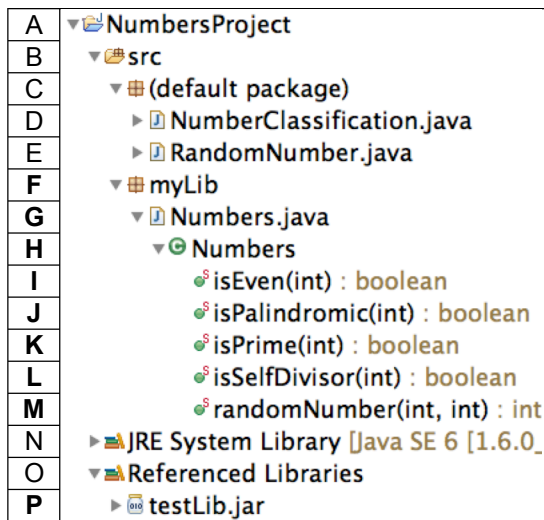


# WHAT SHOULD BE IN YOUR NUMBERSPROJECT

The actual name of the JavaProject that you use won't matter, in this example, it is called "NumbersProject" but you can use any name you like!

Below is a picture of what you should see if you open the disclosure arrows as shown for your project:



For each letter you see, there is an entry explaining what the entry represents.

If your project doesn't have an entry or the entry is different ask for assistance!

**When the assignment is due, you will be given a new version of testLib.jar to put in your project and if it can't run successfully because your project doesn't match the expected layout, you will receive a grade of "F."**

Please be aware that some entries must match *exactly* what is shown on the right while others can be different!

The items that **must** match exactly have **bold** letters next to them!

## A. The Project Folder Name.

You can call your project anything you want so this name doesn't need to match the name shown!

## B. The src Folder.

Every project should have a folder named `src` (`src` stands for source) that contains the packages that in turn contain the source code that makes up your project.

## C. The (default package) package.

The default package should contain any classes that you have written that contain a `main()` method. The contents of your default package may contain more items than those shown!


## D. The NumberClassification.java class file.

The exact name of this file doesn't matter. This file should contain the `main()` method with the code in it that will output the numbers from 0 to 100, identifying if each number is even, prime, a self-divisor, and palindromic.


You can see required import statement and JavaDoc code that should be part of your program.

This is followed by partial code that might be in your program.


```
01 import myLib.Numbers;
02 /**
03  * A program description goes here!
04  *
05  * @author Last, First (Period X)
06  * @version Last Modified MM/DD/YYYY
07  */
08 public class NumberClassification {
09     public static void main(String[] args) {
10         int num=0;
11         while (num<=100) { // Loop through the numbers 0-100
12             System.out.print(num);
13             if (Numbers.isEven(num)) {
14                 System.out.print(" is even");
15             }
16             System.out.println(".");
17             num=num+1;
18         }
19     }
20 }
```

E. The  RandomNumber.java class file.  
The exact name of this file doesn't matter. This file should contain the main() method shown the right:


```
01 import testLib.Test;
02 public class RandomNumber {
03     public static void main(String[] args) {
04         Test.testRandomNumber(1000,null);
05     }
06 }
```

F. The  myLib package.

It is absolutely **critical** that this entry is spelled precisely as shown with no extra characters and each letter having the correct case (capital or lower case).

G. The  Numbers.java class file.


When the disclosure arrow is pointing down as shown in the diagram, you can see an entry indented below it that shows the names of any classes that are part of this source file. There should only be one entry, the Numbers class!


H. The  Numbers class.

Indented to show that it is part of the Numbers.java file, you should check the spelling of this entry closely! It must have the exact name and each letter must have the correct case.


When the disclosure arrow is pointing down in a class entry in the project explorer, indented below it you will find each of the methods (and other items that we will learn more about later) listed.

Each entry will have:


- Keyword indicator symbol: Methods that are public and static should have this symbol: 
- The method name.
- The type of each parameter in parenthesis.
- A colon and then the return type for the method.

I. The  isEven(int) : boolean method.


Your entry must match **exactly!**

J. The  isPalindromic(int) : boolean method.


Your entry must match **exactly!**

K. The  isPrime(int) : boolean method.

Your entry must match **exactly!**

L. The  isSelfDivisor(int) : boolean method.

Your entry must match **exactly!**

M. The  randomNumber(int, int) : int method.

This method won't be used for this test, but will be used later so your entry must match **exactly!**

N. The  JRE System Library entry.

Every project should have this entry. Note that the text in brown square brackets following the word Library will likely be different.

You should check to make sure it does **not** say end in a number below 4!

This indicates that you did not select the current Java Runtime Environment when you created the project or that your system does not have a current version of the Java Runtime Environment installed.

O. The  Referenced Libraries entry.

If you don't see this entry in your project then you have not yet put the testLib.jar file in your project or have not yet added that file to your build path.

See below for instruction about how to do this.

P. The  testLib.jar entry.

If you don't see this entry in your project then you have not yet put the testLib.jar file in your project or have not yet added that file to your build path.

See instructions on the next page about how to do this.

## Getting Ready to Grade Your Project

Before you can begin using the test unit you will have to follow these basic steps:

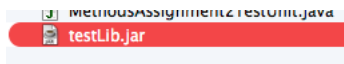
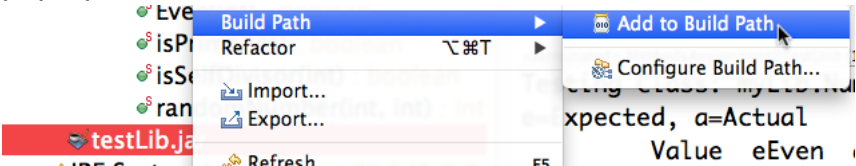
- Connect to the NAS Server and navigate to the Libraries folder on the AP\_CompSci share and verify that you can see these files:

MethodsAssignment2TestUnit.java

testLib.jar

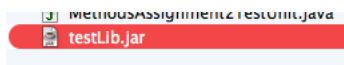
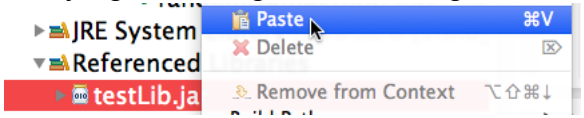
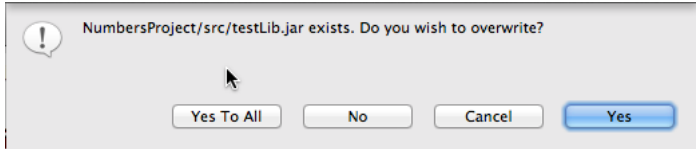
- If you don't already have testLib.jar in your project, follow the steps for "Adding testLib.jar to your project."
- If you already have testLib.jar in your project, follow the steps for "Updating testLib.jar in your project."
- Then follow the steps for "Adding MethodsAssignment2TestUnit.java to your project."

### Adding testLib.jar to Your Project

1. In the window that shows the files for AP\_CompSci/Libraries select the file testLib.jar:
- 
2. Copy the file using the appropriate keyboard shortcut or by right-clicking and selecting "copy" from the pop-up contextual menu.
  3. Switch back to eclipse and select the src folder.
  4. Use the appropriate keyboard shortcut, to paste the file into your project or right-click the src folder and select "paste" from the pop-up contextual menu.
  5. Right-click the testLib.jar file, then select "Build Path" and "Add to Build Path" from the pop-up contextual menu.
- 

You are now ready to add the testing class to your project, so skip to the "Adding MethodsAssignment2TestUnit.java to your project." section.

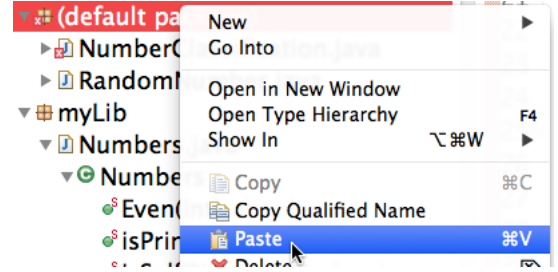
### Updating testLib.jar in Your Project

1. In the window that shows the files for AP\_CompSci/Libraries select the file testLib.jar:
- 
2. Copy the file using the appropriate keyboard shortcut or by right-clicking and selecting "copy" from the pop-up contextual menu.
  3. Switch to eclipse, right-click the testLib.jar file then select "paste" from the pop-up contextual menu.
- 
4. Click the "Yes" button in the dialog to overwrite the old testLib.jar file.
- 

You are now ready to add the testing class to your project, continue on to the "Adding MethodsAssignment2TestUnit.java to your project." section.

## Adding `MethodsAssignment2TestUnit.java` to your project

1. In the window that shows the files for AP\_CompSci/Libraries select the file `MethodsAssignment2TestUnit`:
2. Copy the file using the appropriate keyboard shortcut or by right-clicking and selecting “copy” from the pop-up contextual menu.
3. Switch to eclipse, right-click the `(default package)` package then select “paste” from the pop-up contextual menu.
4. Open the `MethodsAssignment2TestUnit` class. If you don't see any errors, you are ready to run the test unit. If you do see any errors, you likely have not followed all the steps and checked all the files closely on the prior pages of this document!



You are now ready run the `MethodsAssignment2TestUnit` class `main()` method.

## Running `MethodsAssignment2TestUnit` class `main()` method

- Click the run button, you will see output in the console window like this:

```
Testing Class: myLib.Numbers, methods: isEven(int), isPalindromic(int),
isSelfDivisor(int), isPrime(int)
e=Expected, a=Actual
    Value  eEven  aEven  ePalin  aPalin  eSelfD  aSelfD  ePrime  aPrime  #
Pass/Possible
  -2147483648  true   true   false   true   false   true   false   true    1/4
     -1      false  true   false   true   false   true   false   true    0/4
     0       true   true   true    true   true    true   false   true    3/4
     1       false  true   true    true   true    true   false   true    2/4
     2       true   true   true    true   true    true   true    true    4/4
     3       false  true   true    true   true    true   true    true    3/4
     4       true   true   true    true   true    true   false   true    3/4
    19      false  true   false   true   false   true   true    true    1/4
    20      true   true   false   true   false   true   false   true    1/4
    21      false  true   false   true   false   true   false   true    0/4
    22      true   true   true    true   true    true   false   true    3/4
    23      false  true   false   true   false   true   true    true    1/4
    99      false  true   true    true   true    true   false   true    2/4
   100      true   true   false   true   false   true   false   true    1/4
   101      false  true   true    true   false   true   true    true    2/4
  2146226412  true   true   true    true   true    true   false   true    3/4
  2146226413  false  true   false   true   false   true   true    true    1/4
Score: 31/68
```

- Look for lines that show a score of less than 4/4. Check to columns for expected and actual value for each of your methods to identify which of your methods is returning an incorrect value.
- You now know what exact test value you should use to identify and fix the problem in your code!