

COMP 401 PACKAGES

Instructor: Prasan Dewan

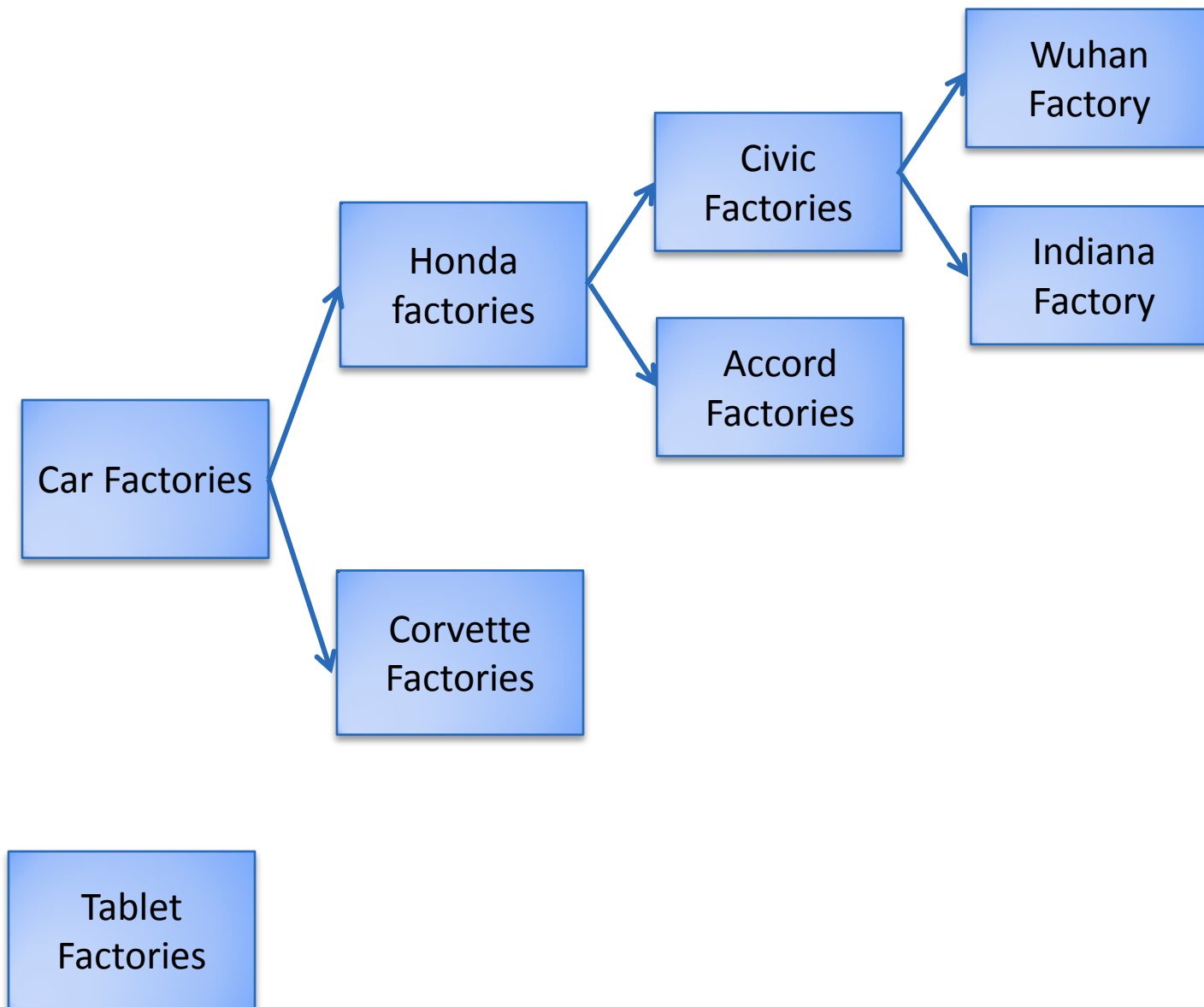


PREREQUISITES

- Objects



GROUPING FACTORIES



GROUPING CLASSES

Package

lectures.functions

- ABMICalculator.java
- ABMICalculatorWithErrors.java
- AMetricConverter.java
- AMonolithicPoundInchBMICalculator.java
- AMyAverageBMICalculator.java
- AMyBMICalculator.java
- APoundInchBMICalculator.java
- ASquareCalculator.java
- FunctionsDriver.java
- SquareCalculatorDriver.java

lectures.generics

lectures.graphics

lectures.inheritance

lectures.inheritance.deep_shallow_copy

lectures.inheritance.equals_polymorphism_overloading

lectures.inheritance.virtual

GROUPING CLASSES

Package

lectures.functions

- ABMCalculator.java
- ABMCalculatorWithErrors.java
- AMetricConverter.java
- AMonolithicPoundInchBMICalculator.java
- AMyAverageBMICalculator.java
- AMyBMICalculator.java
- APoundInchBMICalculator.java
- ASquareCalculator.java
- FunctionsDriver.java
- SquareCalculatorDriver.java

lectures.generics

lectures.graphics

lectures.inheritance

lectures.inheritance.deep_shallow_copy

lectures.inheritance.equals_polymorphism_overloading

lectures.inheritance.virtual

A SIMPLE INSTANTIATED CLASS

No (default)
package

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

```
public class SquareCalculatorDriver
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```



PACKAGES

```
package lectures.functions  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Class in different package must be imported using full name of class (a la full file name)

```
package main;  
import lectures.functions.ASquareCalculator;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```



PACKAGES

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Class in
same
package
need not be
imported

```
package lectures.functions;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```



PACKAGES

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

No package
means package
named default,
hence import
needed

```
import lectures.functions.ASquareCalculator;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```



PACKAGES

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

No package
means package
named default,
hence no import
needed here

```
public class SquareCalculatorDriver
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```

PACKAGES

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

Short name of
class in default
package same
as its full name

```
package main;
public class SquareCalculatorDriver
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```

LONG NAME WITH NO IMPORT

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Can use the full name of class directly

```
package main;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        lectures.functions.ASquareCalculator  
squareCalculator = new lectures.functions.ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

LONG NAME WITH NO IMPORT

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Works but does not tell the reader what is being imported from the package name

```
package main;  
import lectures.functions.*;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

Important role of import is documentation

Programming style violation

WHY IMPORTS/FULL NAME?

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

```
package lectures.safe_functions;  
public class ASquareCalculator  
{  
    public long square(int x)  
    {  
        return x*x;  
    }  
}
```

Twice the
size of ints

```
package main;  
import lectures.functions.ASquareCalculator;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

Disambiguates

AMBIGUOUS IMPORT

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

```
package lectures.safe_functions;  
public class ASquareCalculator  
{  
    public long square(int x)  
    {  
        return x*x;  
    }  
}
```

```
package main;  
import lectures.functions.ASquareCalculator;  
import lectures.safe_functions.ASquareCalculator;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

Ambiguous

UNSUCCESSFUL IMPORT

```
package lectures.functions;
```

```
class ASquareCalculator
```

```
{
```

```
    public int square(int x)
```

```
    {  
        return x*x;  
    }
```

Non public
class usable
only within its
package

```
package lectures.safe_functions;
```

```
public class ASquareCalculator
```

```
{
```

```
    public long square(int x)
```

```
    {
```

```
        return x*x;
```

```
    }
```

```
}
```

```
package main;
```

```
import lectures.functions.ASquareCalculator;
```

```
public class SquareCalculatorDriver
```

```
{
```

```
    public static void main (String[] args) {
```

```
        ASquareCalculator squareCalculator = new
```

```
        ASquareCalculator();
```

```
        System.out.println (squareCalculator.square(5));
```

```
    }
```

```
}
```



WHY PACKAGES?

- Can create competing implementations of same class.
 - A la creating files Test.java in different assignment directories/folders
- Groups related classes together
- Can browse/search for related classes
 - A la browsing through all files in an assignment directory/folder.
- Like directories/folders packages can be hierarchical
 - package** recitations.functions;
 - package** lectures.functions;
- Provides documentation of what unrelated classes are being used

BROWSING JAVA CLASSES

[lectures.annotations](#)
[lectures.arrays](#)
[lectures.assertions](#)
[lectures.collections](#)
[lectures.comments](#)
[lectures.composite_design_pattern](#)
[lectures.demoers](#)
[lectures.exceptions](#)
[lectures.factories](#)
[lectures.functions](#)
[lectures.generics](#)
[lectures.graphics](#)

Package lectures.functions

[ABMICalculator](#)

[AMetricConverter](#)

[ASquareCalculator](#)

```
package lectures.functions;

public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
}
```

BROWSING JAVA CLASSES

Very useful
package

Sun Developer Network (SDN)
APIs Downloads Products Support Training

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform Standard API Specification

This document is the API specification for the Java 2 Platform Standard Edition.

See:
Description

Java 2 Platform Packages	
java.applet	Provides the classes and methods that an applet uses to communicate with the user's computer.
java.awt	Contains all of the classes and methods for the Abstract Window Toolkit (AWT), which provides a simple way to create graphical user interfaces.
java.awt.color	Provides classes for color management.
java.awt.datatransfer	Provides interfaces and classes for data transfer between applications.
java.awt.dnd	Drag and Drop is a direct manipulation interface system that allows two entities logically associated with each other to be moved between them.
java.awt.event	Provides interfaces and classes for event handling by AWT components.
java.awt.font	Provides classes and methods for font rendering.
java.awt.geom	Provides the Java 2D class library.

http://java.sun.com/global/mh/suncom/index.html

LANGUAGE VS. LIBRARY

[java.io](#)
[java.lang](#)
[java.lang.annotation](#)
[java.lang.instrument](#)
[java.lang.management](#)
[java.lang.ref](#)
[java.lang.reflect](#)
[java.math](#)

Built-in
classes

→ [java.lang](#)
Interfaces
[Appendable](#)
[CharSequence](#)
[Cloneable](#)
[Comparable](#)
[Iterable](#)
[Readable](#)
[Runnable](#)
[Thread.UncaughtException](#)
Classes
[Boolean](#)
[Byte](#)
[Character](#)
[Character.Subset](#)
[Character.UnicodeBlock](#)
[Class](#)
[ClassLoader](#)
[Compiler](#)
[Double](#)
[Enum](#)
[Float](#)

Do not have
to be
explicitly
imported

Sun Developer Network (SDN)
APIs Downloads Products Support Training Participate

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform Standard Edition 5.0 API Specification

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See:
Description

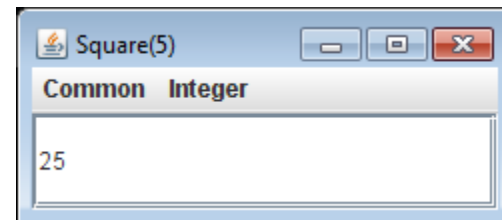
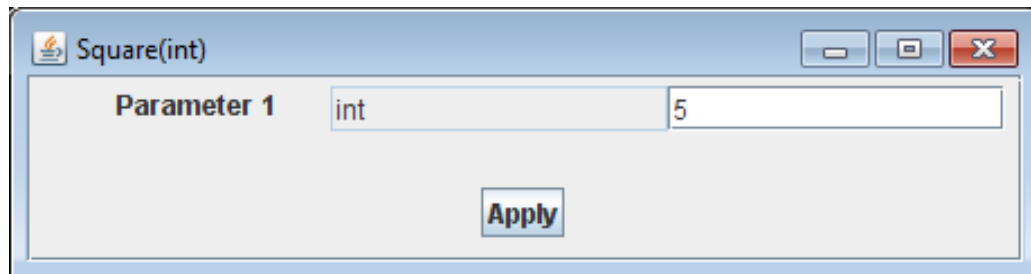
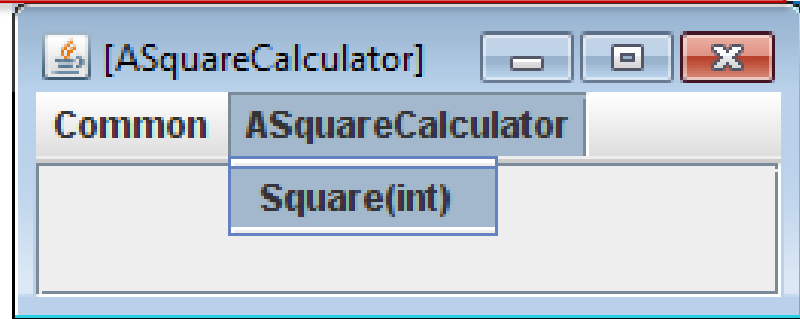
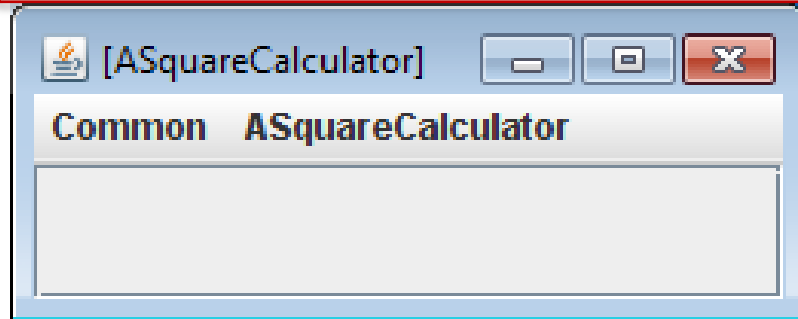
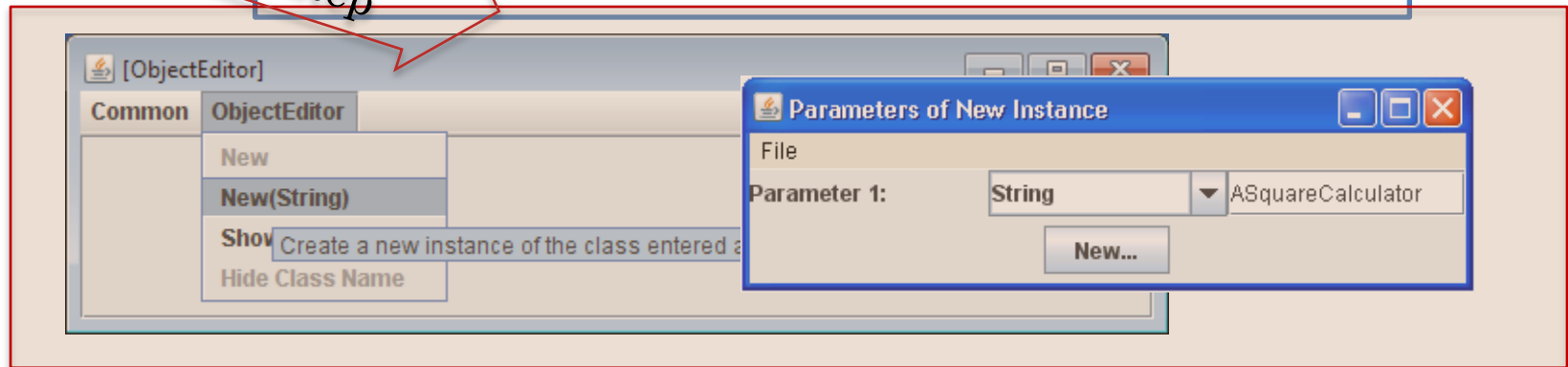
Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes necessary for applet users to communicate with its applet.
java.awt	Contains all of the classes for creating user graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transfer applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture interface system that provides a mechanism for associating two entities logically associated with presentation.
java.awt.event	Provides interfaces and classes for dealing with AWT components.
java.awt.font	Provides classes and interface relating to font.
java.awt.geom	Provides the Java 2D classes for defining and manipulating geometric shapes.

Done Internet | Protected Mode

STARTING OBJECT EDITOR INTERACTIVELY

Can we tell short circuit this step

javort classpath .;oeall20.jar Comp110ObjectEditor



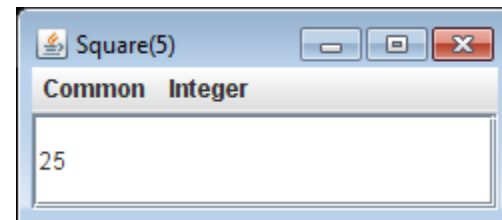
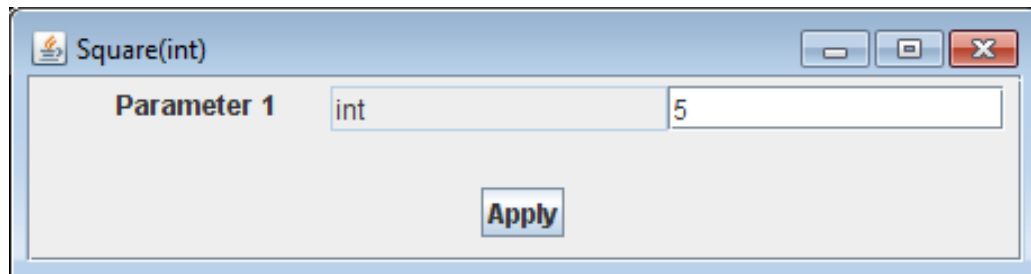
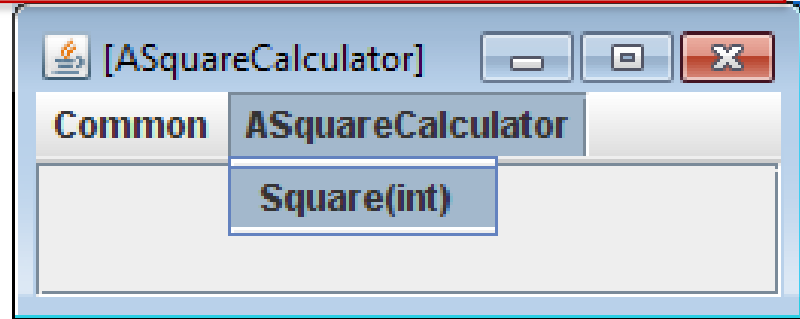
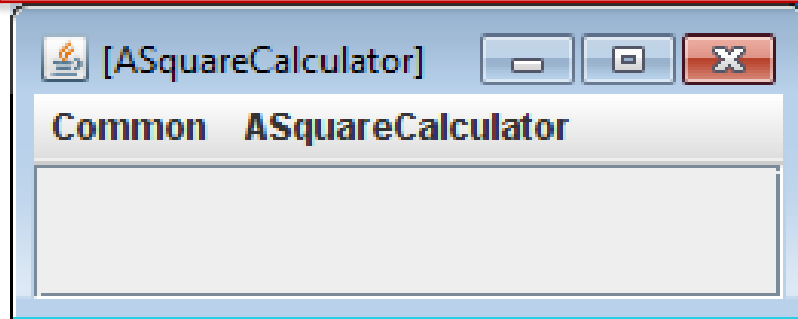
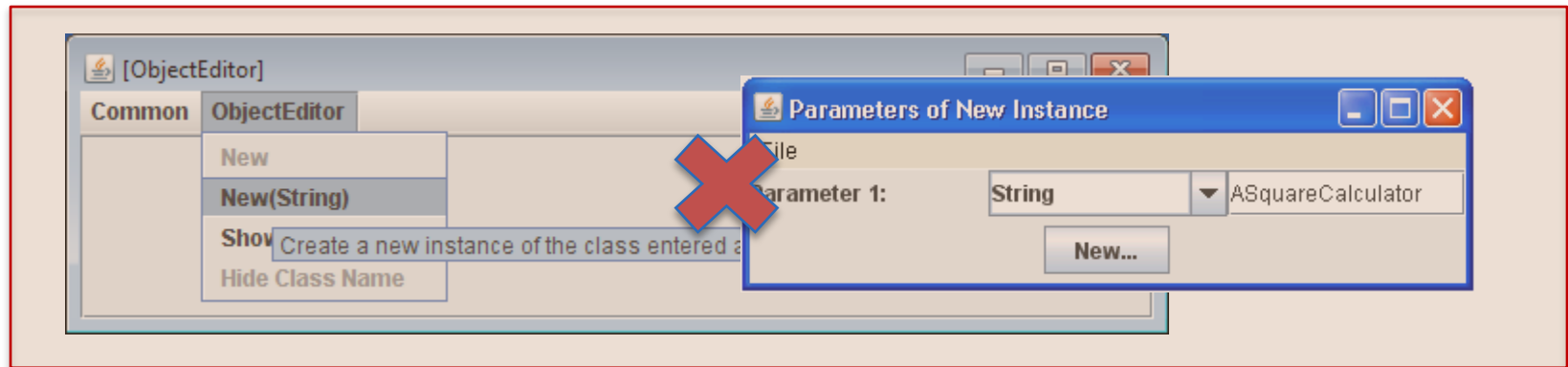
STARTING OBJECTEDITOR PROGRAMMATICALLY

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

ObjectEditor
is predefined
packaged
class

```
package main;  
import lectures.functions.ASquareCalculator;  
import bus.uigen.ObjectEditor;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        ObjectEditor.edit(squareCalculator );  
    }  
}
```

NO NEED TO ENTER CLASS NAME



REMEMBERING PACKAGE NAMES?

```
package math;
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

What if we do not remember the package names?

In Eclipse
CTRL_SHIFT_O

```
package main;

public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
ASquareCalculator();
        ObjectEditor.edit(squareCalculator );
    }
}
```


AUTOMATIC IMPORTS IN ECLIPSE

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

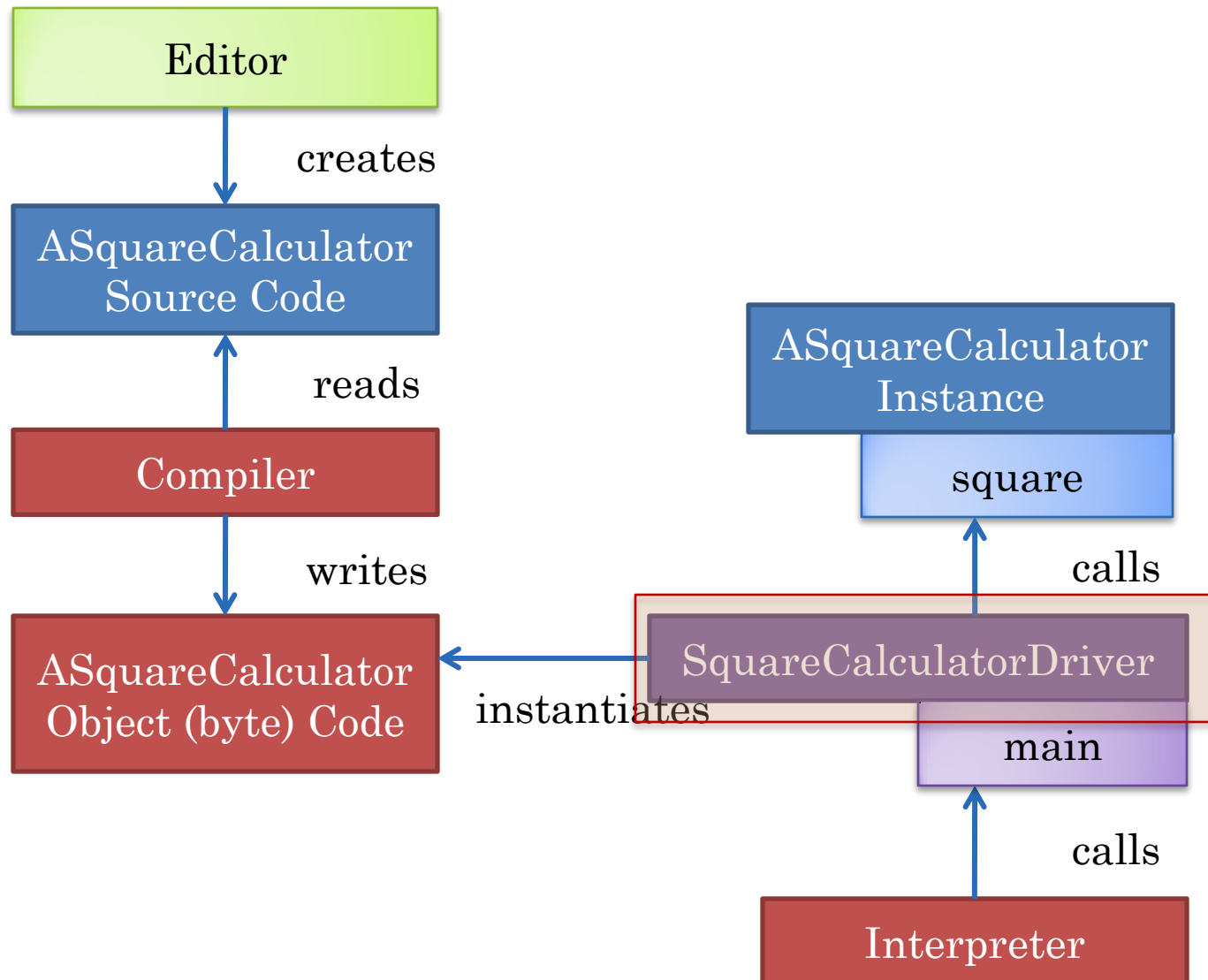
Automatically
added

Gives a dialogue
box in case
multiple classes
in same package

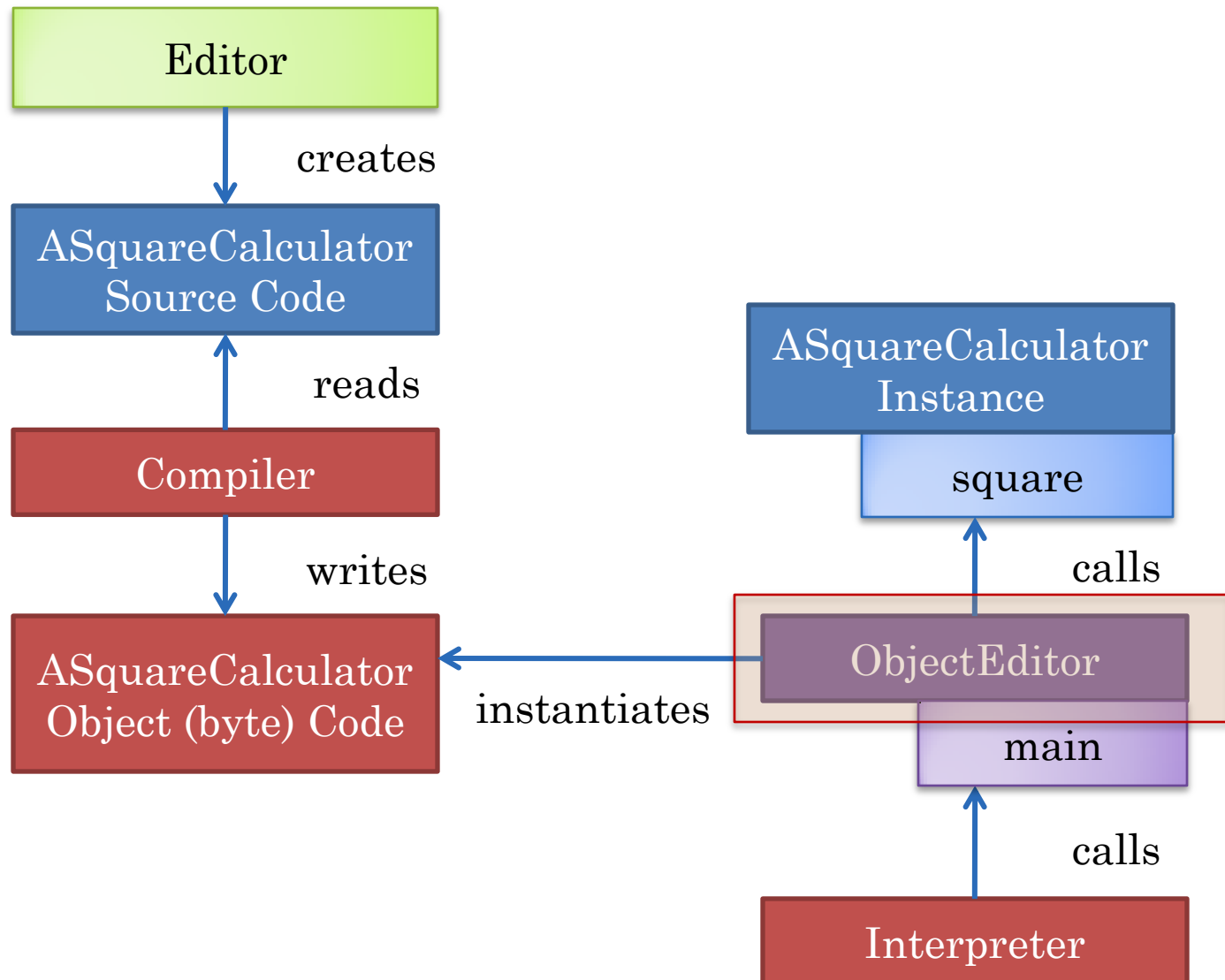
```
package main;  
import math.ASquareCalculator;  
import bus.uigen.ObjectEditor;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
ASquareCalculator();  
        ObjectEditor.edit(squareCalculator );  
    }  
}
```

Package names
will not be given
in class examples

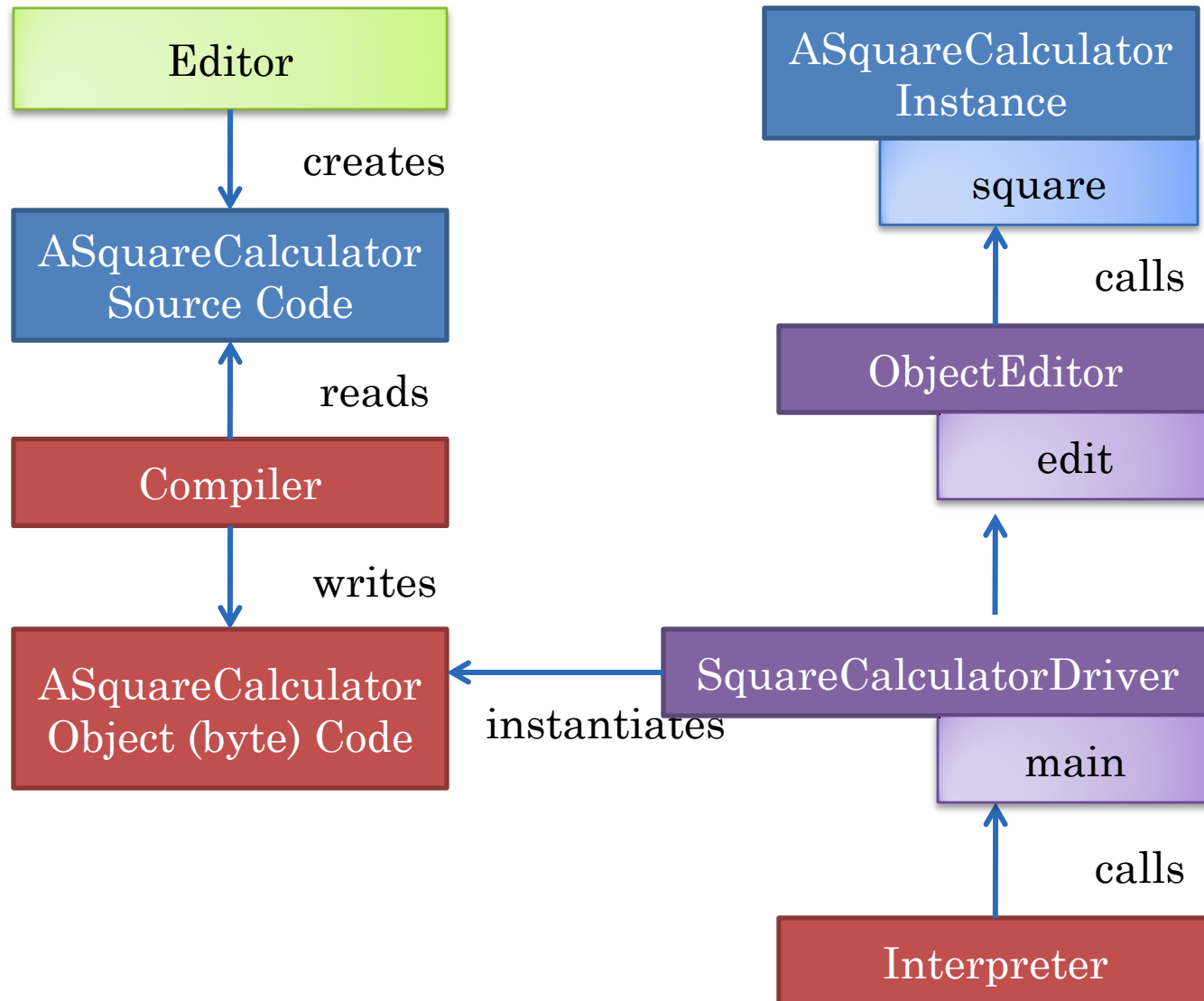
PROGRAMMED MAIN AND METHOD CALLS



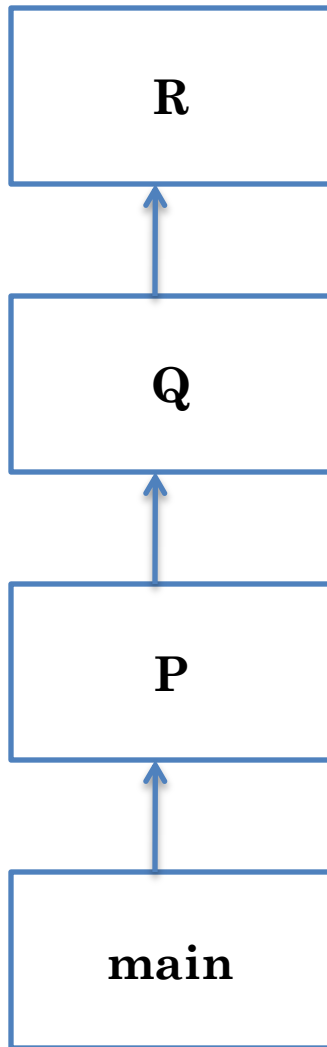
INTERACTIVE MAIN AND METHOD INVOCATION



PROGRAMMED MAIN AND INTERACTIVE METHOD INVOCATION



CALL CHAINS



Main method starts the computation, and can call other methods.

Can put complete program in main method

Like having one big paragraph in an essay

Method decomposition important modularization technique even in conventional programming

In O-O programming multiple classes involved in computation

TWO CLASSES

```
package lectures.functions;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Can we
combine the
two classes?

```
package main;  
import lectures.functions.ASquareCalculator;  
import bus.uigen.ObjectEditor;  
public class SquareCalculatorDriver  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
ASquareCalculator();  
        ObjectEditor.edit(squareCalculator );  
    }  
}
```

COMBINING THE CLASSES FOR TESTING

```
package lectures.functions;
import bus.uigen.ObjectEditor;
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
    public static void main (String[] args) {
        ObjectEditor.edit(new ASquareCalculator());
    }
}
```

Operation
on an instance

Start operations in the factory

Display the
instance

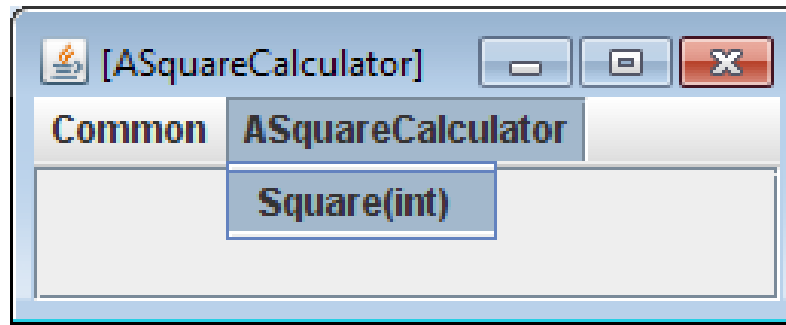
Create a test
instance

```
java -classpath .;oeall20.jar ASquareCalculator
```

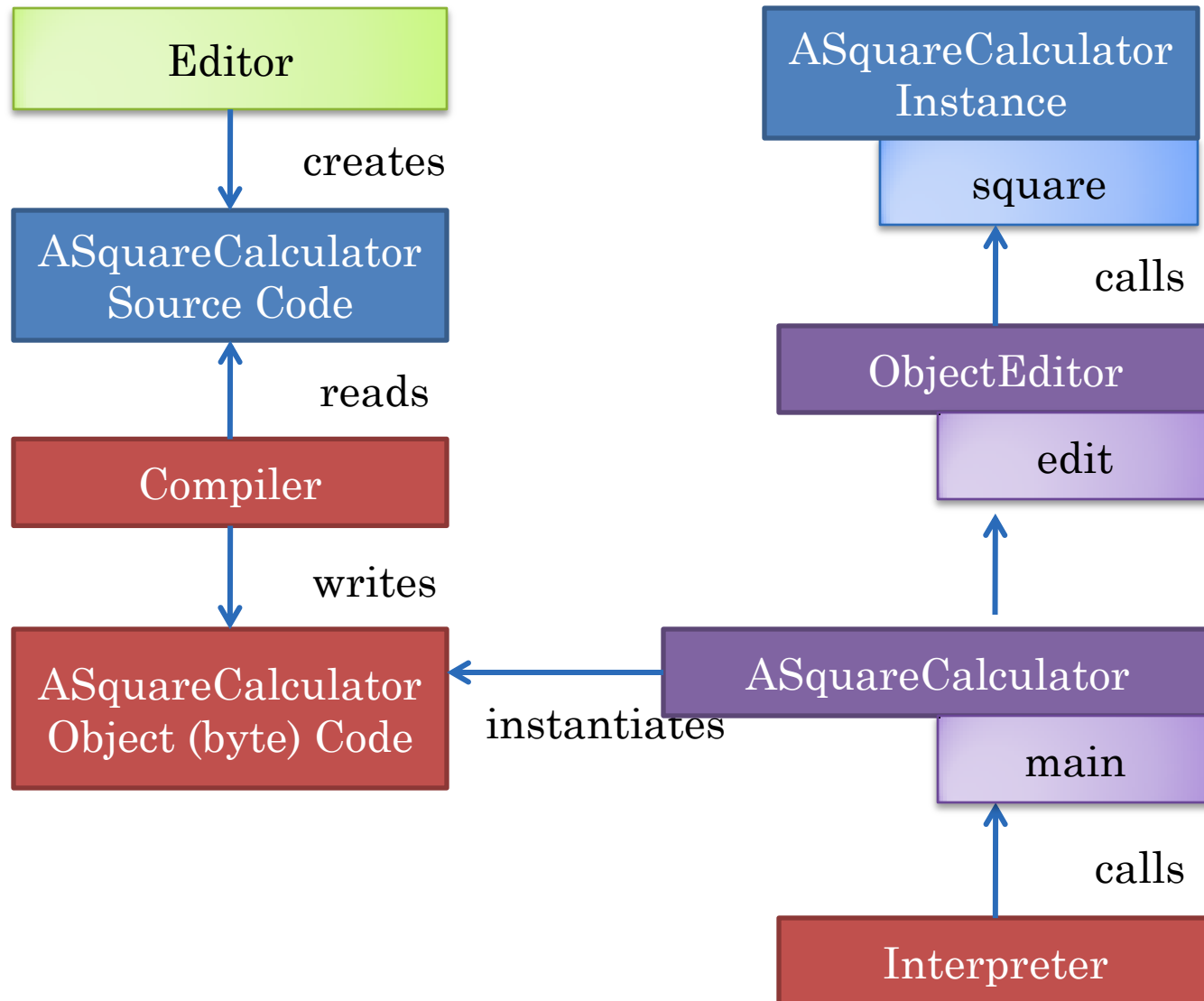


ASquareCalculator.java

Convenient
but possibly
confusing!



PROGRAMMED MAIN AND INTERACTIVE METHOD INVOCATION



EXTRA SLIDES

No PACKAGE

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

```
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```

PACKAGES

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Class in different package must be imported using full name of class (a la full file name)

```
package main;  
import math.ASquareCalculator;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

PACKAGES

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Class in
same
package
need not be
imported

```
package math;  
  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

PACKAGES

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

No package
means package
named default,
hence import
needed

```
import math.ASquareCalculator;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

PACKAGES

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

No package
means package
named default,
hence no import
needed here

```
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```

PACKAGES

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

Short name of
class in default
package same
as its full name

```
package main;
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```

LONG NAME WITH NO IMPORT

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Can use the full name of class directly

```
package main;  
  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        math.ASquareCalculator squareCalculator = new  
        math.ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```


LONG NAME WITH NO IMPORT

```
package math;
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

Works but does not tell the reader what is being imported from the package name

Important role of import is documentation

Programming style violation

```
package main;
import math.*;
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
```

WHY IMPORTS/FULL NAME?

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

```
package safemath;  
public class ASquareCalculator  
{  
    public long square(int x)  
    {  
        return x*x;  
    }  
}
```

Twice the
size of ints

```
package main;  
import math.ASquareCalculator;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

Disambiguates

AMBIGUOUS IMPORT

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

```
package safemath;  
public class ASquareCalculator  
{  
    public long square(int x)  
    {  
        return x*x;  
    }  
}
```

```
package main;  
import math.ASquareCalculator;  
import safemath.ASquareCalculator;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```



Ambiguous

UNSUCCESSFUL IMPORT

```
package math;  
class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Non public
class usable
only within its
package

```
package safemath;  
public class ASquareCalculator  
{  
    public long square(int x)  
    {  
        return x*x;  
    }  
}
```

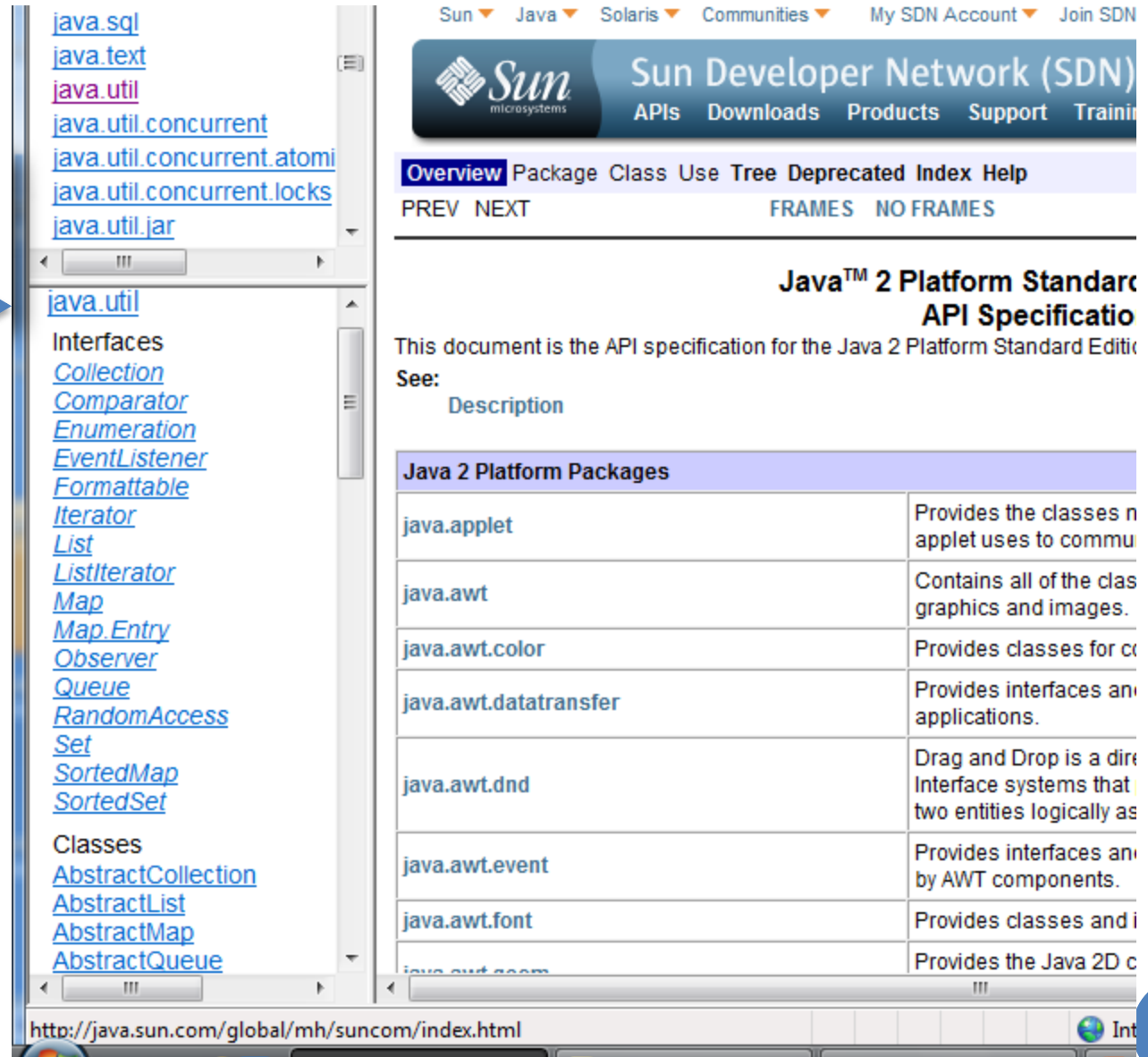
```
package main;  
import math.ASquareCalculator;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```

WHY PACKAGES?

- Can create competing implementations of same class.
 - A la creating files Test.java in different assignment directories/folders
- Groups related classes together
- Can browse/search for related classes
 - A la browsing through all files in an assignment directory/folder.
- Like directories/folders packages can be hierarchical
package math.power;
public class ACubeCalculator {...}
- Provides documentation of what unrelated classes are being used

BROWSING JAVA CLASSES

Very
useful
package



Sun Java Solaris Communities My SDN Account Join SDN

Sun Developer Network (SDN)
APIs Downloads Products Support Training

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform Standard API Specification

This document is the API specification for the Java 2 Platform Standard Edition

See:
Description

Java 2 Platform Packages	
java.applet	Provides the classes and methods that an applet uses to communicate with the host environment.
java.awt	Contains all of the classes and methods for the Abstract Window Toolkit (AWT), which provides a simple, portable, and efficient way to create graphical user interfaces.
java.awt.color	Provides classes for color management.
java.awt.datatransfer	Provides interfaces and classes for data transfer between applications.
java.awt.dnd	Drag and Drop is a direct manipulation interface system that allows two entities logically associated with each other to be moved between them.
java.awt.event	Provides interfaces and classes for event handling by AWT components.
java.awt.font	Provides classes and methods for font rendering.
java.awt.geom	Provides the Java 2D classes for geometric shapes and transformations.

http://java.sun.com/global/mh/suncom/index.html



LANGUAGE VS. LIBRARY

[java.io](#)
[java.lang](#)
[java.lang.annotation](#)
[java.lang.instrument](#)
[java.lang.management](#)
[java.lang.ref](#)
[java.lang.reflect](#)
[java.math](#)

Built-in
classes

→ [java.lang](#)
Interfaces
[Appendable](#)
[CharSequence](#)
[Cloneable](#)
[Comparable](#)
[Iterable](#)
[Readable](#)
[Runnable](#)
[Thread.UncaughtExceptionHandler](#)
Classes
[Boolean](#)
[Byte](#)
[Character](#)
[Character.Subset](#)
[Character.UnicodeBlock](#)
[Class](#)
[ClassLoader](#)
[Compiler](#)
[Double](#)
[Enum](#)
[Float](#)

Do not have
to be
explicitly
imported

Sun Developer Network (SDN)
APIs Downloads Products Support Training Participate

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform Standard Edition 5.0 API Specification

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See:
Description

Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes necessary for an applet to communicate with its applet viewer.
java.awt	Contains all of the classes for creating user graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transfer applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture interface system that provides a mechanism for associating two entities logically associated with presentation.
java.awt.event	Provides interfaces and classes for dealing with AWT components.
java.awt.font	Provides classes and interface relating to font rendering.
java.awt.geom	Provides the Java 2D classes for defining and manipulating geometric shapes.

Done Internet | Protected Mode

CHANGING PARAMETER

```
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

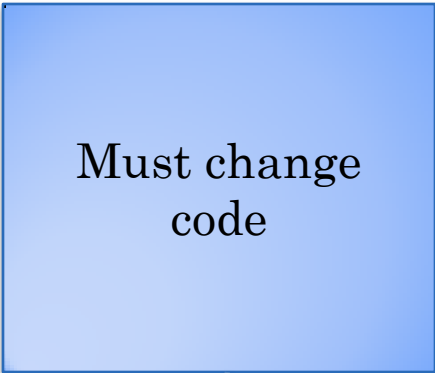
Calculates 5*5

```
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
        ASquareCalculator();  
        System.out.println (squareCalculator.square(5));  
    }  
}
```


CHANGING PARAMETER

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

Must change
code



```
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println (squareCalculator.square(341));
    }
}
```

RERUN PROGRAM

How to not re-run program without writing tedious UI code?

```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

Must re-run program

```
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        System.out.println
        (squareCalculator.square(Integer.parseInt(args[0]));
    }
}
```

OBJECTEDITOR

```
package math;
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

ObjectEditor
is predefined
packaged
class

```
package main;
import math.ASquareCalculator;
import bus.uigen.ObjectEditor;
public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
        ASquareCalculator();
        ObjectEditor.edit(squareCalculator );
    }
}
```

REMEMBERING PACKAGE NAMES?

```
package math;
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

What if we do not remember the package names?

In Eclipse
CTRL_SHIFT_O

```
package main;

public class SquareCalculatorTester
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new
ASquareCalculator();
        ObjectEditor.edit(squareCalculator );
    }
}
```

AUTOMATIC IMPORTS IN ECLIPSE

```
package math;  
public class ASquareCalculator  
{  
    public int square(int x)  
    {  
        return x*x;  
    }  
}
```

Automatically
added

Gives a dialogue
box in case
multiple classes
in same package

```
package main;  
import math.ASquareCalculator;  
import bus.uigen.ObjectEditor;  
public class SquareCalculatorTester  
{  
    public static void main (String[] args) {  
        ASquareCalculator squareCalculator = new  
ASquareCalculator();  
        ObjectEditor.edit(squareCalculator );  
    }  
}
```

Package names
will not be given
in class examples

SUMMARY: JAVA VS. REAL-WORLD

Java	Real-world
Class	Factory
Computer Object	Manufactured Physical Object
Method	Operation
Invoking/Executing a Method	Performing an Operation
Instance of a Class	Manufactured by a Factory
Defining/Declaring a Class	Constructing a Factory
Instantiating a Class	Manufacturing an Object
Class (Static) Method	Operation on a Factory
Main Class Method	Initiates computation
Instance (Non static) method	Operation on an instance of a class
Instance (Non static) method	Grouping of factories by states, country