Object Oriented Programming 1

Week 8 – Other OOP Skills

Dr. Thiago Braga Rodrigues

tbrodrigues@ait.ie

Connect & Discover

Athlone Institute of Technology
2021

Chapter 10

Other object-oriented programming skills

Objectives

Applied

- Add two or more classes to a package and make the classes in that package available to other classes.
- Create a JAR file that contains a library of one or more packages and make that library available to other applications.
- Add javadoc comments to the classes in one or more packages and generate the documentation for those packages.
- Use your web browser to view the documentation you added to a package.
- Code more than one class per file. When necessary, use nested classes.
- Declare and use an enumeration.

Objectives (cont.)

- Enhance an enumeration by adding methods that override the methods of the Java and Enum classes. Use methods of the enumeration constants when necessary.
- Use a static import to import the constants of an enumeration or the static fields and methods of a class.

Knowledge

- List two reasons that you might store classes in a package.
- Describe how to create a directory structure for a package.
- Describe how to make one or more packages available to other applications.
- Explain why you might add javadoc comments to the packages you create.

Objectives (cont.)

- Explain the purpose of using HTML and javadoc tags within a javadoc comment.
- Explain when you might code two or more classes in the same file and describe the advantage and disadvantage of doing that.
- Describe the difference between an inner class and a static inner class in terms of how they're related to the outer class.
- Explain what a local class is.
- Explain what an enumeration is and how you use one.
- Explain what static imports are and how you use them.

The directories and files for an application that uses packages

```
ch10_LineItem\src
    murach
        business
            LineItem.java
            Product.java
        database
            ProductDB.java
        lineitem
            LineItemApp.java
        presentation
            Validator.java
```

The LineItem class

```
package murach.business;

import java.text.NumberFormat;

public class LineItem {...}
```

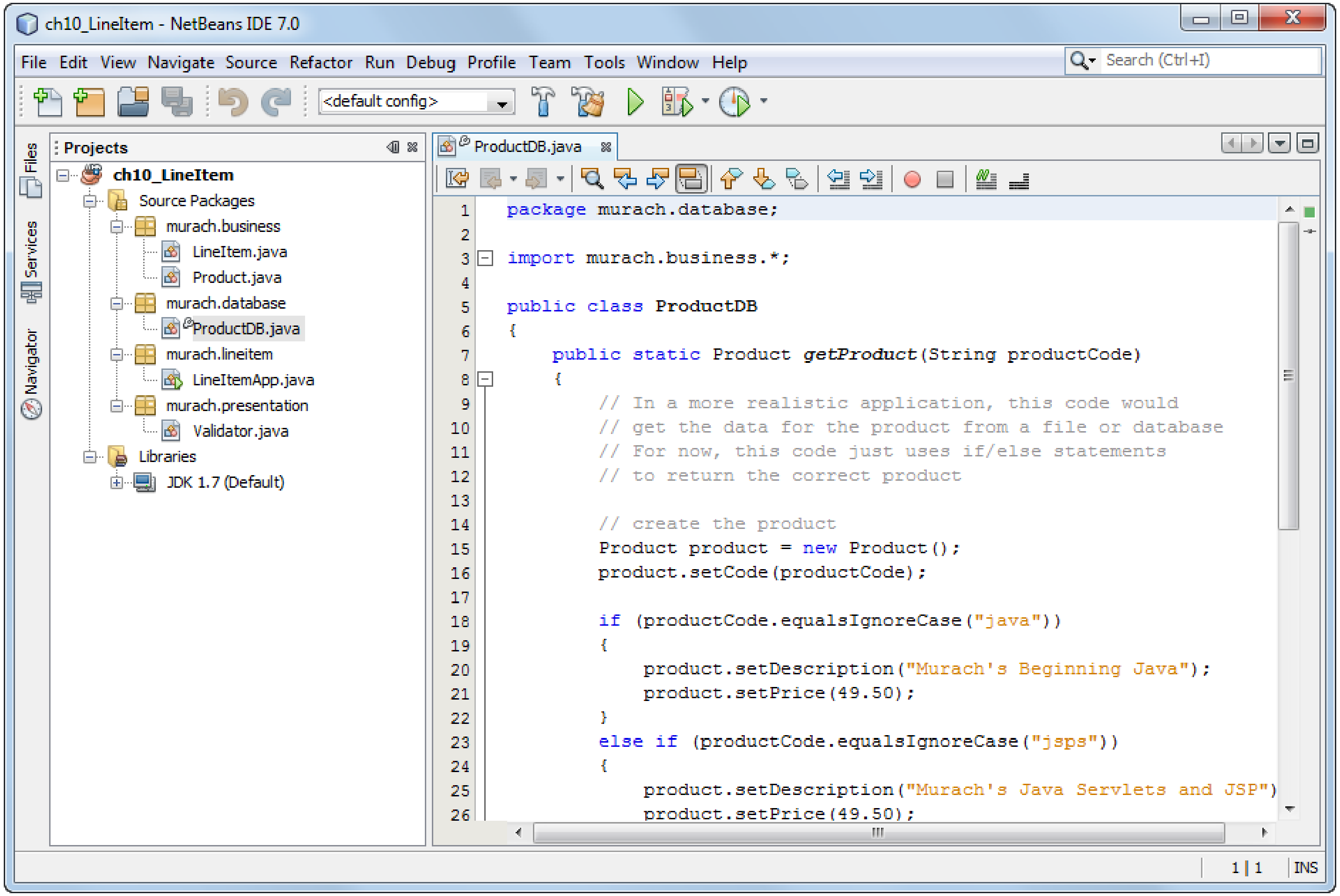
The Product class

```
package murach.business;

import java.text.NumberFormat;

public class Product {...}
```

The ProductDB class

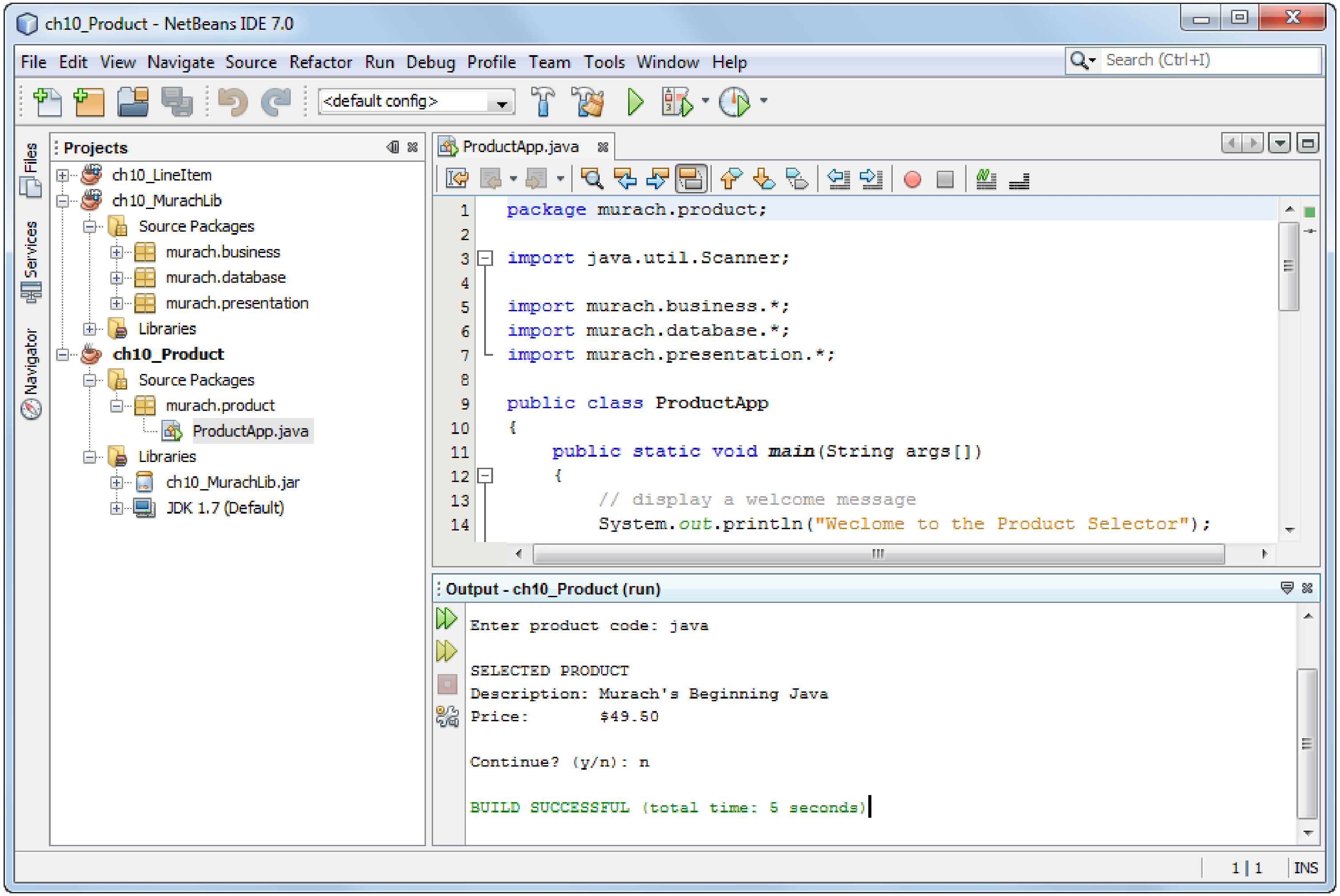
```
package murach.database;

import murach.business.*;

public class ProductDB {...}
```

A NetBeans project that contains multiple packages



```
package murach.database;

import murach.business.*;

public class ProductDB
{
    public static Product getProduct(String productCode)
    {
        // In a more realistic application, this code would
        // get the data for the product from a file or database
        // For now, this code just uses if/else statements
        // to return the correct product

        // create the product
        Product product = new Product();
        product.setCode(productCode);

        if (productCode.equalsIgnoreCase("java"))
        {
            product.setDescription("Murach's Beginning Java");
            product.setPrice(49.50);
        }
        else if (productCode.equalsIgnoreCase("jsps"))
        {
            product.setDescription("Murach's Java Servlets and JSP")
            product.setPrice(49.50);
```

A NetBeans project that uses a library



```
package murach.product;

import java.util.Scanner;

import murach.business.*;
import murach.database.*;
import murach.presentation.*;

public class ProductApp
{
    public static void main(String args[])
    {
        // display a welcome message
        System.out.println("Weclome to the Product Selector");
```

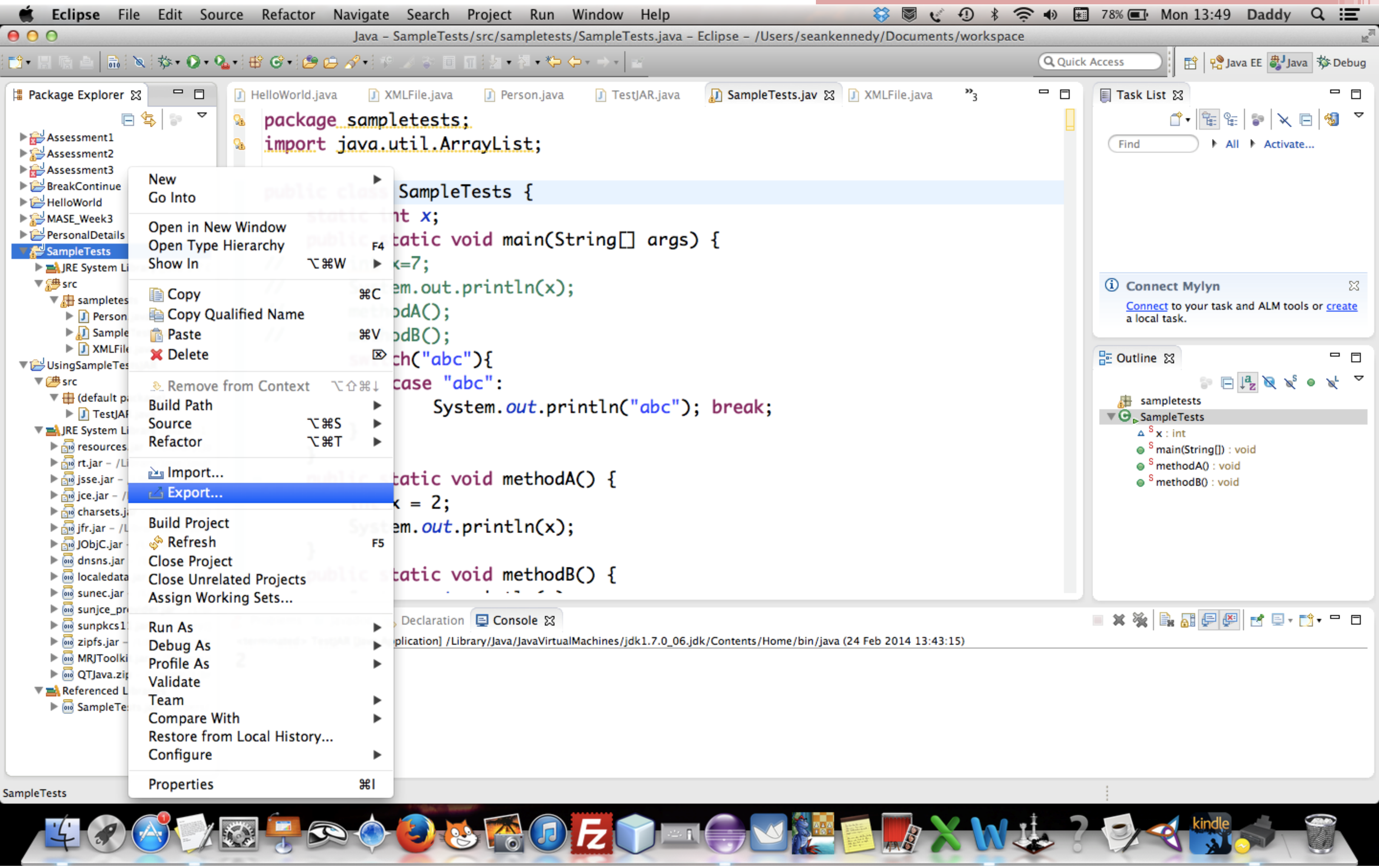
```
Enter product code: java

SELECTED PRODUCT
Description: Murach's Beginning Java
Price:       $49.50

Continue? (y/n): n

BUILD SUCCESSFUL (total time: 5 seconds)
```

How to create a library

1. Create a project that contains just the packages and classes that you want to include in the library.
2. Right-click on the project and select the Build command to compile the project. Then, NetBeans automatically creates a JAR file for the project and it stores it in the dist subdirectory for the project.

How to use a library

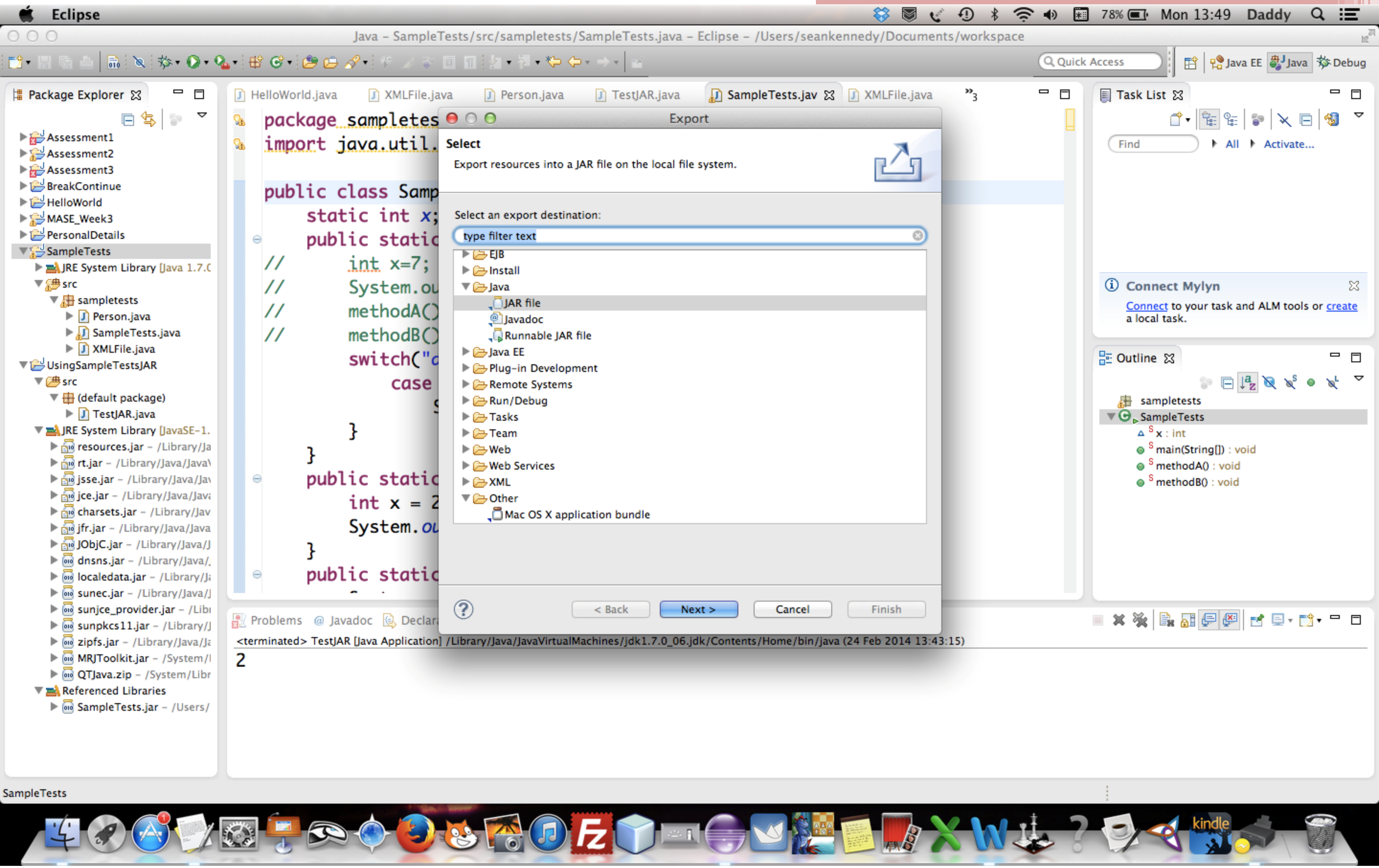
3. Create or open the project that will use the library.
4. Right-click on the Libraries directory and select the "Add JAR/Folder" command. Then, use the resulting dialog box to select the JAR file for the library.
5. Code the import statements for the packages and classes in the library that you want to use. Then, you can use the classes stored in those packages.



Java – SampleTests/src/sampletests/SampleTests.java – Eclipse – /Users/seankennedy/Documents/workspace

```
package sampletests;
import java.util.ArrayList;

public class SampleTests {
    static int x;
    public static void main(String[] args) {
//      int x=7;
//      System.out.println(x);
        methodA();
//      methodB();
        switch("abc"){
        case "abc":
            System.out.println("abc"); break;
        }
    }

    public static void methodA() {
        int x = 2;
        System.out.println(x);
    }

    public static void methodB() {
```

```
<terminated> TestJAR [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
2
```



Java - SampleTests/src/sampletests/SampleTests.java - Eclipse - /Users/seankennedy/Documents/workspace

```
package sampletes
import java.util.

public class Samp
    static int x;
    public static
//      int x=7;
//      System.ou
//      methodA()
//      methodB()
        switch("c
            case
                 S
        }
    }
    public static
        int x = 2
        System.ou
    }
    public static
```

Export

Select

Export resources into a JAR file on the local file system.

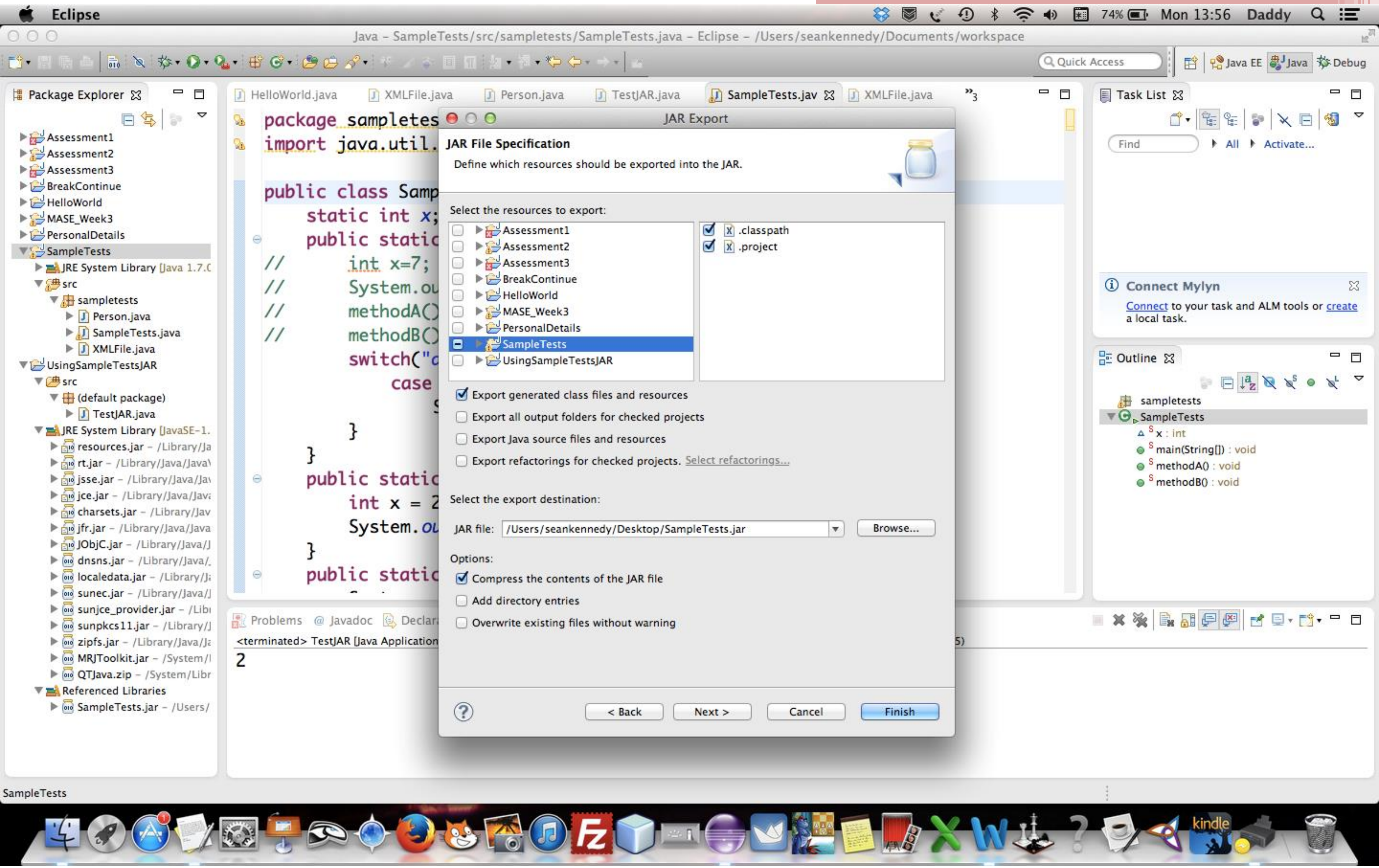
Select an export destination:

type filter text

```
<terminated> TestJAR [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
2
```

Connect Mylyn

Connect to your task and ALM tools or create a local task.



Java - SampleTests/src/sampletests/SampleTests.java - Eclipse - /Users/seankennedy/Documents/workspace

JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- □ Assessment1
- □ Assessment2
- □ Assessment3
- □ BreakContinue
- □ HelloWorld
- □ MASE_Week3
- □ PersonalDetails
- ⊟ SampleTests
- □ UsingSampleTestsJAR

- ☑ .classpath
- ☑ .project

- ☑ Export generated class files and resources
- □ Export all output folders for checked projects
- □ Export Java source files and resources
- □ Export refactorings for checked projects. Select refactorings...

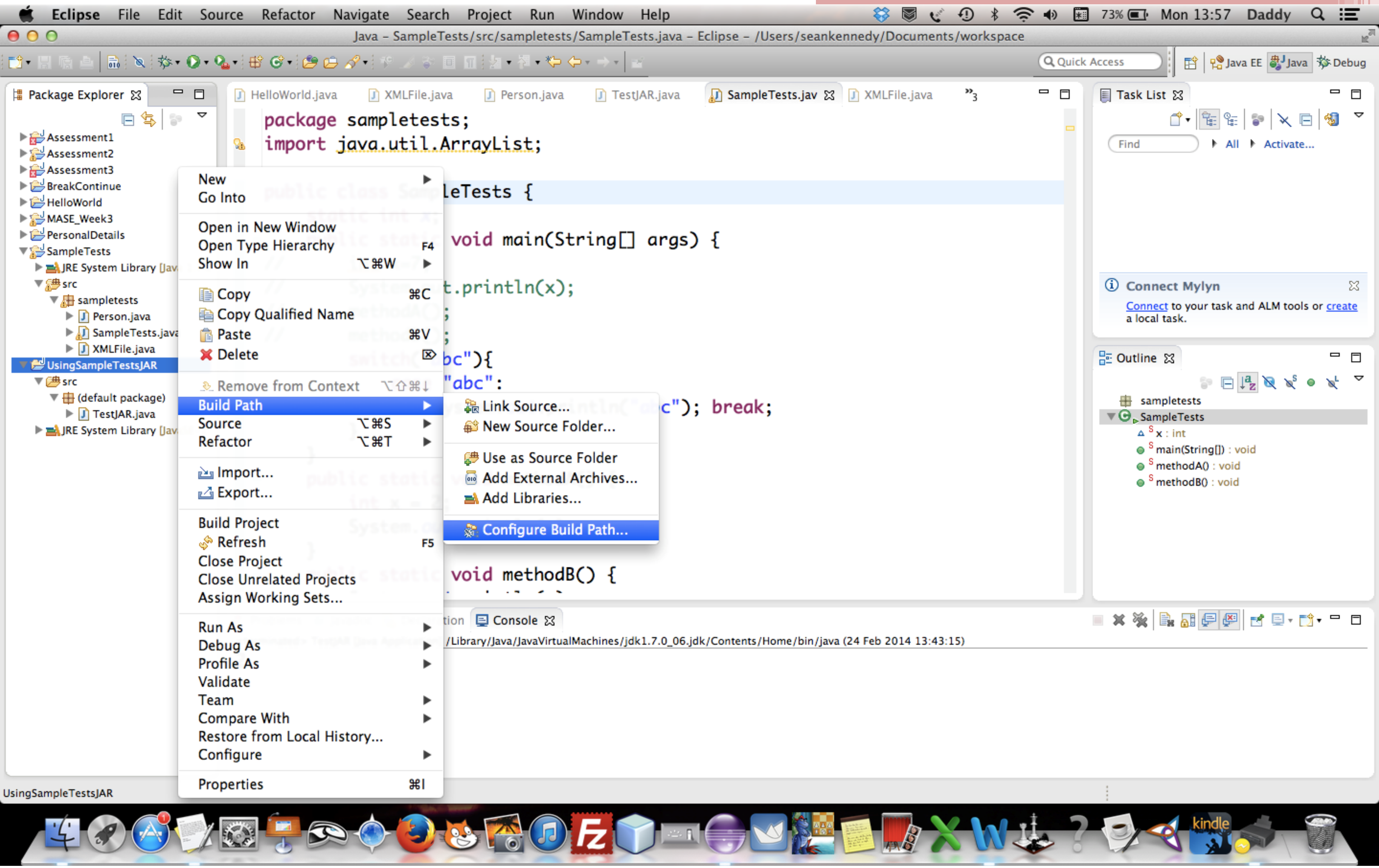
Select the export destination:

JAR file: /Users/seankennedy/Desktop/SampleTests.jar

Options:

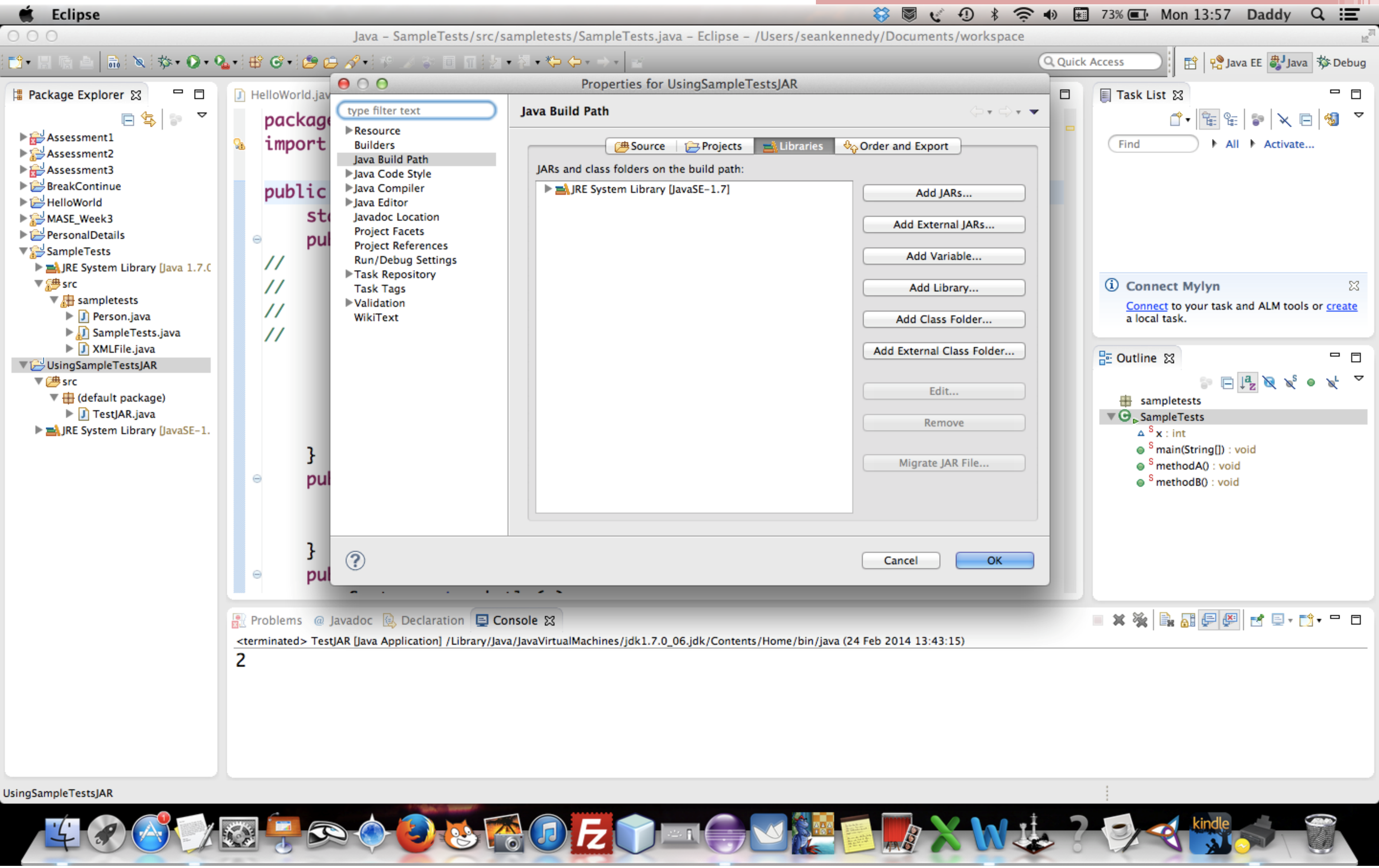
- ☑ Compress the contents of the JAR file
- □ Add directory entries
- □ Overwrite existing files without warning

```
<terminated> TestJAR [Java Application
2
```



```
package sampletests;
import java.util.ArrayList;

leTests {

void main(String[] args) {

t.println(x);
;
;
bc"){
"abc":
c"); break;

void methodB() {
```

```
/Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
```



Java - SampleTests/src/sampletests/SampleTests.java - Eclipse - /Users/seankennedy/Documents/workspace

Properties for UsingSampleTestsJAR

Java Build Path

JARs and class folders on the build path:

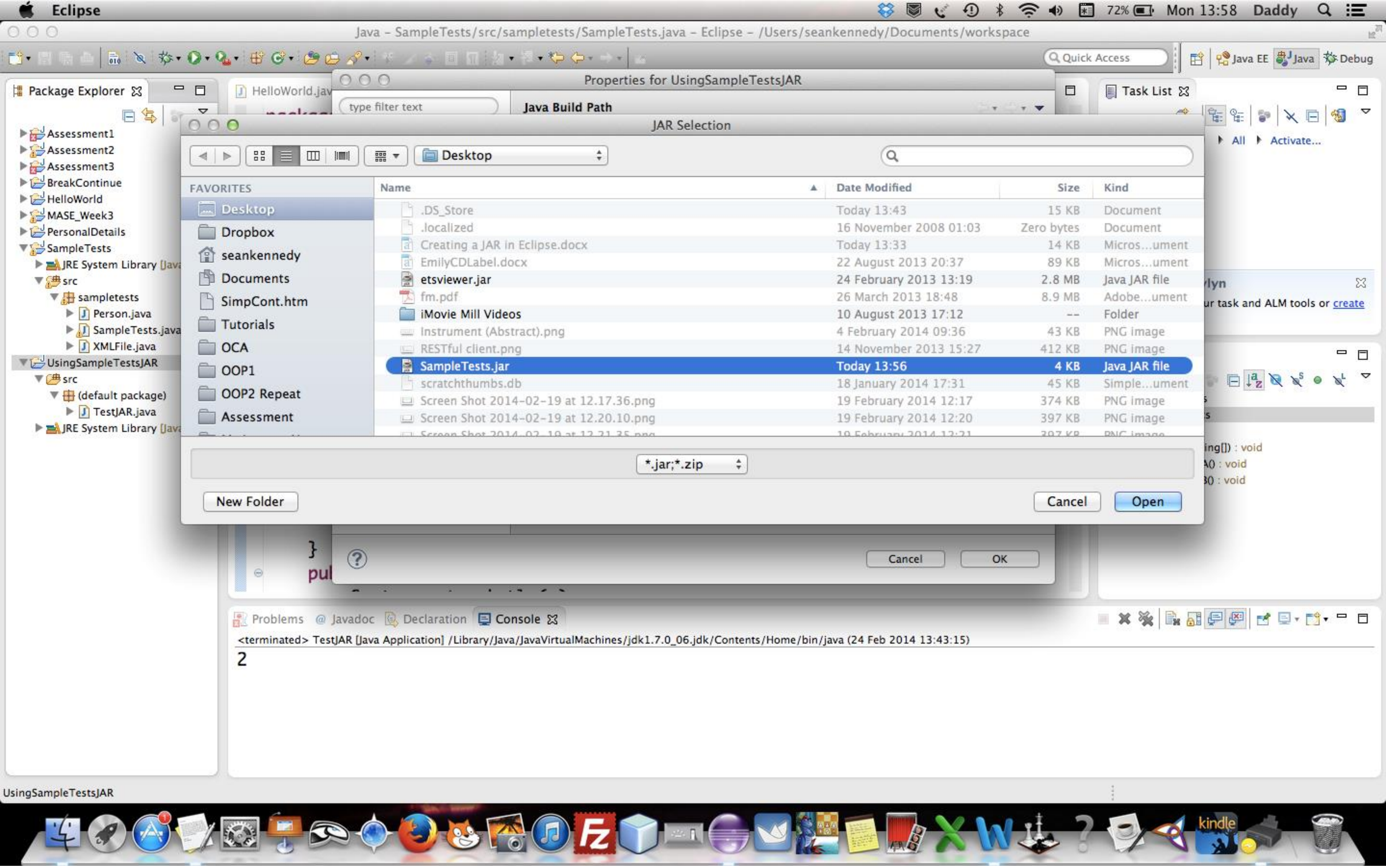
- JRE System Library [JavaSE-1.7]

Connect Mylyn

Connect to your task and ALM tools or create a local task.

```
<terminated> TestJAR [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
2
```

UsingSampleTestsJAR



Java - SampleTests/src/sampletests/SampleTests.java - Eclipse - /Users/seankennedy/Documents/workspace

Properties for UsingSampleTestsJAR

Java Build Path

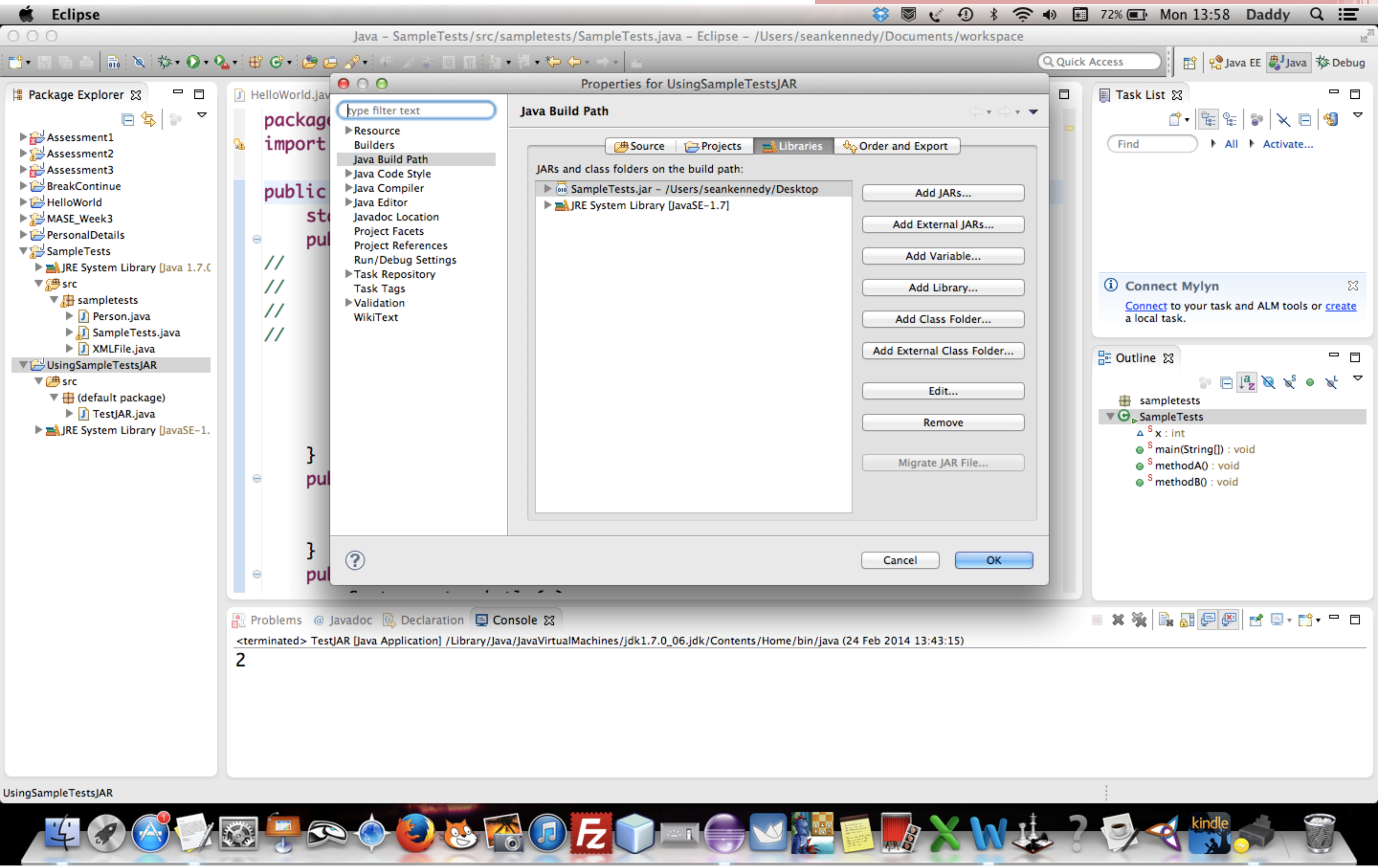
JAR Selection

Desktop

Name	Date Modified	Size	Kind
.DS_Store	Today 13:43	15 KB	Document
.localized	16 November 2008 01:03	Zero bytes	Document
Creating a JAR in Eclipse.docx	Today 13:33	14 KB	Micros...ument
EmilyCDLabel.docx	22 August 2013 20:37	89 KB	Micros...ument
etsviewer.jar	24 February 2013 13:19	2.8 MB	Java JAR file
fm.pdf	26 March 2013 18:48	8.9 MB	Adobe...ument
iMovie Mill Videos	10 August 2013 17:12	--	Folder
Instrument (Abstract).png	4 February 2014 09:36	43 KB	PNG image
RESTful client.png	14 November 2013 15:27	412 KB	PNG image
SampleTests.jar	Today 13:56	4 KB	Java JAR file
scratchthumbs.db	18 January 2014 17:31	45 KB	Simple...ument
Screen Shot 2014-02-19 at 12.17.36.png	19 February 2014 12:17	374 KB	PNG image
Screen Shot 2014-02-19 at 12.20.10.png	19 February 2014 12:20	397 KB	PNG image

.jar;.zip

```
<terminated> TestJAR [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
2
```



Java – SampleTests/src/sampletests/SampleTests.java – Eclipse – /Users/seankennedy/Documents/workspace

Properties for UsingSampleTestsJAR

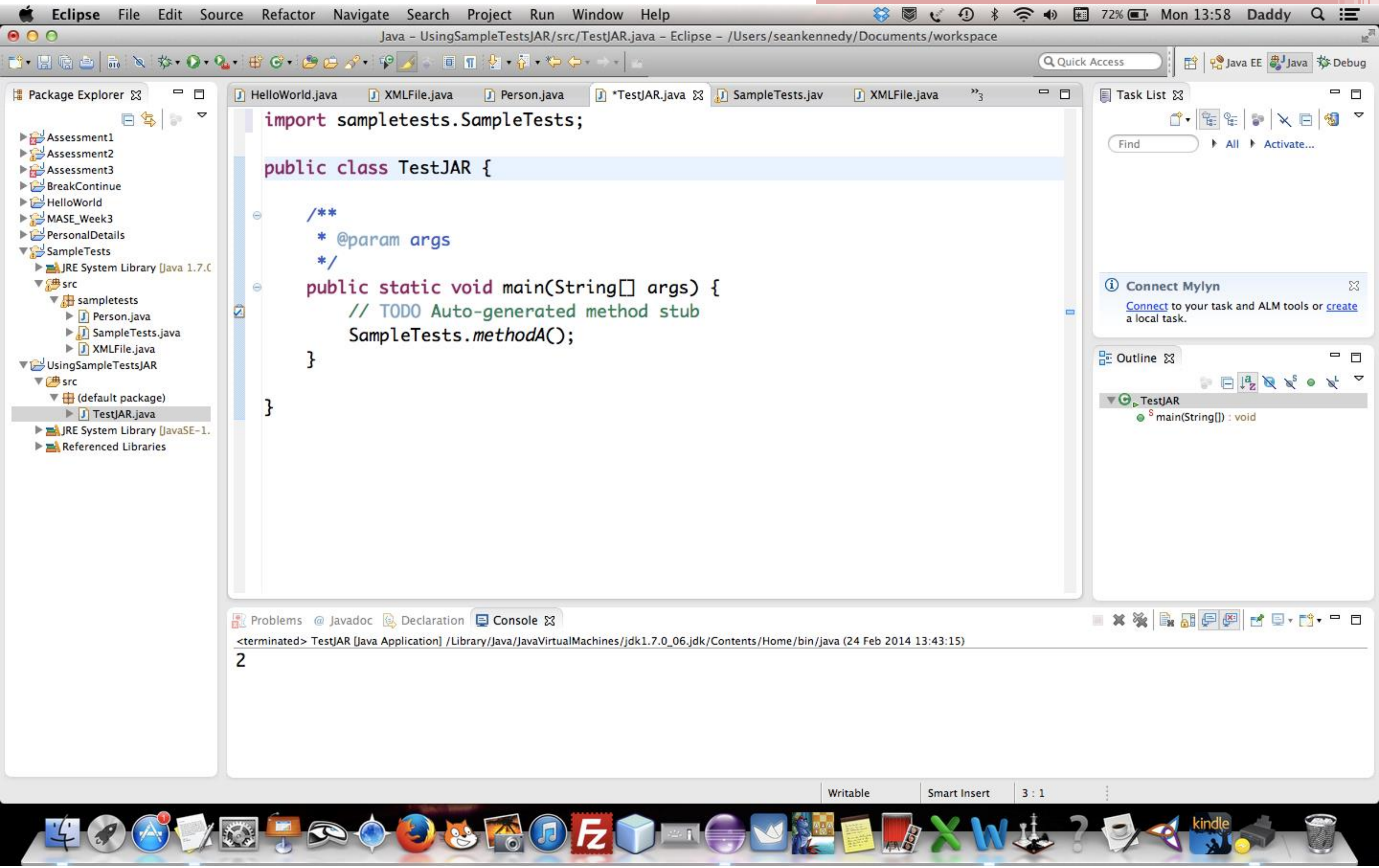
Java Build Path

Source | Projects | **Libraries** | Order and Export

JARs and class folders on the build path:

- SampleTests.jar – /Users/seankennedy/Desktop
- JRE System Library [JavaSE-1.7]

```
<terminated> TestJAR [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
2
```



```java
import sampletests.SampleTests;

public class TestJAR {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SampleTests.methodA();
    }

}
```

```
<terminated> TestJAR [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/bin/java (24 Feb 2014 13:43:15)
2
```

The Product class with javadoc comments

```
package murach.business;

import java.text.NumberFormat;

/**********************************************************
 * The Product class represents a product and is used by
 * the LineItem and ProductDB classes.
 **********************************************************/
public class Product
{
    private String code;
    private String description;
    private double price;
```

Check for Javadoc!!!!!

https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

The Product class with javadoc comments (cont.)

```
/*******************************************************
 * Creates a new Product with default values.
 ******************************************************/
public Product()
{
    code = "";
    description = "";
    price = 0;
}

/*******************************************************
 * Sets the product code to the specified String.
 ******************************************************/
public void setCode(String code)
{
    this.code = code;
}
```

The Product class with javadoc comments (cont.)

```
    /*******************************************************
     * Returns a String that represents the product code.
     ******************************************************/
    public String getCode()
    {
        return code;
    }
    .
    .
    .
}
```

Common HTML tag used to format javadoc comments

- `<code></code>`

Common javadoc tags

- `@author`
- `@version`
- `@param`
- `@return`

The Product class with comments that use HTML and javadoc tags

```
package murach.business;

/********************************************************
 * The <code>Product</code> class represents a product
 * and is used by the <code>LineItem</code> and
 * <code>ProductDB</code> classes.
 * @author Joel Murach
 * @version 1.0.0
 *******************************************************/
public class Product {
    private String code;
    private String description;
    private double price;
```

The Product class with comments that use HTML and javadoc tags (cont.)

```
/***************************************************
 * Creates a <code>Product</code> with default
 * values.
 ***************************************************/
public Product(){
    code = "";
    description = "";
    price = 0;
}

/***************************************************
 * Sets the product code to the specified
 * <code>String</code>.
 * @param code A <code>String</code> for the product
 * code.
 ***************************************************/
public void setCode(String code){
    this.code = code;
}
```

The Product class with comments that use HTML and javadoc tags (cont.)

```
    /***************************************************
     * Returns a <code>String</code> that represents the
     * product code.
     * @return A <code>String</code> for the product
     * code.
     ***************************************************/
    public String getCode(){
        return code;
    }
    .
    .
    .
}
```

Two classes declared within a file named LineItem.java

```
import java.text.NumberFormat;

public class LineItem
{
    private Product product;
    private int quantity;
    private double total;
    .
    .
}

class Product
{
    // body of Product class
}
```

The generated class files

```
LineItem.class
Product.class
```

Two classes nested within another class

```
public class OuterClassName{
    // can contain instance variables and methods
    // can contain static variables and methods
    class InnerClassName{
        // can contain instance variables and methods
        // can't contain static variables or methods
        // can access all variables and methods of
        // OuterClass
    }
    static class StaticInnerClassName{
        // can contain instance variables and methods
        // can contain static variables and methods
        // can access static variables and methods of
        // OuterClass
        // can't access instance variables or methods of
        // OuterClass
    }
}
```

The class files generated for the nested classes

```
OuterClassName.class
OuterClassName$InnerClassName.class
OuterClassName$StaticInnerClassName.class
```

A class nested within a method

```
public class ClassName{
   // code for the outer class
    public void methodName(){
        class InnerClassName{
            // code for the inner class
        }
        // code for the method
    }
}
```

The class files generated for this class

```
ClassName.class
ClassName$InnerClassName.class
```

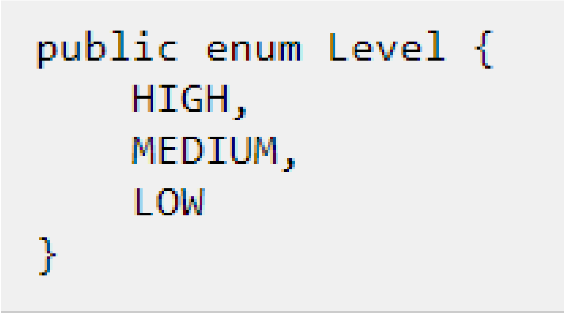
Enums

List of things that never change?

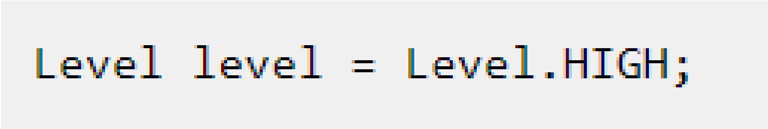
Why?

Can we use other things?

Enum declaration

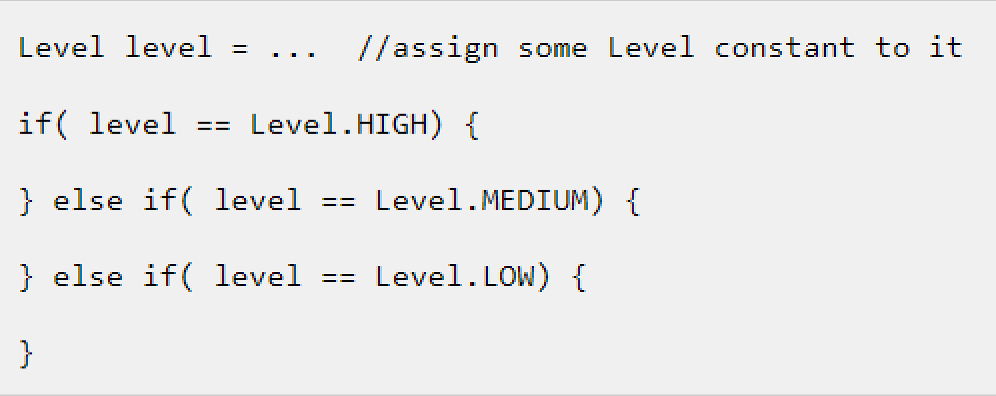


```
public enum Level {
    HIGH,
    MEDIUM,
    LOW
}
```

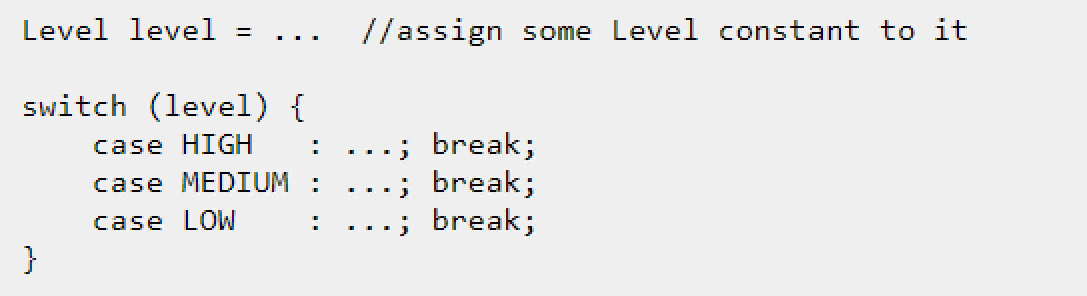


```
Level level = Level.HIGH;
```

Enum if/switch



```
Level level = ...  //assign some Level constant to it

if( level == Level.HIGH) {

} else if( level == Level.MEDIUM) {

} else if( level == Level.LOW) {

}
```



```
Level level = ...  //assign some Level constant to it

switch (level) {
    case HIGH   : ...; break;
    case MEDIUM : ...; break;
    case LOW    : ...; break;
}
```

Useful enum information

http://tutorials.jenkov.com/java/enums.html#:~:text=A%20Java%20Enum%20is%20a,were%20added%20in%20Java%205.

The syntax for declaring an enumeration

```
public enum EnumerationName{
    CONSTANT_NAME1[,
    CONSTANT_NAME2]...
}
```

An enumeration that defines three shipping types

```
public enum ShippingType{
    UPS_NEXT_DAY,
    UPS_SECOND_DAY,
    UPS_GROUND
}
```

A statement that uses the enumeration and one of its constants

```
ShippingType secondDay = ShippingType.UPS_SECOND_DAY;
```

A method that uses the enumeration as a parameter type

```
public static double getShippingAmount(ShippingType st){
    double shippingAmount = 2.99;
    if (st == ShippingType.UPS_NEXT_DAY)
        shippingAmount = 10.99;
    else if (st == ShippingType.UPS_SECOND_DAY)
        shippingAmount = 5.99;
    return shippingAmount;
}
```

A statement that calls the method

```
double shippingAmount =
    getShippingAmount(ShippingType.UPS_SECOND_DAY);
// double shippingAmount2 = getShippingAmount(1);
                            // Wrong type, not allowed
```

Two methods of an enumeration constant

- `name()`
- `ordinal()`

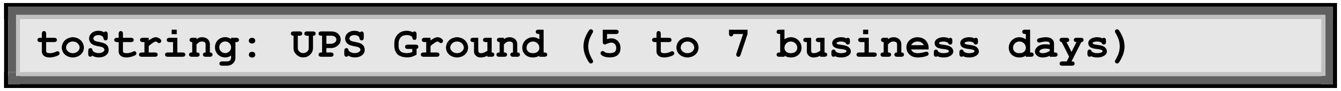
An enumeration that overrides the toString method

```
public enum ShippingType{
    UPS_NEXT_DAY,
    UPS_SECOND_DAY,
    UPS_GROUND;

    @Override
    public String toString()
    {
        String s = "";
        if (this.ordinal() == 0)
            s = "UPS Next Day (1 business day)";
        else if (this.ordinal() == 1)
            s = "UPS Second Day (2 business days)";
        else if (this.ordinal() == 2)
            s = "UPS Ground (5 to 7 business days)";
        return s;
    }
}
```

Code that uses the overridden toString method

```
ShippingType ground = ShippingType.UPS_GROUND;
System.out.println("toString: " + ground.toString() +
    "\n");
```

Resulting output



```
toString: UPS Ground (5 to 7 business days)
```

How to code a static import statement

```
import static murach.business.ShippingType.*;
```

The code above when a static import is used

```
ShippingType ground = UPS_GROUND;
System.out.println(
    "toString: " + ground.toString() + "\n");
```