



Packages and Information Hiding

- Packages as a structuring concept
- Information hiding through access specifiers

OOAD 1998/99

Claudia Niederée, Joachim W. Schmidt
Software Systems Institute

c.niederee@tu-harburg.de <http://www.sts.tu-harburg.de>

Package: the Library Unit



- Managing “name spaces”
 - Class members are already hidden inside class
 - Class names could clash
 - Need completely unique name
- Packages
 - organize classes into libraries
 - structure name space for classes
 - restrict visibility
 - may be nested

Creating a Library of Classes

```
package mypackage;
```

```
public class Class1{ ... }
```

Class1.java

```
package mypackage.mysubpackage1;
```

```
public class Class5{ ... }
```

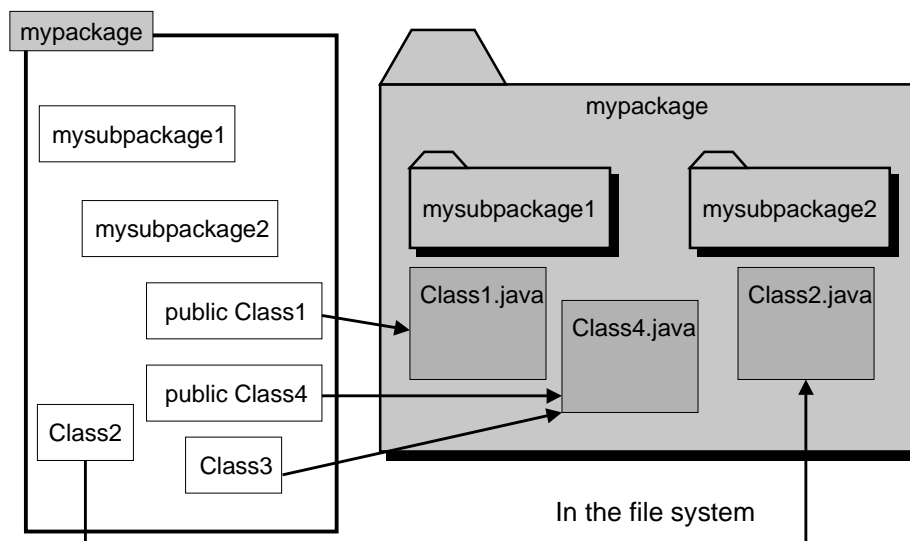
Class5.java

- **public** class is under the umbrella **mypackage**
- Client programmer must import the package

```
import mypackage.*;
```

```
import java.util.Vector;
```


Organizing Your Packages



Compilation Units

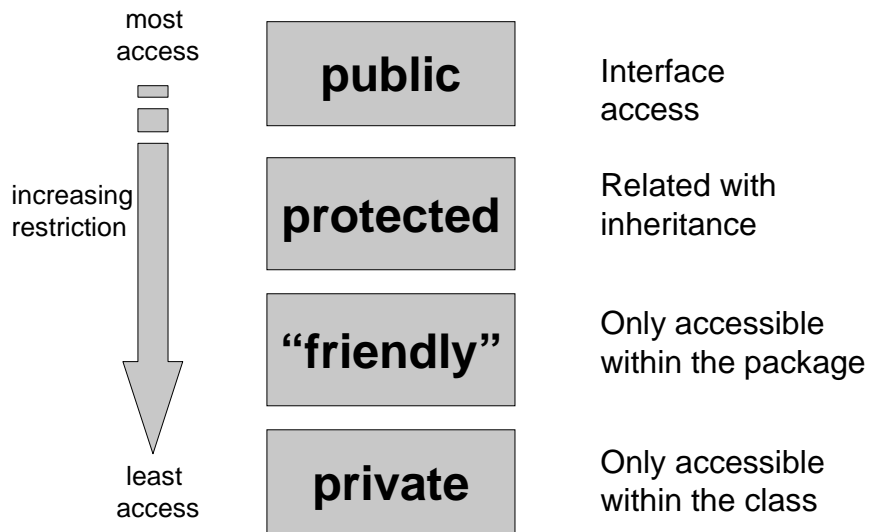
- Compilation units (**.java** files)
 - Name of **.java** file == name of single **public** class
 - Other non-**public** classes are not visible
 - Each class in file gets its own **.class** file
 - Program is a bunch of **.class** files

Library Location

- Creating unique package names
 - Location on disk encoded into package name
 - Convention: first part of package name is Internet domain name of class creator (reverse)
- sts.tu-harburg.de  de.tu-harburg.sts.mypackage
- Java interpreter
 - uses CLASSPATH environment variable as starting point for search
 - looks for package x.y.z in a folder on the path x/y/z
 - CLASSPATH takes care of first part:

```
CLASSPATH=. ;D:\JAVA\LIB;C:\DOC\JavaClasses
```

Java Access Control



OOAD98/99-STS-Objects, Classes, and Information Hiding

7

"Friendly"

- Default access, has no keyword
- public to other members of the same package, private to anyone outside the package.
- Easy interaction for related classes (that you place in the same package)
- Also referred to as "package access"

OOAD98/99-STS-Objects, Classes, and Information Hiding

8

public: Interface Access

```
package food.dessert;
public class Cookie {
    public Cookie() {
        System.out.println("Cookie constructor");
    }
    void decorate() { System.out.println("Cookie
decorated"); }
}

// Separate file:
import food.dessert.*;
public class Dinner {

    public static void main(String args[]) {
        Cookie c = new Cookie();
        //! c.decorate(); // can't access
    }
}
```

OOAD98/99-STS-Objects, Classes, and Information Hiding

9

private: Can't Touch That!

```
class Resource {
    private static count = 5;
    private Resource() {}
    static Resource makeAResource() {
        if (count > 0) {
            count --; return new Resource();
        }
        else return null;
    }
}

public class ResourceUser {
    public static void main(String args[]) {
        //! Resource r = new Resource();
        Resource r = Resource.makeAResource();
    }
}
```

OOAD98/99-STS-Objects, Classes, and Information Hiding

10

Class Access

- Classes as a whole can be **public** or “friendly”
- Only one **public** class per file, usable outside the library
- All other classes “friendly,” only usable within the library