

CIS 27P

Class 7

Java Packages

- Create with a package statement as first statement in a Java file
- import statement when using other packages
- import reduces need to specify full package when referring to classes
- Periods create a hierarchy of packages

Java Packages (cont)

- Package hierarchy must be reflected in file system
- Run main() with complete hierarchy in command
- Default package
- All API classes in packages, none in default.
- Possible name collision
- Reverse URL convention

Classpath

- Shows possible root of all packages
- Good idea to include current directory
- Use `-cp` flag of Java executable

Class Member Access

	Priv.	Def.	Prot.	Pub.
Same Class	Yes	Yes	Yes	Yes
Same pkg subclass	No	Yes	Yes	Yes
Same pkg non subclass	No	Yes	Yes	Yes
Different pkg subclass	No	No	Yes	Yes
Different pkg non subclass	No	No	No	Yes

Interfaces

- A fully abstract class.
- No method bodies.
- Access must be public or not specified
- Use the implements keyword
- A class can implement multiple interfaces

Sample Interface

```
public interface Employee
{
    public final String company = "ACME";

    public void hire( int salary );
    public void promote();
}
```

Implementing Interfaces

- Classes that implement interfaces must define all interface methods with exact signature, or be abstract.
- Classes must be public or with no access modifier.
- Methods that implement interface methods must be public.
- Method signatures of implementing class must be exactly the same as interface.

Interfaces - Misc.

- Classes implementing interfaces can define their own additional members.
- Variables declared in interfaces act as final in implementing classes
- An interface can extend another. Classes that implement a sub-interface must implement all methods in the hierarchy.

```
public class RegularEmployee implements Employee
{
    String firstName;
    String lastName;
    int salary;
    int positionLevel;

    public RegularEmployee( String fn, String ln )
    {
        firstName = fn;
        lastName = ln;
        salary = 0;
        positionLevel = 0;
    }
    public void hire( int sal )
    {
        salary = sal;
    }
    public void promote()
    {
        positionLevel++;
        salary += 1000;
    }
    public void printInfo()
    {
        System.out.print("Employee: "+firstName+" "+lastName+" ");
        System.out.println("Salary: "+salary+" Level: "
            +positionLevel);
    }
}
```

```
public class Manager implements Employee
{
    String firstName;
    String lastName;
    int salary;
    int positionLevel;

    public Manager( String fn, String ln )
    {
        firstName = fn;
        lastName = ln;
        salary = 0;
        positionLevel = 10;
    }
    public void hire( int sal )
    {
        salary = sal;
    }
    public void promote()
    {
        positionLevel++;
        salary += 5000;
    }
    public void printInfo()
    {
        System.out.print("Manager: "+firstName+" "+lastName+" ");
        System.out.println("Salary: "+salary+" Level:
            "+positionLevel);
    }
}
```

```
public class UseEmployee
{
    public static void main( String args[] )
    {
        Manager m = new Manager("Scott", "McNealy");
        m.printInfo();
        m.hire( 10000 );
        m.promote();
        m.printInfo();

        RegularEmployee re = new RegularEmployee("James",
            "Gosling");
        re.printInfo();
        re.hire( 10000 );
        re.promote();
        re.printInfo();

        Employee e = m;
        e.promote();
        ((Manager)e).printInfo();

        e = re;
        re.promote();
        ((RegularEmployee)e).printInfo();
    }
}
```

Exercise

Write an interface named `Shape` with an `area` method and a `PI` variable defined as 3.14.

Write two objects named `Square` and `Circle` that implement `Shape`. `Square` should have a constructor that accepts the length of a side, and `Circle` should have a constructor that accepts length of the radius.

```
public interface Shape
{
    double PI = 3.14;
    public double area();
}
class Circle implements Shape
{
    double radius;
    Circle( double r )
    {
        radius = r;
    }
    public double area()
    {
        return PI * radius * radius;
    }
}
class Square implements Shape
{
    double lengthOfSide;
    Square( double los )
    {
        lengthOfSide = los;
    }
    public double area()
    {
        return lengthOfSide * lengthOfSide;
    }
}
```

```
public class UseShape
{
    public static void main( String[] args )
    {
        Circle c = new Circle( 2.0 );
        System.out.println("c.area(): "+c.area());

        Square s = new Square( 3.0 );
        System.out.println("s.area(): "+s.area());

        Shape sh = c;
        System.out.println("sh.area(): "+sh.area());

        sh = s;
        System.out.println("sh.area(): "+sh.area());

    }
}
```

OUTPUT:

```
c.area() : 12.56
s.area() : 9.0
sh.area() : 12.56
sh.area() : 9.0
```