Packages and Jars

- Packages and Imports
- Nested Packages
- Static Imports
- CLASSPATH
- Jar files

Introduction to Java

See Also

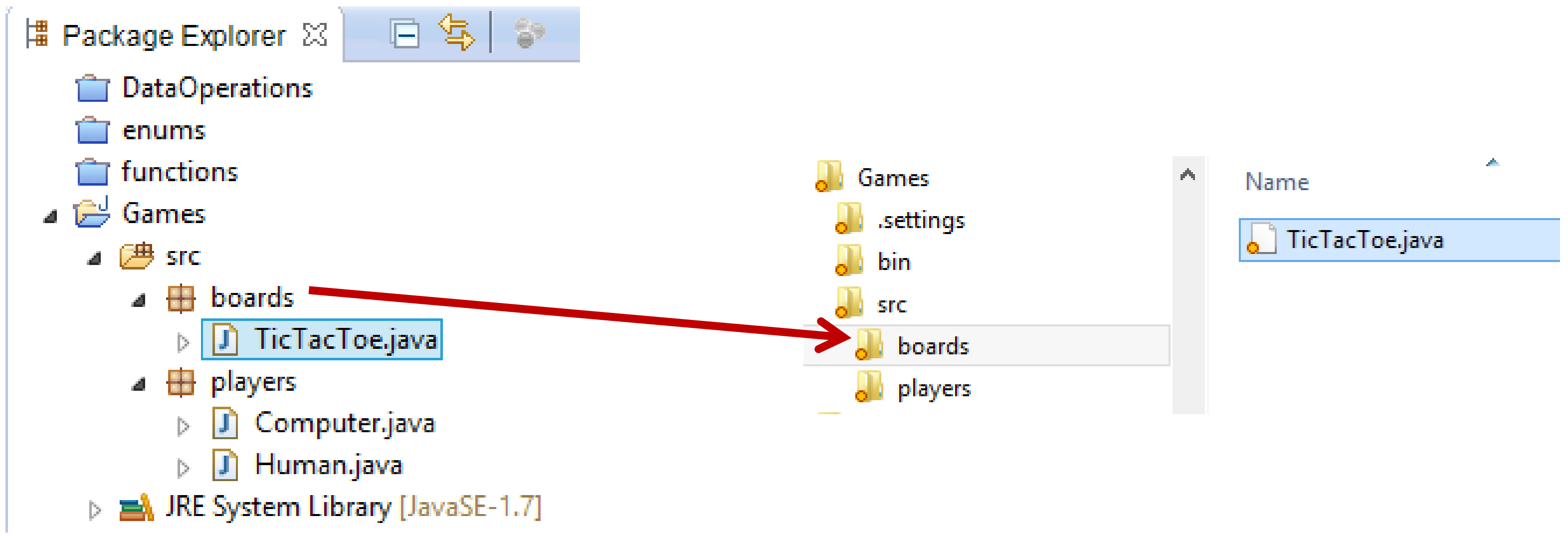
https://docs.oracle.com/javase/tutorial/java/package/packages.html

https://docs.oracle.com/javase/tutorial/java/package/

https://www.youtube.com/watch?v=l5SviD48vOQ

Packages Are Directories

- Organize related classes into packages
- Class names must be unique within a package
- A package is a directory



Packages

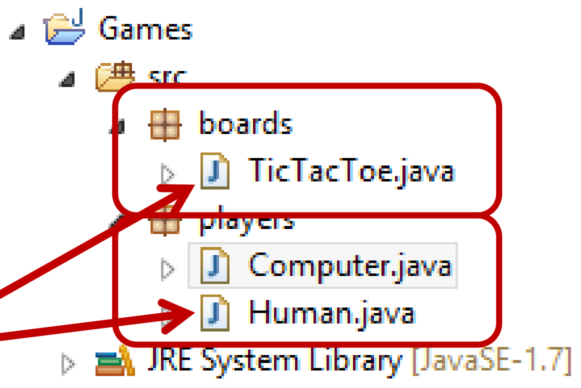
- A class in a package must have a "package" statement at the top
- Use of the "default" package is strongly discouraged
- Convention: use all lower-case names

```
package players;

public class Computer {

    public static void main(String [] args) {


    }

}
```

Imports

- The compiler can locate classes that are in the same package
- Classes in another package must be imported

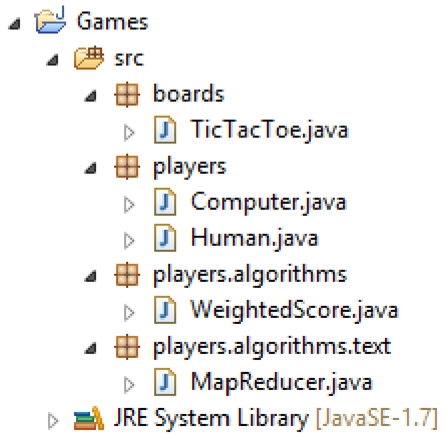
```
package players;

import boards.TicTacToe;
import boards.*;

public class Computer {

    public static void main(String [] args) {
        Human hum = new Human();
        TicTacToe t = new TicTacToe();
    }

}
```



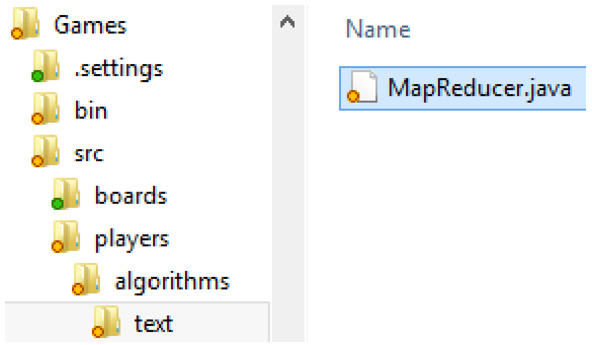
Nested Packages

- It is common to nest packages. These become nested directories.
- Use dots "." in the package statements and imports.
- There is NO special relationship between parents and children. It is strictly organizational.

```
package players.algorithms.text;

import players.algorithms.WeightedScore;

public class MapReducer {

}
```



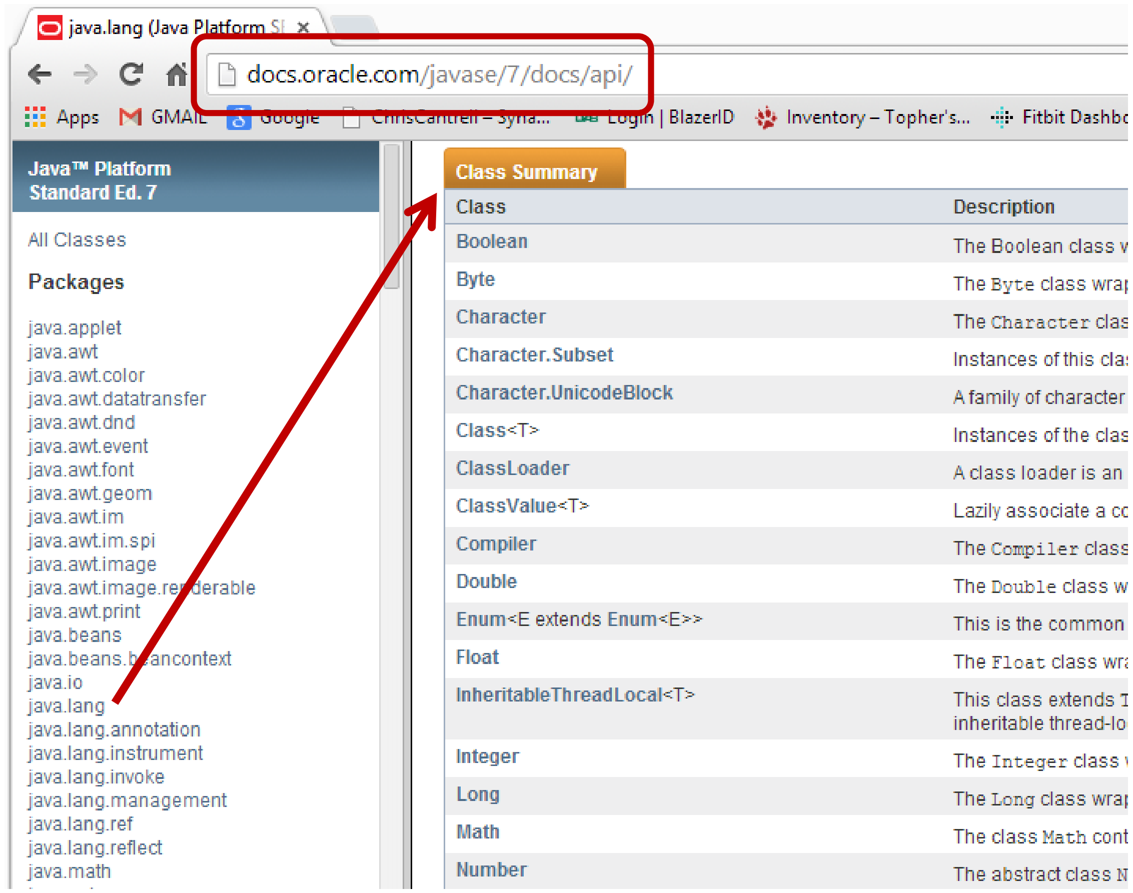
```
Games
  src
    boards
      TicTacToe.java
    players
      Computer.java
      Human.java
    players.algorithms
      WeightedScore.java
    players.algorithms.text
      MapReducer.java
  JRE System Library [JavaSE-1.7]
```



```
Games
  .settings
  bin
  src
    boards
    players
      algorithms
        text
```

MapReducer.java

java.lang.*



Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math

Class Summary

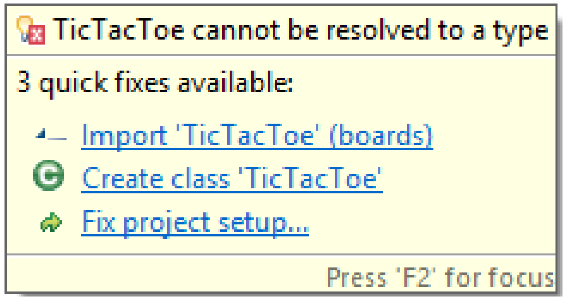
Class	Description
Boolean	The Boolean class w
Byte	The Byte class wrap
Character	The Character clas
Character.Subset	Instances of this cla
Character.UnicodeBlock	A family of character
Class<T>	Instances of the clas
ClassLoader	A class loader is an
ClassValue<T>	Lazily associate a co
Compiler	The Compiler class
Double	The Double class w
Enum<E extends Enum<E>>	This is the common
Float	The Float class wra
InheritableThreadLocal<T>	This class extends T inheritable thread-lo
Integer	The Integer class
Long	The Long class wrap
Math	The class Math cont
Number	The abstract class N

- Automatically imported
- Use the online docs
- java, javax, and org

Eclipse and Imports

- Eclipse will add the import statement for you

```
package players;

import boards.TicTacToe;

public class Computer {

    public static void main(String [] args) {
        Human hum = new Human();
        TicTacToe t = new TicTacToe();
    }

}
```



TicTacToe cannot be resolved to a type

3 quick fixes available:

Import 'TicTacToe' (boards)

Create class 'TicTacToe'

Fix project setup...

Press 'F2' for focus

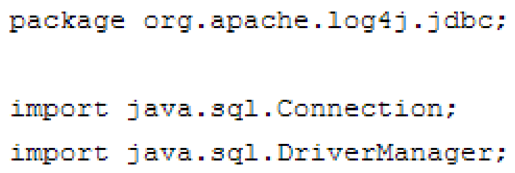
Static Imports

- Use to import the static members of another class
- Very useful for constants

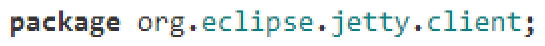
```java
package players;

import static java.lang.Math.PI;
import static java.lang.Math.E;
import static java.lang.Math.atan;

import static java.lang.Math.*;

public class Computer {

    public static void main(String [] args) {

        double d = Math.PI + Math.E * Math.atan(Math.PI);

        double e = PI + E * atan(PI);

    }

}
```

Unique Package Names

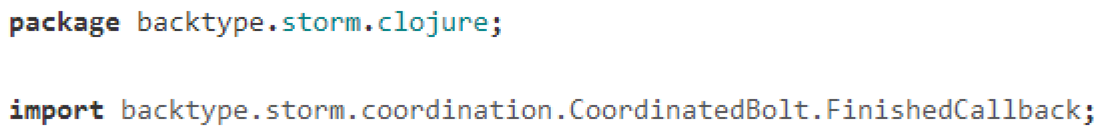
- Libraries come from lots of sources
- Must be unique to avoid name collision
- Most adopt a namespace based on their URL, which is unique (by domain registration)
- Use as much as you need (be ready for the future)



```
package org.apache.log4j.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
```



```
package org.eclipse.jetty.client;
```



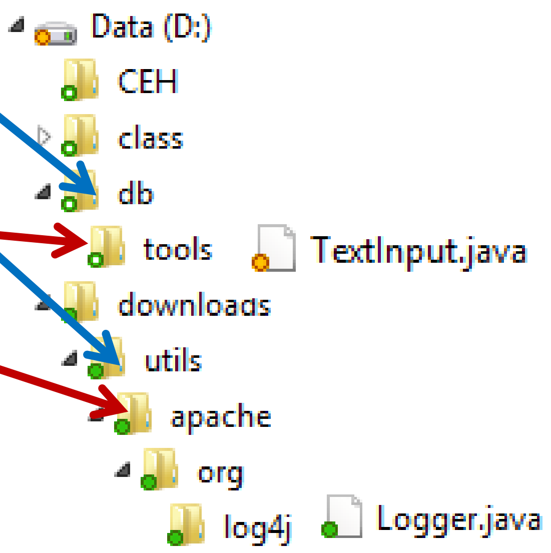
```
package backtype.storm.clojure;

import backtype.storm.coordination.CoordinatedBolt.FinishedCallback;
```

The "CLASSPATH"

- Difficult to keep all the source files for a program together in one directory
- You'll have different projects and 3rd party libraries
- The "CLASSPATH" environment variable points to "roots"

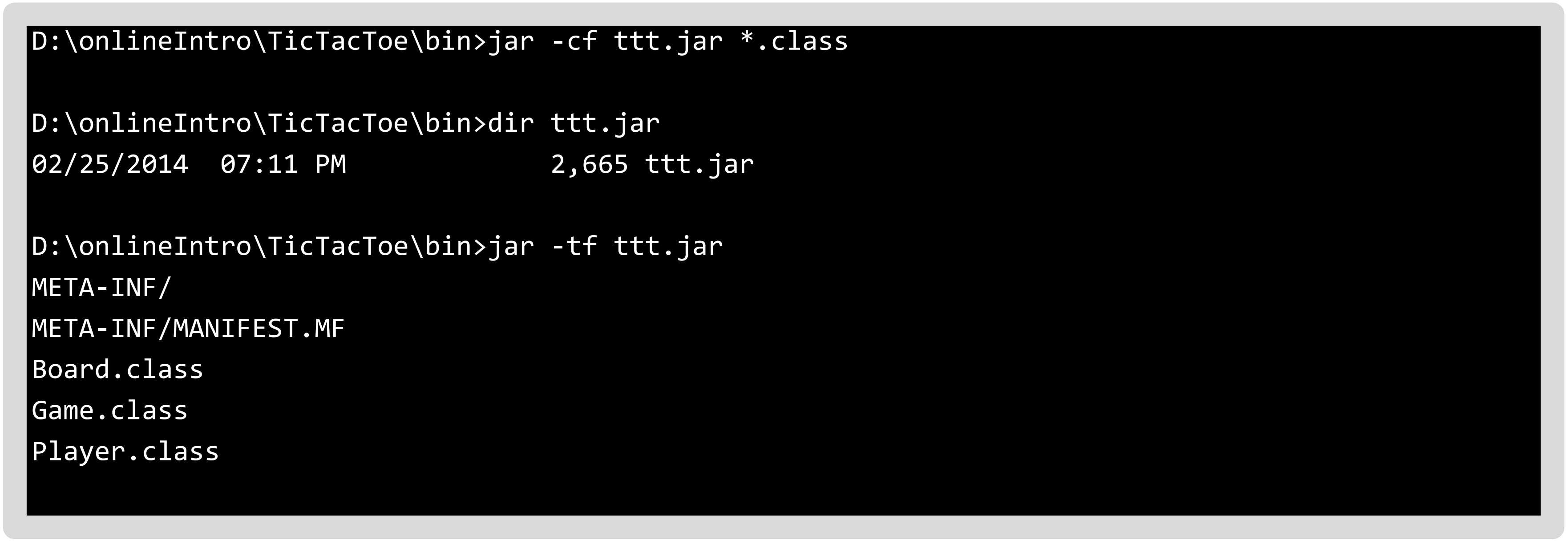
```
D:\>set CLASSPATH=d:\downloads\utils;d:\db;d:\onlineIntro\TicTacToe\src
```

```java
import apache.org.log4j.Logger;

import tools.TextInput;

public class Game {

    public static void main(String [] args) {

    }
}
```



Jar Files

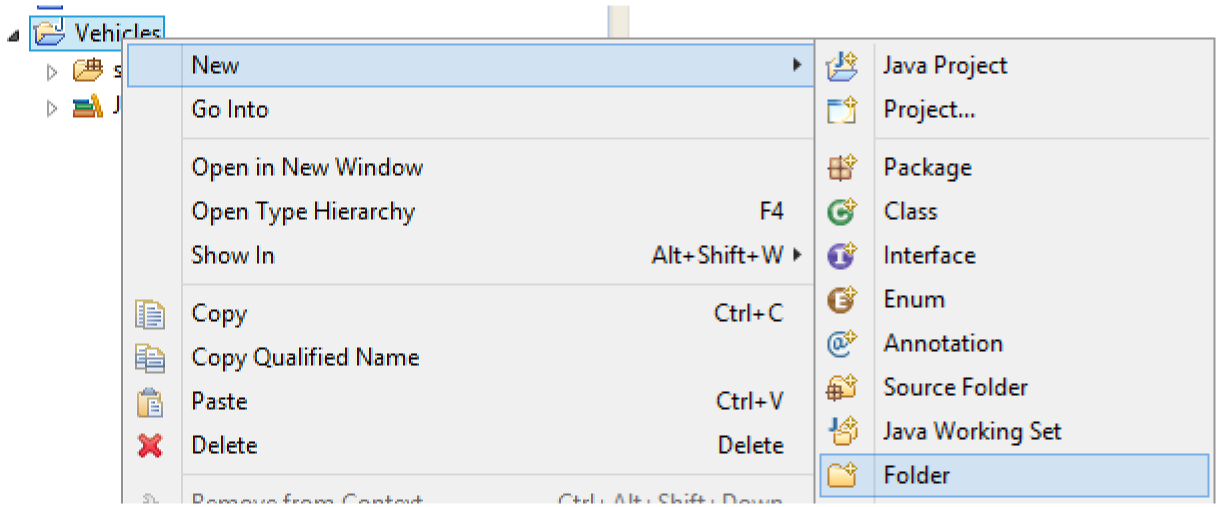
- Difficult to deliver a directory of files
- Java can read directories from ZIP files
- Use the extension ".jar"
- The "jar" tool does the zipping and unzipping
- Add jar files to the CLASSPATH too (individually)



```
D:\onlineIntro\TicTacToe\bin>jar -cf ttt.jar *.class

D:\onlineIntro\TicTacToe\bin>dir ttt.jar
02/25/2014  07:11 PM             2,665 ttt.jar

D:\onlineIntro\TicTacToe\bin>jar -tf ttt.jar
META-INF/
META-INF/MANIFEST.MF
Board.class
Game.class
Player.class
```

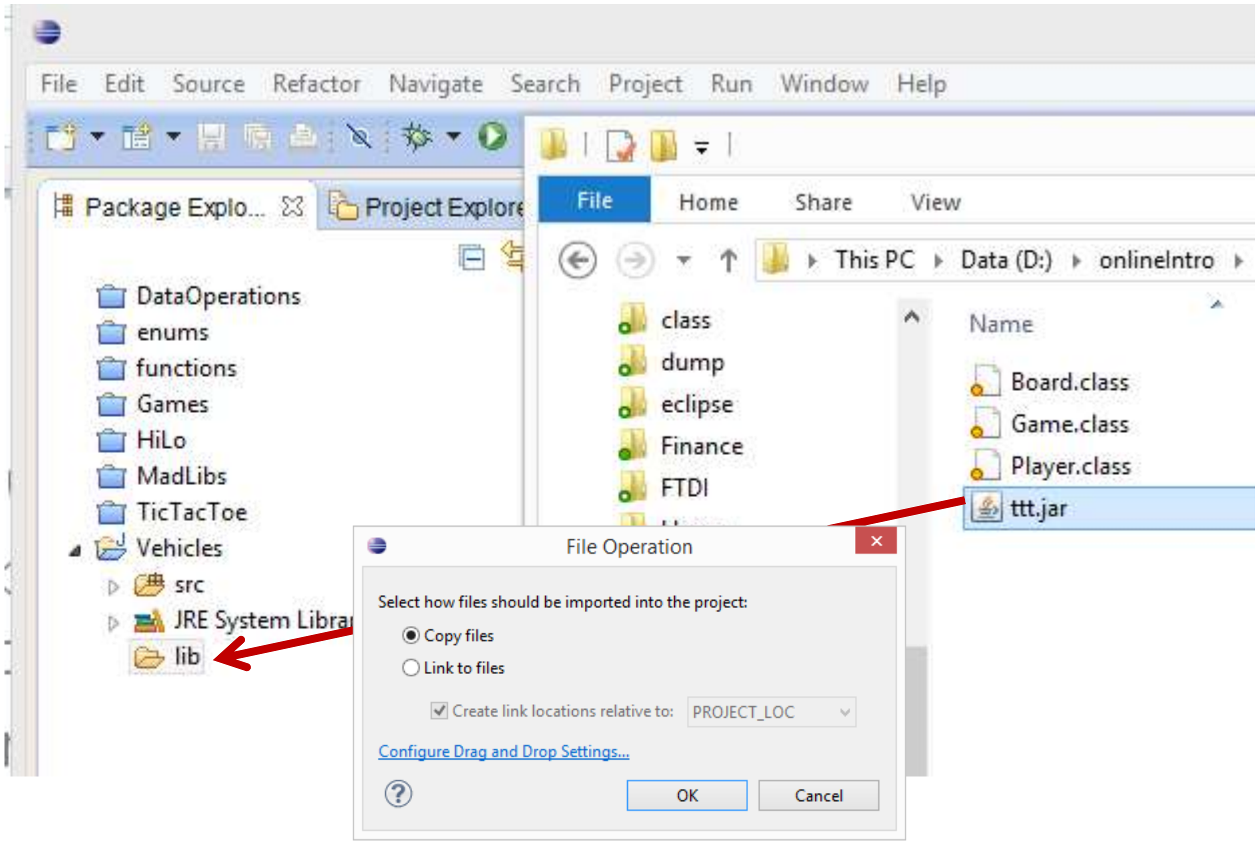
Eclipse and Jars

- You can keep JARs anywhere
- I like to drop 3rd party JARs right into the projects that use them
- I create a "lib" directory in my project



Eclipse and Jars

- Then copy the jars to the "lib" directory



Select how files should be imported into the project:

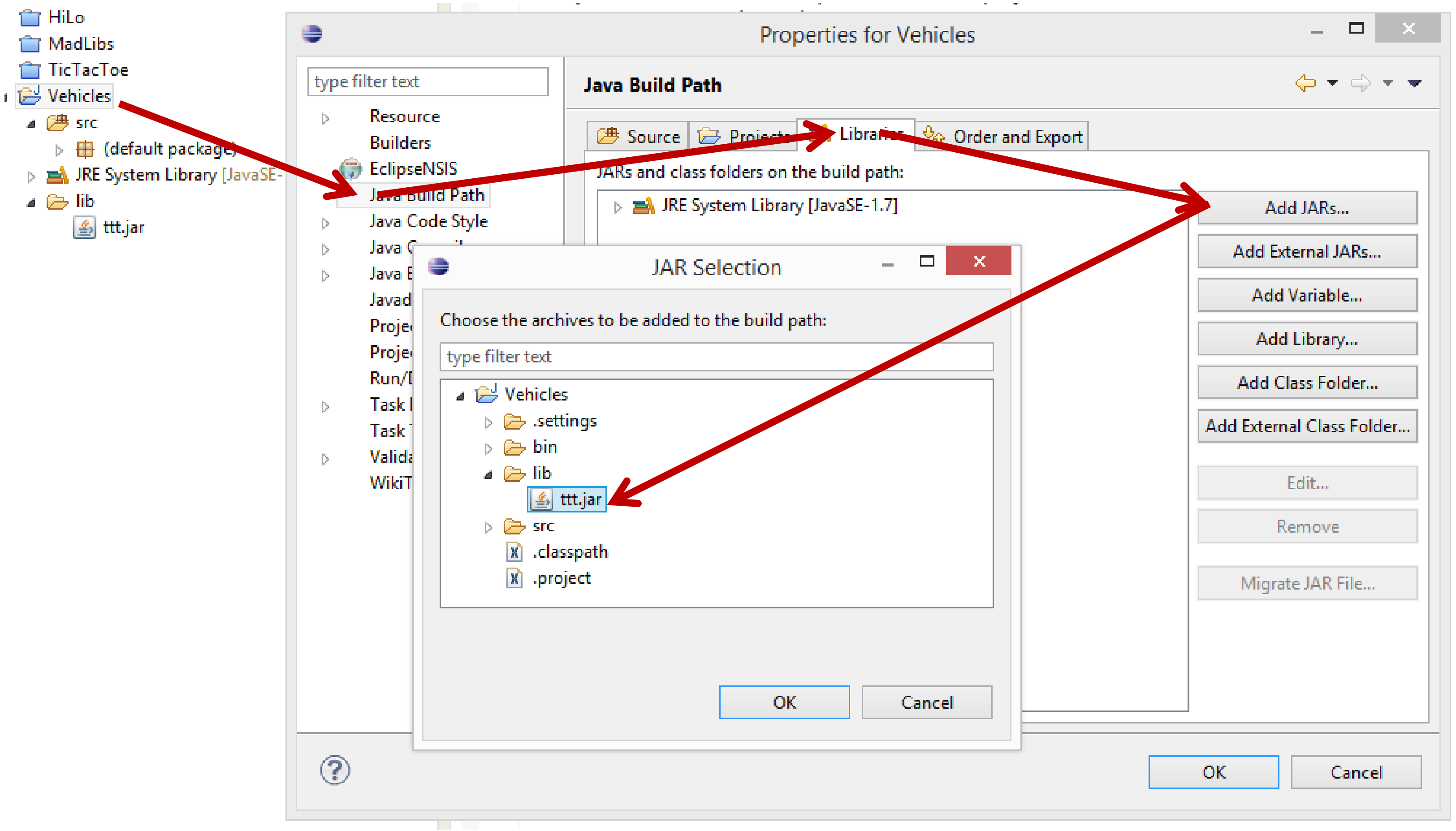
[✓ Copy files]

Link to files

☑ Create link locations relative to: PROJECT_LOC

Eclipse and Jars

- Right click on the project and add JARs to the path



Your Turn

- Google “Apache String Utils”. Download the jar and add it to a project.
- Use Eclipse to “surf” the contents.
- Try using some of the utilities:
  - isNumeric
  - Repeat

The Apache Software Foundation
Community-led development since 1999.