Packages and Interfaces

- **Packages** are groups of related classes.
- Packages help organize your code and provide another layer of encapsulation.
- An **interface** defines a set of methods that will be implemented by a class.
- Thus, an interface gives you a way to specify what a class will do, but not how it will do it.
- Packages and interfaces give you greater control over the organization of your program.

Defining a Package

- To create a package, put a **package** command at the top of a Java source file.
- The classes declared within that file will then belong to the specified package.
- Since a package defines a namespace, the names of the classes that you put into the file become part of that package’s namespace.

Defining a Package

- This is the general form of the package statement:

```
package pkg;
```

- Here, *pkg* is the name of the package. For example, the following statement creates a package called **mypack**:

```
package mypack;
```

Defining a Package

- Typically, Java uses the file system to manage packages, with each package stored in its own directory
- For example, the **.class** files for any classes you declare to be part of **mypack** must be stored in a directory called **mypack**.
- More than one file can include the same **package** statement. The package statement simply specifies to which **package** the classes defined in a file belong.

Defining a Package

- You can create a hierarchy of packages. To do so, simply separate each package name from the one above it by use of a period. The general form of a multileveled package statement is shown here:

package *pack1.pack2.pack3...packN*;

- Of course, you must create directories that support the package hierarchy that you create. For example,

```
package alpha.beta.gamma;
```

must be stored in .../alpha/beta/gamma, where ... specifies the path to the specified directories.

A Short Package Example

```
// A short package demonstration.
package bookpack;

class Book {
  private String title;
  private String author;
  private int pubDate;

  Book(String t, String a, int d) {
    title = t;
    author = a;
    pubDate = d;
  }
```

This file is part of the **bookpack** package.

Thus, **Book** is part of **bookpack**.

```
  void show() {
    System.out.println(title);
    System.out.println(author);
    System.out.println(pubDate);
    System.out.println();
  }
}
```

BookDemo is also part of **bookpack**.

```
class BookDemo {
  public static void main(String[] args) {
    Book[] books = new Book[5];

    books[0] = new Book("Java: A Beginner's Guide",
                        "Schildt", 2022);
    books[1] = new Book("Java: The Complete Reference",
                        "Schildt", 2022);
    books[2] = new Book("1984",
                        "Orwell", 1949);
    books[3] = new Book("Red Storm Rising",
                        "Clancy", 1986);
    books[4] = new Book("On the Road",
                        "Kerouac", 1955);

    for(int i=0; i < books.length; i++) books[i].show();
  }
}
```

Call this file **BookDemo.java** and put it in a directory called **bookpack**.

Next, compile the file. You can do this by specifying

```
javac bookpack/BookDemo.java
```

from the directory directly above **bookpack**.

Then try executing the class, using the following

```
java bookpack.BookDemo
```

As explained, **BookDemo** and **Book** are now part of the package **bookpack**.

This means that **BookDemo** cannot be executed by itself.

That is, you cannot use this command line:

```
java BookDemo
```

Instead, **BookDemo** must be qualified with its package name.

Packages and Member Access

	Private Member	Default Member	Protected Member	Public Member
Visible within same class	Yes	Yes	Yes	Yes
Visible within same package by subclass	No	Yes	Yes	Yes
Visible within same package by non-subclass	No	Yes	Yes	Yes
Visible within different package by subclass	No	No	Yes	Yes
Visible within different package by non-subclass	No	No	No	Yes

A Package Access Example

```
// Book recoded for public access.
package bookpack;

public class Book {
  private String title;
  private String author;
  private int pubDate;

  // Now public.
  public Book(String t, String a, int d) {
    title = t;
    author = a;
    pubDate = d;
  }
```

Book and its members must be **public** in order to be used by other packages.

```
    // Now public.
    public void show() {
      System.out.println(title);
      System.out.println(author);
      System.out.println(pubDate);
      System.out.println();
    }
  }
```

- To use **Book** from another package, either you must use the **import** statement described in the next section, or you must fully qualify its name to include its full package specification.
- For example, here is a class called **UseBook**, which is contained in the **bookpackext** package. It fully qualifies **Book** in order to use it.

```
// This class is in package bookpackext.
package bookpackext;

// Use the Book class from bookpack.
class UseBook {
  public static void main(String[] args) {
    bookpack.Book[] books = new bookpack.Book[5];

    books[0] = new bookpack.Book("Java: A Beginner's Guide",
                         "Schildt", 2022);
    books[1] = new bookpack.Book("Java: The Complete Reference",
                         "Schildt", 2022);
    books[2] = new bookpack.Book("1984",
                         "Orwell", 1949);
    books[3] = new bookpack.Book("Red Storm Rising",
                         "Clancy", 1986);
    books[4] = new bookpack.Book("On the Road",
                         "Kerouac", 1955);

    for(int i=0; i < books.length; i++) books[i].show();
  }
}
```

Qualify **Book** with its package name: **bookpack**.

Understanding Protected Members

- the **protected** modifier creates a member that is accessible within its package and to subclasses in other packages.
- Thus, a **protected** member is available for all subclasses to use but is still protected from arbitrary access by code outside its package.
- To better understand the effects of **protected**, let's work through an example. First, change the **Book** class so that its instance variables are **protected**, as shown here:

Understanding Protected Members

```
// Make the instance variables in Book protected.
package bookpack;

public class Book {
  // these are now protected
  protected String title;
  protected String author;
  protected int pubDate;

  public Book(String t, String a, int d) {
    title = t;
    author = a;
    pubDate = d;
  }
```

These are now **protected**.

Understanding Protected Members

```
    public void show() {
        System.out.println(title);
        System.out.println(author);
        System.out.println(pubDate);
        System.out.println();
    }
}
```

Understanding Protected Members

- Next, create a subclass of **Book**, called **ExtBook**, and a class called **ProtectDemo** that uses **ExtBook**.
- **ExtBook** adds a field that stores the name of the publisher and several accessor methods.
- Both of these classes will be in their own package called **bookpackext**. They are shown here:

Understanding Protected Members

```
// Demonstrate protected.
package bookpackext;

class ExtBook extends bookpack.Book {
  private String publisher;

  public ExtBook(String t, String a, int d, String p) {
    super(t, a, d);
    publisher = p;
  }

  public void show() {
    super.show();
    System.out.println(publisher);
    System.out.println();
  }
```

Understanding Protected Members

```
  public String getPublisher() { return publisher; }
  public void setPublisher(String p) { publisher = p; }

  /* These are OK because subclass can access
     a protected member. */
  public String getTitle() { return title; }
  public void setTitle(String t) { title = t; }
  public String getAuthor() { return author; }
  public void setAuthor(String a) { author = a; }
  public int getPubDate() { return pubDate; }
  public void setPubDate(int d) { pubDate = d; }
}
```

Access to **Book**'s members is allowed for subclasses.

Understanding Protected Members

```
class ProtectDemo {
  public static void main(String[] args) {
    ExtBook[] books = new ExtBook[5];

    books[0] = new ExtBook("Java: A Beginner's Guide",
                           "Schildt", 2022, "McGraw Hill");
    books[1] = new ExtBook("Java: The Complete Reference",
                           "Schildt", 2022, "McGraw Hill");
    books[2] = new ExtBook("1984",
                           "Orwell", 1949,
                           "Harcourt Brace Jovanovich");
    books[3] = new ExtBook("Red Storm Rising",
                           "Clancy", 1986, "Putnam");
    books[4] = new ExtBook("On the Road",
                           "Kerouac", 1955, "Viking");

    for(int i=0; i < books.length; i++) books[i].show();
```

Understanding Protected Members

```
    // Find books by author
    System.out.println("Showing all books by Schildt.");
    for(int i=0; i < books.length; i++)
      if(books[i].getAuthor() == "Schildt")
        System.out.println(books[i].getTitle());

//    books[0].title = "test title"; // Error - not accessible
  }
}
```

Access to **protected** field not allowed by non-subclass.

Importing Packages

- Here is the general form of the **import** statement:

import *pkg.classname*;

- Here, *pkg* is the name of the package, which can include its full path, and *classname* is the name of the class being imported.
- If you want to import the entire contents of a package, use an asterisk (*) for the class name. Here are examples of both forms:

Importing Packages

```
import mypack.MyClass
import mypack.*;
```

Importing Packages

- You can use **import** to bring the **bookpack** package into view so that the **Book** class can be used without qualification.
- To do so, simply add this **import** statement to the top of any file that uses **Book**.

```
import bookpack.*;
```

For example, here is the **UseBook** class recoded to use **import**:

```
// Demonstrate import.
package bookpackext;
import bookpack.*;
```

Import **bookpack**.

```
// Use the Book class from bookpack.
class UseBook {
  public static void main(String[] args) {
    Book[] books = new Book[5];

    books[0] = new Book("Java: A Beginner's Guide",
                        "Schildt", 2022);
    books[1] = new Book("Java: The Complete Reference",
                        "Schildt", 2022);
    books[2] = new Book("1984",
                        "Orwell", 1949);
    books[3] = new Book("Red Storm Rising",
                        "Clancy", 1986);
    books[4] = new Book("On the Road",
                        "Kerouac", 1955);

    for(int i=0; i < books.length; i++) books[i].show();
  }
}
```

Now, you can refer to **Book** directly, without qualification.

Java's Class Library Is Contained in Packages

- Java defines a large number of standard classes that are available to all programs.
- This class library is often referred to as the Java API (Application Programming Interface).
- The Java API is stored in packages. At the top of the package hierarchy is **java**.
- Descending from **java** are several subpackages.

Java's Class Library Is Contained in Packages

- Here are a few examples:

Subpackage	Description
java.lang	Contains a large number of general-purpose classes
java.io	Contains I/O classes
java.net	Contains classes that support networking
java.util	Contains a large number of utility classes, including the Collections Framework
java.awt	Contains classes that support the Abstract Window Toolkit