Packaging and Compartmentalization

"Classes namespaces, visibility, and accessibility"

Advanced Programming

Shakirullah Waseeb
shakir.waseeb@gmail.com

Nangarhar University

March 1, 2017

Agenda

1. Introduction
2. Packages in Java
3. Questions and Discussion

Introduction

- Compartmentalization (dividing into groups and categories) of class name space

Introduction

- Compartmentalization (dividing into groups and categories) of class name space
- To avoid class name collision

Introduction

- Compartmentalization (dividing into groups and categories) of class name space
- To avoid class name collision
- Mechanisms for partitioning the class name space into more manageable chunks

Introduction

- Compartmentalization (dividing into groups and categories) of class name space
- To avoid class name collision
- Mechanisms for partitioning the class name space into more manageable chunks
- Naming and visibility control mechanism

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package
- Allows classes to have intimate knowledge of each other, but not expose that knowledge to external world

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package
- Allows classes to have intimate knowledge of each other, but not expose that knowledge to external world
- A java source file can contain any (or all) of the following four internal parts:
  - A single package statement (optional)

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package
- Allows classes to have intimate knowledge of each other, but not expose that knowledge to external world
- A java source file can contain any (or all) of the following four internal parts:
  - A single package statement (optional)
  - Any number of import statements (optional)

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package
- Allows classes to have intimate knowledge of each other, but not expose that knowledge to external world
- A java source file can contain any (or all) of the following four internal parts:
  - A single package statement (optional)
  - Any number of import statements (optional)
  - A single public class declaration (required)

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package
- Allows classes to have intimate knowledge of each other, but not expose that knowledge to external world
- A java source file can contain any (or all) of the following four internal parts:
  - A single package statement (optional)
  - Any number of import statements (optional)
  - A single public class declaration (required)
  - Any number of classes private to the package (optional)

Class packaging in Java

- Java uses ***package*** to compartmentalize class name space
- Class can be defined inside a **package** that are not accessible outside the package
- Even class members can be defined are only exposed to other members of the same package
- Allows classes to have intimate knowledge of each other, but not expose that knowledge to external world
- A java source file can contain any (or all) of the following four internal parts:
  - A single package statement (optional)
  - Any number of import statements (optional)
  - A single public class declaration (required)
  - Any number of classes private to the package (optional)

Defining a Package

- Quite easy:

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored
- if package statement is omitted, class names are put into default package, having no name

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored
- if package statement is omitted, class names are put into default package, having no name
- general form of **package** statement:

  package *pkgname*
- Java uses file system directories to store packages

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored
- if package statement is omitted, class names are put into default package, having no name
- general form of **package** statement:

  package *pkgname*
- Java uses file system directories to store packages
- More than one file can include the same **package** statement

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored
- if package statement is omitted, class names are put into default package, having no name
- general form of **package** statement:

  package *pkgname*
- Java uses file system directories to store packages
- More than one file can include the same **package** statement
- packages hierarchy can be created using a period

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored
- if package statement is omitted, class names are put into default package, having no name
- general form of **package** statement:

  package *pkgname*
- Java uses file system directories to store packages
- More than one file can include the same **package** statement
- packages hierarchy can be created using a period

  package *pkg1[.pkg2[.pkg3]]*

Defining a Package

- Quite easy:simply include a **package** statement as the first statement in a Java source file
- any classes declared in this file will belong to specified package
- **package** statement defines a name space in which classes are stored
- if package statement is omitted, class names are put into default package, having no name
- general form of **package** statement:

  package *pkgname*
- Java uses file system directories to store packages
- More than one file can include the same **package** statement
- packages hierarchy can be created using a period

  package *pkg1[.pkg2[.pkg3]]*

  package java.awt.image;

Visibility of class members

- Java addresses four categories of visibility for class members:

Visibility of class members

- Java addresses four categories of visibility for class members:
  - Subclasses in the same package

Visibility of class members

- Java addresses four categories of visibility for class members:
  - Subclasses in the same package
  - Non-subclasses in the same package

Visibility of class members

- Java addresses four categories of visibility for class members:
  - Subclasses in the same package
  - Non-subclasses in the same package
  - Subclasses in different packages

Visibility of class members

- Java addresses four categories of visibility for class members:
  - Subclasses in the same package
  - Non-subclasses in the same package
  - Subclasses in different packages
  - Classes that are neither in the same package nor subclasses

	Private	No modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Table 9-1. *Class Member Access*

[1]

Importing Packages

- Use **import** statement to bring certain classes, or entire package, into visibility

Importing Packages

- Use **import** statement to bring certain classes, or entire package, into visibility
- In Java **import** statements occur immediately following the **package** statement (if it exists) and before any class definition

Importing Packages

- Use **import** statement to bring certain classes, or entire package, into visibility
- In Java **import** statements occur immediately following the **package** statement (if it exists) and before any class definition
- General form of **import** statement:
- **import** *pkg1[.pkg2[.classname—*]]*

Importing Packages

- Use **import** statement to bring certain classes, or entire package, into visibility
- In Java **import** statements occur immediately following the **package** statement (if it exists) and before any class definition
- General form of **import** statement:
- **import** *pkg1[.pkg2[.classname—*]]*
- **import** *java.util.Date*

Importing Packages

- Use **import** statement to bring certain classes, or entire package, into visibility
- In Java **import** statements occur immediately following the **package** statement (if it exists) and before any class definition
- General form of **import** statement:
- **import** *pkg1[.pkg2[.classname—*]]*
- **import** *java.util.Date*
- **import** *java.lang.**

Your Turn: Time to hear from you!

[1]

[1] https://fensafitters.files.wordpress.com/2013/07/3d095.jpg

References

Herbert Schildt

The complete reference Java2, 5th Edition .

McGraw-Hill/Osborne, 2002.