Visibility

- Attribute and method access control with modifiers:

```
public [static] type variable;
private [static] type variable;
protected [static] type variable;
```

- Similarly for methods
- Public members can be accessed by every class in the program
- Private members can be accessed only by the same class (although an object can access another object's private members if and only if the two objects are of the same class.)
- Protected members can be accessed by anyone in the same *package*.

Packages and protected

- Large scale Java programs are divided into packages.
- Each package consists of many classes.
- Each package is stored in a separate directory
- To put a class in a package one must:
  - Put it in the package directory
  - Declare the class in the package by using the following in the first line:

    `package` *`package_name`* `;`

- To use a public class from a package (after the package declaration, but before the class declarations):

  `import` *`package_name.class_name`* `;`

Packages (contd.)

- To use all classes in a package:

  `import` *package_name*`.*;`

- Only public classes can be imported in other packages.
- Packages can have subpackages (in subdirectories)
- If a class, attribute or method is declared as `protected`, then it is visible by all classes in the same package only.

Packages (contd.)

```
// File A.java in directory mypackage
package mypackage;
import pack1.C1;
import pack2.C2;
// ...
public class A {
  //...
}
class B {
  //...
}

// File C.java in directory mypackage
package MyPackage;
import pack3.C3;
import pack2.C2;
public class C {
  //...
}
class D {
  //...
}
```

Packages namespaces

- Each package has its own individual namespace
- Different packages can have classes with the same name
- The full name of a class is given by:

  `package_name.class_name`

which must be stored in a file called

  `class_name.java`

in a directory named

  `package_name`

- Windows notation: `package_name\class_name.java`
- Unix notation: `package_name/class_name.java`

Example

```
// File: pack1\A.java
package pack1;
class A
{
    private String x;
    A(String x)
    {
        this.x = x;
    }
    protected String get_x()
    {
        return x;
    }
}
```

Example (contd.)

```
// File: pack1\B.java
package pack1;
public class B
{
    private A my_a;
    public B()
    {
        my_a = new A(“hello”);
    }
    public String m()
    {
        return my_a.get_x();
    }
}
```

Example (contd.)

```
// File: Test.java
import pack1.*;
public class Test
{
  public static void main(String[] args)
  {
    B my_b = new B();
    System.out.println(my_b.m());

    A some_a = new A(“bye”); // Error
  }
}
```

Example (contd.)

```
// File: pack1\A.java
package pack1;
public class A
{
    private String x;
    public A(String x)
    {
        this.x = x;
    }
    protected String get_x()
    {
        return x;
    }
}
```

Example (contd.)

```
// File: Test.java
import pack1.*;
public class Test
{
  public static void main(String[] args)
  {
    B my_b = new B();
    System.out.println(my_b.m());

    A some_a = new A(“bye”); // OK
    System.out.println(some_a.get_x()); //Error
  }
}
```

Example (contd.)

```
// File: pack2\A.java
package pack2;
public class A
{
    public String get_x()
    {
        return “x”;
    }
}
```

Example (contd.)

```
// File: Test.java
import pack1.A;
import pack2.A;   // Error
public class Test
{
  public static void main(String[] args)
  {
    B my_b = new B();
    System.out.println(my_b.m());

    A some_a = new A("bye"); // Which A?
    A other_a = new A(); // Which A?
  }
}
```

Namespace conflicts

- Different packages can have classes with the same name
- A class C from a package p can be accessed by:
  - Using the import statement at the beginning of the file which uses it

    ```
    import p.C;
    ```

  - Or, using the fully qualified name of the class in place, e.g.

    ```
    p.C myCobject = new p.C();
    ```

Example (contd.)

```
// File: Test.java
import pack1.*;
public class Test
{
  public static void main(String[] args)
  {
    B my_b = new B();
    System.out.println(my_b.m());

    A some_a = new A(“bye”);
    pack2.A other_a = new pack2.A();
  }
}
```

Subpackages

- Subpackages must be in subdirectories.
- If a package `p` has `s` as a subpackage, then there must be directories `p` and `p\s`
- If the subpackage `s` of `p` has a public class `A`, the fully quallified name of `A` is

  ```
  p.s.A
  ```

- so it must be imported as

  ```
  import p.s.A;
  ```

Nested classes

- A class can have internal class definitions:

```
class Class_name
{
    Attribute definitions;

    Method definitions;

    Class definitions;
}
```

- These are useful to represent private classes which only the container class knows about.

- An instance of a contained class always has an enclosing instance of the container class.

Example (contd.)

```
class A {
    double x;
    void inc()
    {
        B u = new B();
        u.set_x(7);
        x = u.get_x();
    }
    class B {
        private double x;
        void set_x(double x) { this.x = x; }
        double get_x() { return x; }
    }
}
```

Scope in nested classes

```
class A {
    String x;
    void inc()
    {
        B u = new B();
        x = "rty";
        u.set_y("qwe");
        x = u.get_y();
    }
    public class B {
        private String y;
        void set_y(String y) { this.y = x + y; }
        String get_y() { return y; }
    }
}
```