Java Programming I

Session 5

EXTENDING CLASSES AND INHERITANCE ACCESS QUALIFIERS

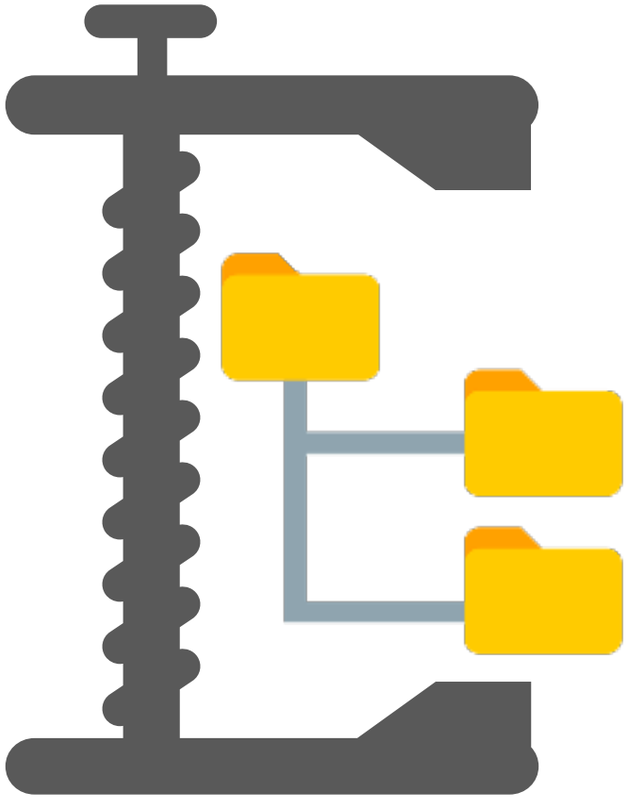
Juan Carlos Moreno - UCLA Ex

Agenda

- **Packages**
- **Super**
- **Instance of**
- **Abstract**
- **Interfaces**
- **Final**
- **Coding exercises**

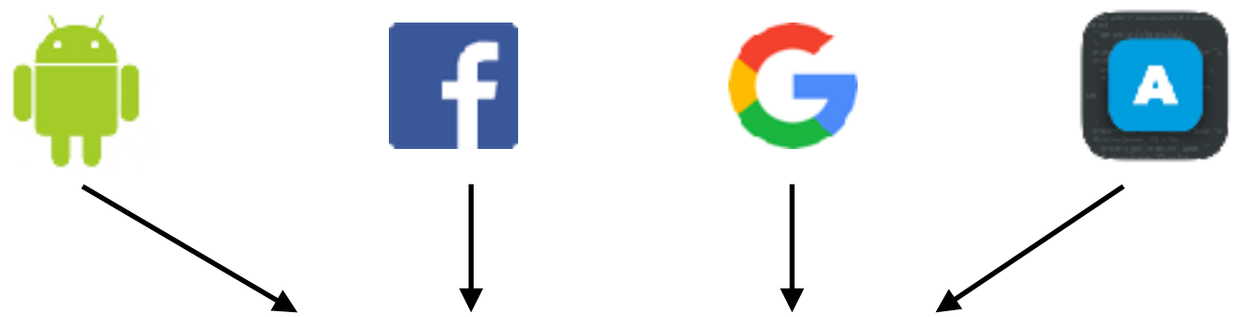
Packages

A way to organize objects



Packages

Grouping avoiding collisions



public class Account

?

android.accounts.Account

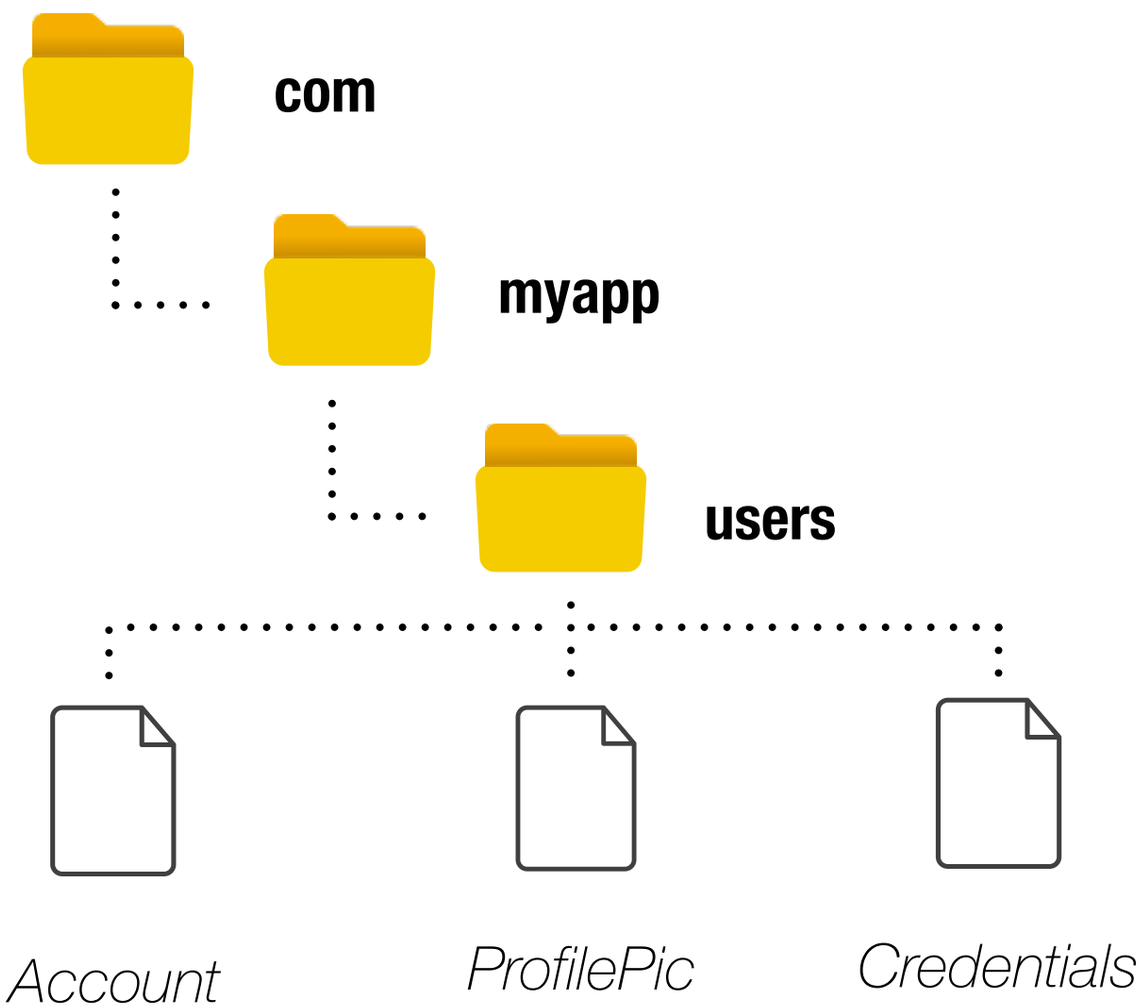
com.myapp.users.Account

com.facebook.accountkit.Account

com.google.api.client.auth.Account

Packages

Grouping avoiding collisions



import

getting classes from packages

Super

calling the parent class

```java
package edu.ucla.ex.java.summer;
import Thread;

public class Cookie {
    private Shape shape;
    private Ingredient ingredients[];
    public String flavor_name;
    public Oven oven;

    public Cookie(Ingredient new_ingredients, CookieCutter cutter){
        oven = Oven.singletonInstance();
        this.ingredients = new_ingredients;
        this.mix();
        this.shape = cutter.cut();
        this.bake();
    }

    public void mix() {
        int times = 5;
        for (int fold = 0; fold < times; fold++) {
            Random rand = new Random();
            for (int i = 0; i < this.ingredients.length; i++) {
                int random_element = rand.nextInt(100) % this.ingredients.length;
                Ingredient to_swap = this.ingredients[i];
                this.ingredients[i] = this.ingredients[random_element];
                this.ingredients[random_element] = to_swap;
            }
        }
    }

    public void bake(){
        while(oven.temperature == 400){
            Thread.sleep(50000);
            if (oven.check()) {
                break;
            }
        }
    }
}
```

Super

calling the parent class

```
public class ChocolateChipCookie extends Cookie{

    private Ingredient choc_chips[];

    public void mix() {
        super.mix();
        int len = choc_chips.length + this.ingredients;
        Ingredient[] new_ingredients = new Ingredient[len];

        // Add cookies to the end
        for (int i=0; i<len; i++)
        {
            if (i < this.ingredients.length)
            {
                new_ingredients[i] = this.ingredients[i];
            }else{
                int choc_index = i - this.ingredients.length;
                new_ingredients[i] = this.choc_chips[choc_index];
            }

        }
    }
}
```

instanceof

Are all of these cookies?

```
if (my_oreo instanceof Cookie){
    CookieMonster.eat(my_oreo);
}
```

Abstract class

Not quite there yet

Abstract class

calling the parent class

```
public abstract class Cookie {
    private Shape shape;
    private Ingredient ingredients[];
    public String flavor_name;
    public Oven oven;

    public Cookie(Ingredient new_ingredients, CookieCutter cutter){
        oven = Oven.singletonInstance();
        this.ingredients = new_ingredients;
        this.mix();
        this.shape = cutter.cut();
        this.bake();
    }

    public void mix() {
        ....
        ....
    }

    public void bake(){
        ....
        ....
    }

    abstract void decorate();
}
```

Interface

Just the rules

```
public interface Chocolate

public class ChocolateCookie extends Cookie implements Chocolate
```

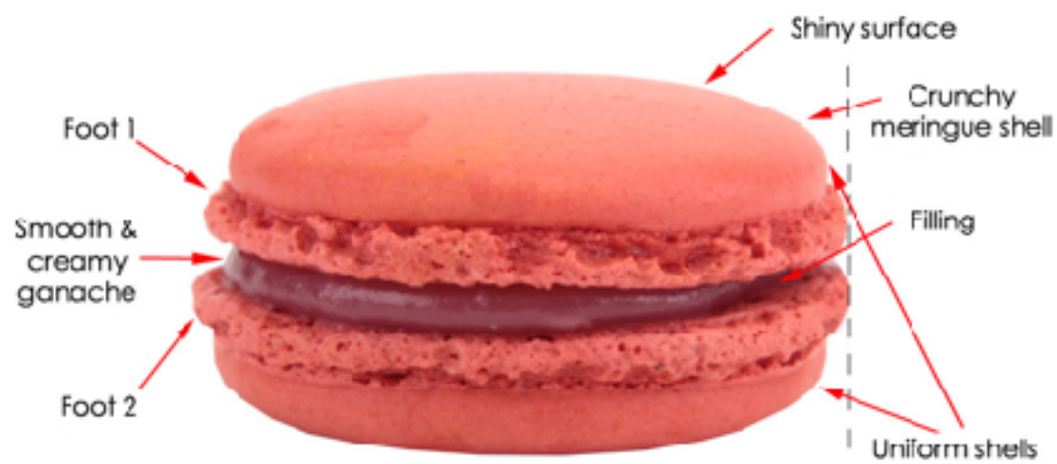
class

+

Interface

Final

Don’t mess with perfection



Final prevents the class from being subclassed or overridden

```
public final class Macaron extends Cookie {
   // Can’t be inherited
}

public class MySecretCookie extends Cookie {
   // Can be inherited
   private final  Int folds = 10;  // Can’t change this value

   public final void mix() {
        // My secret method can’t be changed
   }
}
```

Coding Challenge

The Bakery

```
public interface Bakery{

    public Oven bakeryOven;

    public Cookie[] order(String cookie[], quantity []);
}

// Create class UCLABakery implementing Bakery

public class CookieOrder{
    public static void main(String args[]){
        UCLABakery bakery = new UCLABakery();
        String cookie_order[] = [“ChocolateChip”, “SnickerDoodle”, “Macaron”, “Oreo”];
        int quantities[] = {5, 12, 12, 5};
        Cookie cookies[] = bakery.order(cookie_order, quantities);

        // Testing
        for (Cookie cookie : cookies)
        {
            for (int i=0; i<cookie_order; i++){
                String cookie_name = cookie_order[i];
                if (cookie_name == cookie.flavor_name){
                    System.out.println(cookie.getClass.getName());
                    break;
                }
            }
        }
    }
}
```