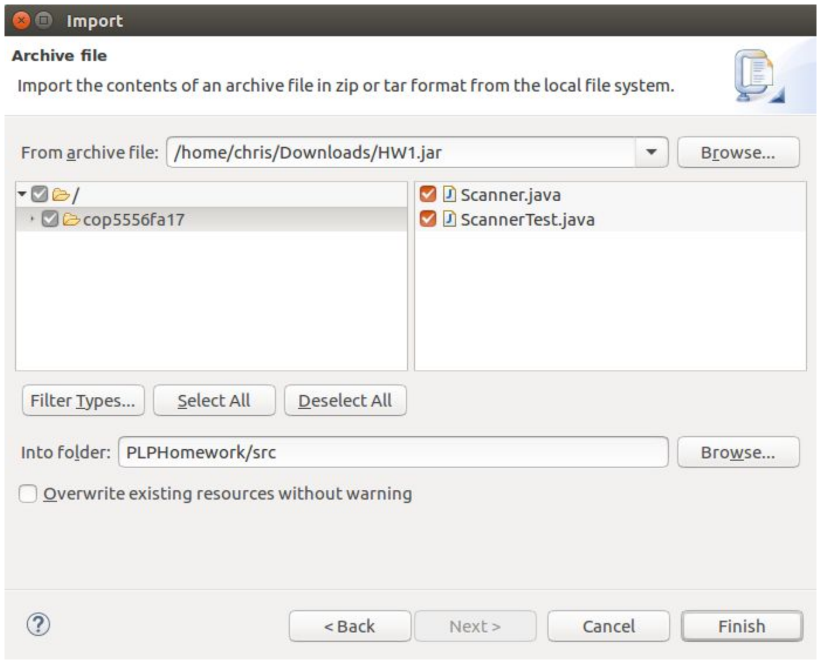
Select

Import resources from an archive file into an existing project.

Select an import wizard:

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.

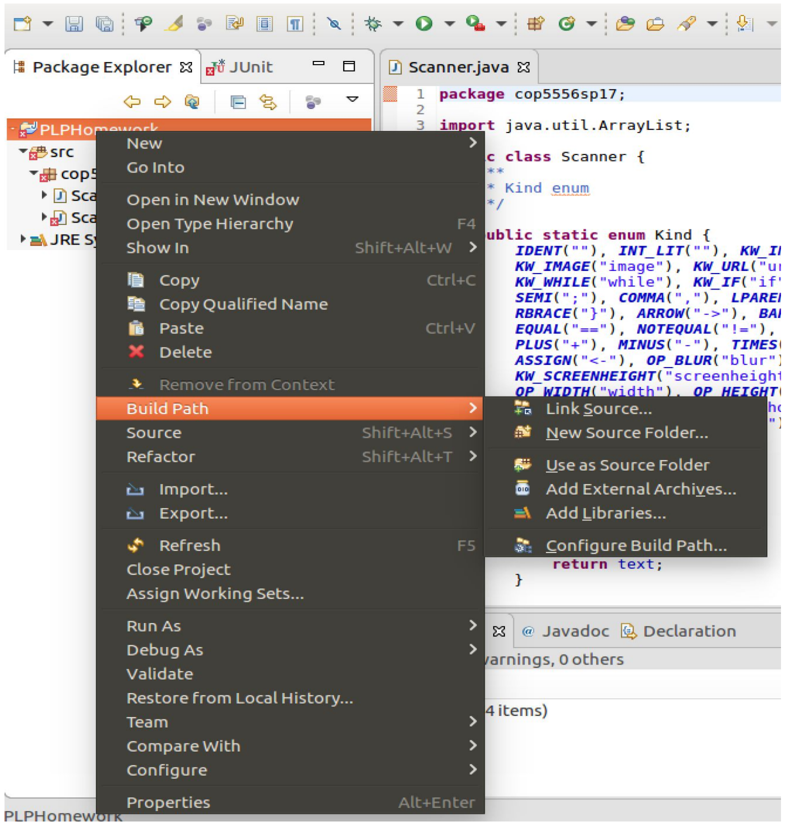
From archive file: /home/chris/Downloads/HW1.jar

- [✓] /
  - [✓] 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…`



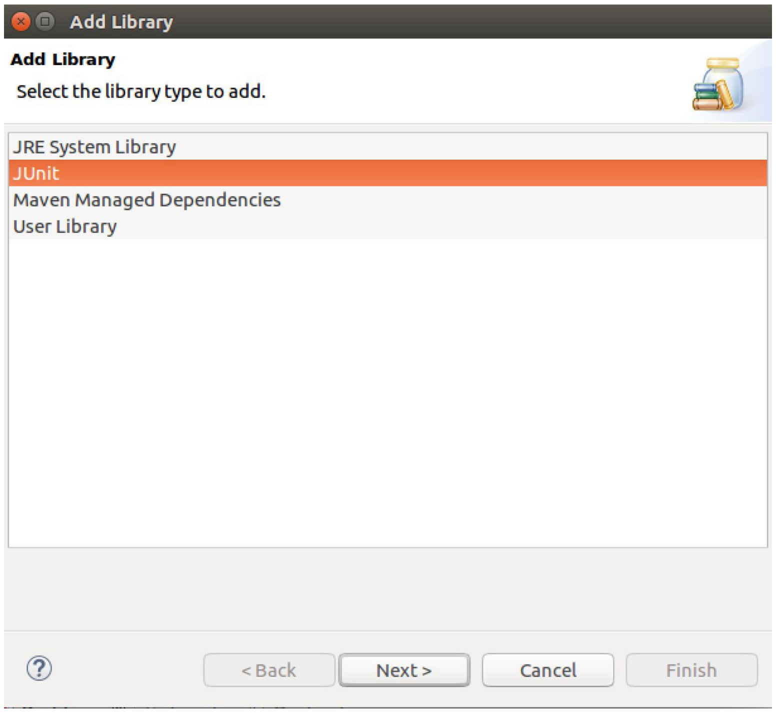
```
package cop5556sp17;

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("screenheigh
    OP_WIDTH("width"), OP_HEIGHT
        return text;
    }
```

In the list, choose JUnit

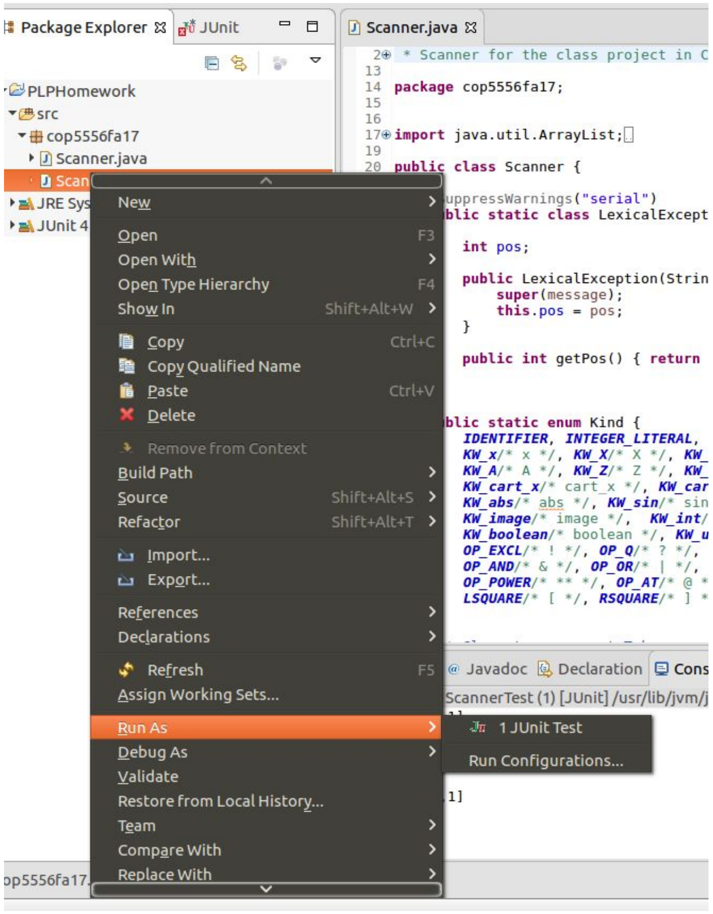


Add Library

Select the library type to add.

- JRE System Library
- JUnit
- Maven Managed Dependencies
- User Library

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]
```