# Working with Packages

Jim Wilson

@hedgehogjim | blog.jwhh.com | jimw@jwhh.com

# What to Expect in This Module

- What is a package?
- Packages as a namespace
- Importing packages
- Limiting access to package contents
- Distributing packages and archive files

## What Is a Package?

A package is a group of related types

- Create a namespace
- Provide an access boundary
- Act as a unit of distribution

## Declaring Packages

Each source file identifies the associated package

Use package keyword

Package declaration must appear before any type declarations

Applies to all types within that source file

```
package xxxxxx ;

public class Flight {
  // members elided for clarity
}
```

## Packages Create a Namespace

Package name is part of the type name

Convention creates unique package name

Follows reverse domain name structure

```
package com.pluralsight.travel ;

public class Flight {
  // members elided for clarity
}
```

## Packages Create a Namespace

Package name is part of the type name

Convention creates unique package name

Follows reverse domain name structure

Type name is qualified by package name

```
package com.pluralsight.travel ;

public class Flight {
  // members elided for clarity
}
```

```
com.pluralsight.travel.Flight lax175 =
  new com.pluralsight.travel.Flight(175);
```

## Determining a Type's Package

Compiler needs to know each type's package

Explicitly qualifying each type is impractical

Java offers several alternatives to explicitly qualifying types

Allows use of a type's simple name in code

Types in current package do not need to be qualified

Types in the java.lang package do not need to be qualified

Object, primitive wrapper classes, String, StringBuilder, many more

Use type imports

# Type Imports

Type imports guide compiler to map simple names to qualified names

Use import keyword

Single type import

Provides mapping for a single type

# Single Type Import

```
package com.pluralsight.travel;
public class Flight {...}
```

```
package com.pluralsight.travel;
public class Passenger {...}
```

```
package com.xyzcompany.bar;
public class Beer {...}
```

```
package com.xyzcompany.bar;
public class Wine {...}
```

```
import com.pluralsight.travel.Flight;
import com.xyzcompany.bar.Beer;
import com.pluralsight.travel.Passenger;
import com.xyzcompany.bar.Wine;

Flight lax175 = new Flight(175);
Beer liteBeer = new Beer();
Passenger jane = new Passenger();
Wine merlot = new Wine();
```

## Type Imports

Type imports guide compiler to map simple names to qualified names

Use import keyword followed by qualified name

**Single type import**

Provides mapping for a single type

**Import on demand**

Provides mapping for all types in a package

# Import on Demand

```java
package com.pluralsight.travel;
public class Flight {...}
```

```java
package com.pluralsight.travel;
public class Passenger {...}
```

```java
package com.xyzcompany.bar;
public class Beer {...}
```

```java
package com.xyzcompany.bar;
public class Wine {...}
```

```java
import com.pluralsight.travel.Flight;
import com.xyzcompany.bar.Beer;
import com.pluralsight.travel.Passenger;
import com.xyzcompany.bar.Wine;

Flight lax175 = new Flight(175);
Beer liteBeer = new Beer();
Passenger jane = new Passenger();
Wine merlot = new Wine();
```

# Import on Demand

```java
package com.pluralsight.travel;
public class Flight {...}
```

```java
package com.pluralsight.travel;
public class Passenger {...}
```

```java
package com.xyzcompany.bar;
public class Beer {...}
```

```java
package com.xyzcompany.bar;
public class Wine {...}
```

```java
package com.xyzcompany.bar;
public class Flight {...}
```

```java
import com.pluralsight.travel.*;
import com.xyzcompany.bar.*;



Flight lax175 = new Flight(175);
Beer liteBeer = new Beer();
Passenger jane = new Passenger();
Wine merlot = new Wine();
```

# Single Type Import

```java
package com.pluralsight.travel;
public class Flight {...}
```

```java
package com.pluralsight.travel;
public class Passenger {...}
```

```java
package com.xyzcompany.bar;
public class Beer {...}
```

```java
package com.xyzcompany.bar;
public class Wine {...}
```

```java
package com.xyzcompany.bar;
public class Flight {...}
```

```java
import com.pluralsight.travel.Flight;
import com.xyzcompany.bar.Beer;
import com.pluralsight.travel.Passenger;
import com.xyzcompany.bar.Wine;

Flight lax175 = new Flight(175);
Beer liteBeer = new Beer();
Passenger jane = new Passenger();
Wine merlot = new Wine();
```

## Type Imports

Type imports guide compiler to map simple names to qualified names

Use import keyword followed by qualified name

**Single type import**

Provides mapping for a single type

Preferred way to import types

Most modern IDEs will add automatically

**Import on demand**

Provides mapping for all types in a package

Use with caution

Exposes code to potential breakage from changes in referenced packages

# Limiting Access to Package Content

Packages can serve as an access boundary

Often referred to as package private

Useful for creating types and features to support functionality provided by the package

Types and features are not meant to be used stand-alone

Can apply to a type

Entire type is inaccessible outside of the package

Can apply to type members

Specific members of an otherwise accessible type are inaccessible outside of the package

# Access Modifiers

| Modifier | Visibility | Usable on Types | Usable on Members |
|---|---|---|---|
| *no access modifier* | Only within its own package (a.k.a. package private) | Y | Y |
| public | Everywhere | Y | Y |
| private | Only within its own class | N * | Y |
| protected | Only within its own class and subclasses | N * | Y |

* As applies to top-level classes; can be applied to nested-classes

## Demo

## Separating CalcEngine into Packages

# Packages Provide a Unit of Distribution

Packages provide a predictable software structure

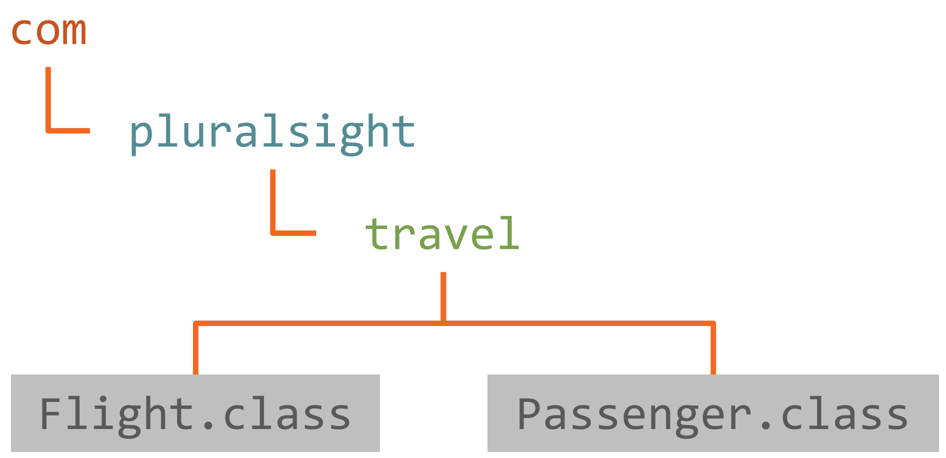
Simplifies distribution

Class files organized in hierarchical folders reflecting the package name

Each part of the package name is a separate folder

```
package com.pluralsight.travel;
public class Flight {...}
```

```
package com.pluralsight.travel;
public class Passenger {...}
```



# Archive Files

Package folder structure can be placed into an archive file

Commonly known as a jar file

Places package folder structure into a single file

Supports compressing content

Optionally includes a manifest

Provides information regarding archive content

List of name-value pairs

Commonly used to define startup class

# Creating Archive Files

The JDK provides a utility for creating archive files

The jar command-line utility

Capability included with many other tools

**Integrated development environments**

Commonly known as IDEs

- Intellij IDEA
- NetBeans

**Build automation systems**

Commonly known as build managers

- Gradle
- Maven

## Demo
## Distributing CalcEngine as a Jar File

## Summary

- A package is a group of related types
  - Package declaration must appear in source file before any type declarations
- Type name qualified by package name
  - Use import statements to map simple names to qualified names
- Packages serve as an access boundary
- Packages simplify distribution
  - Types organized hierarchically according to the package name
  - Archive files store package hierarchy in a single file