Interface Example

LAB 8

Course Teacher

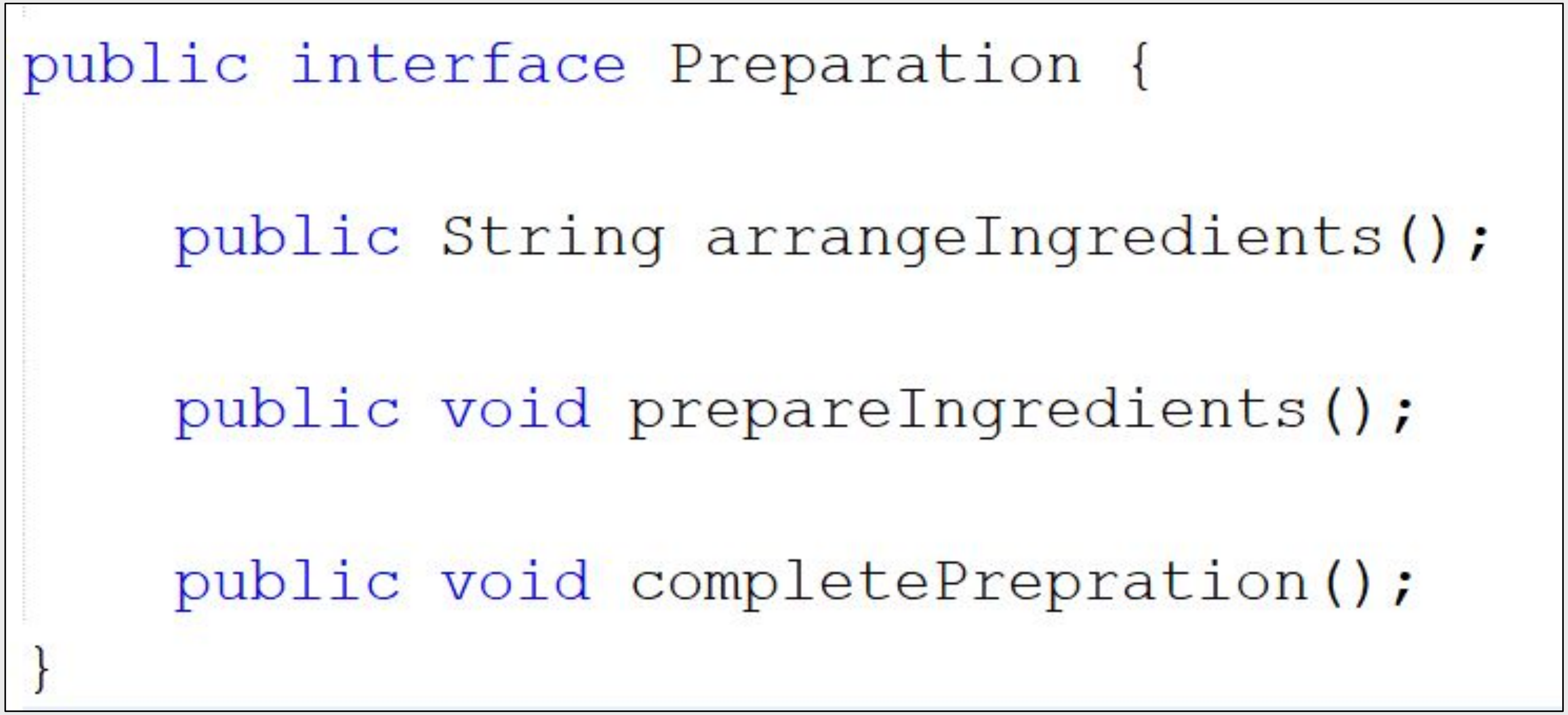
Dr. Shahriar Mahbub, Professor
Tanvir Rouf Shawon, Lecturer
Ashna Nawar Ahmed, Lecturer

Suppose we are making a website where different cooking games can be played. All games do some common things which must be followed.

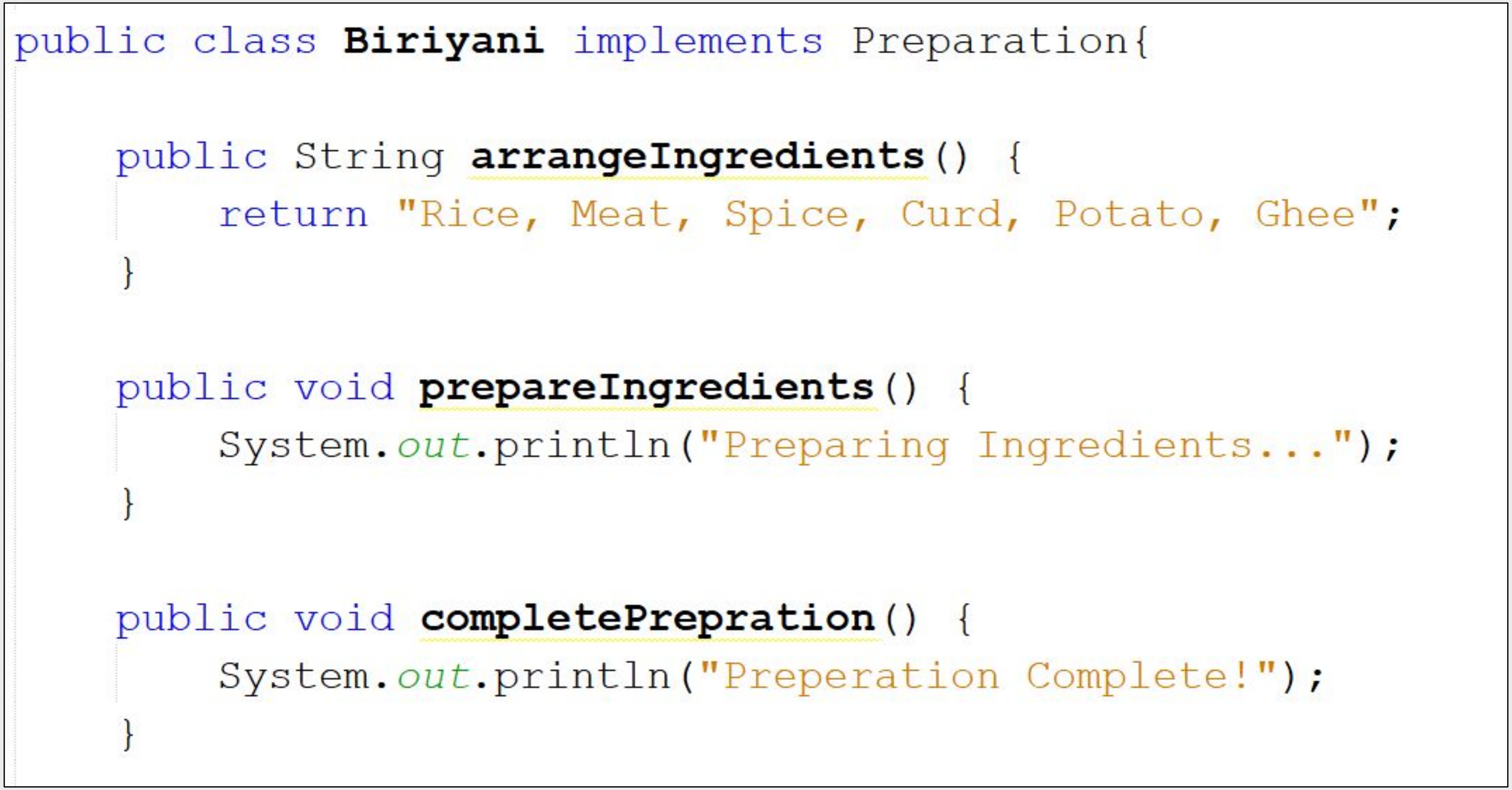
Basic Activities

- Preparation
  - arrangeIngredients
  - prepareIngredients
  - completePreparation
- Assembling
- Selling

Create an Interface for the first Activity.



```
public interface Preparation {

    public String arrangeIngredients();

    public void prepareIngredients();

    public void completePrepration();
}
```

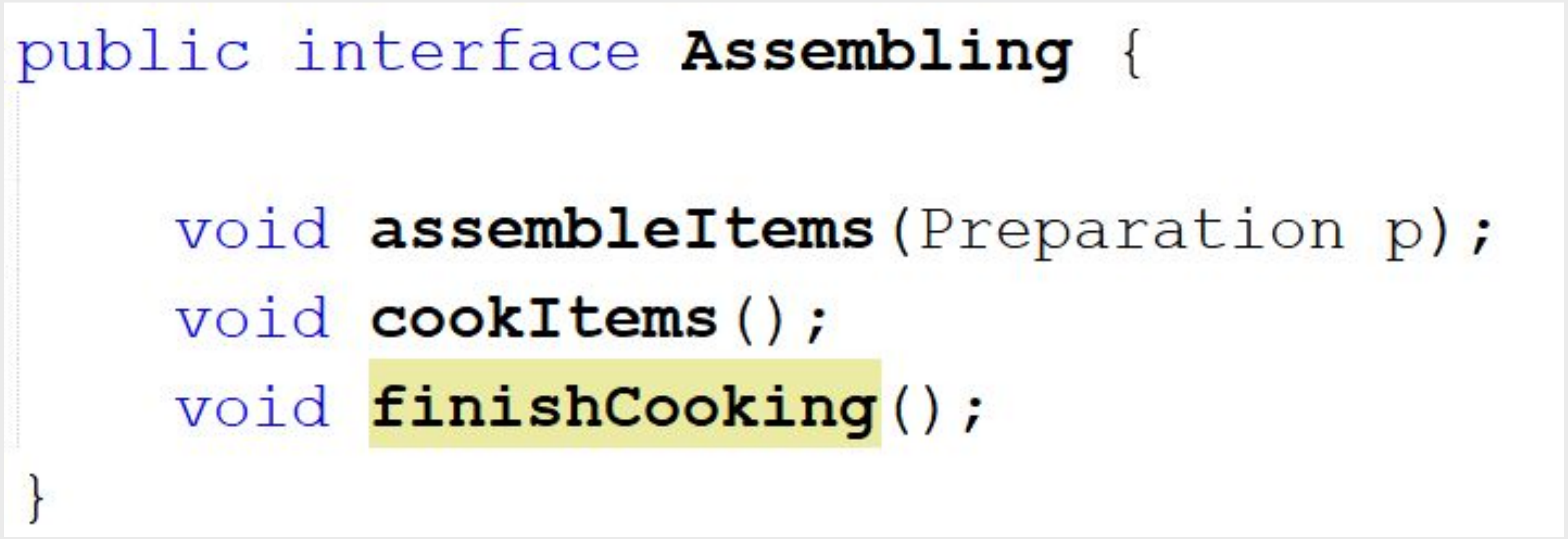


```java
public class Biriyani implements Preparation{

    public String arrangeIngredients() {
        return "Rice, Meat, Spice, Curd, Potato, Ghee";
    }

    public void prepareIngredients() {
        System.out.println("Preparing Ingredients...");
    }

    public void completePrepration() {
        System.out.println("Preperation Complete!");
    }
```

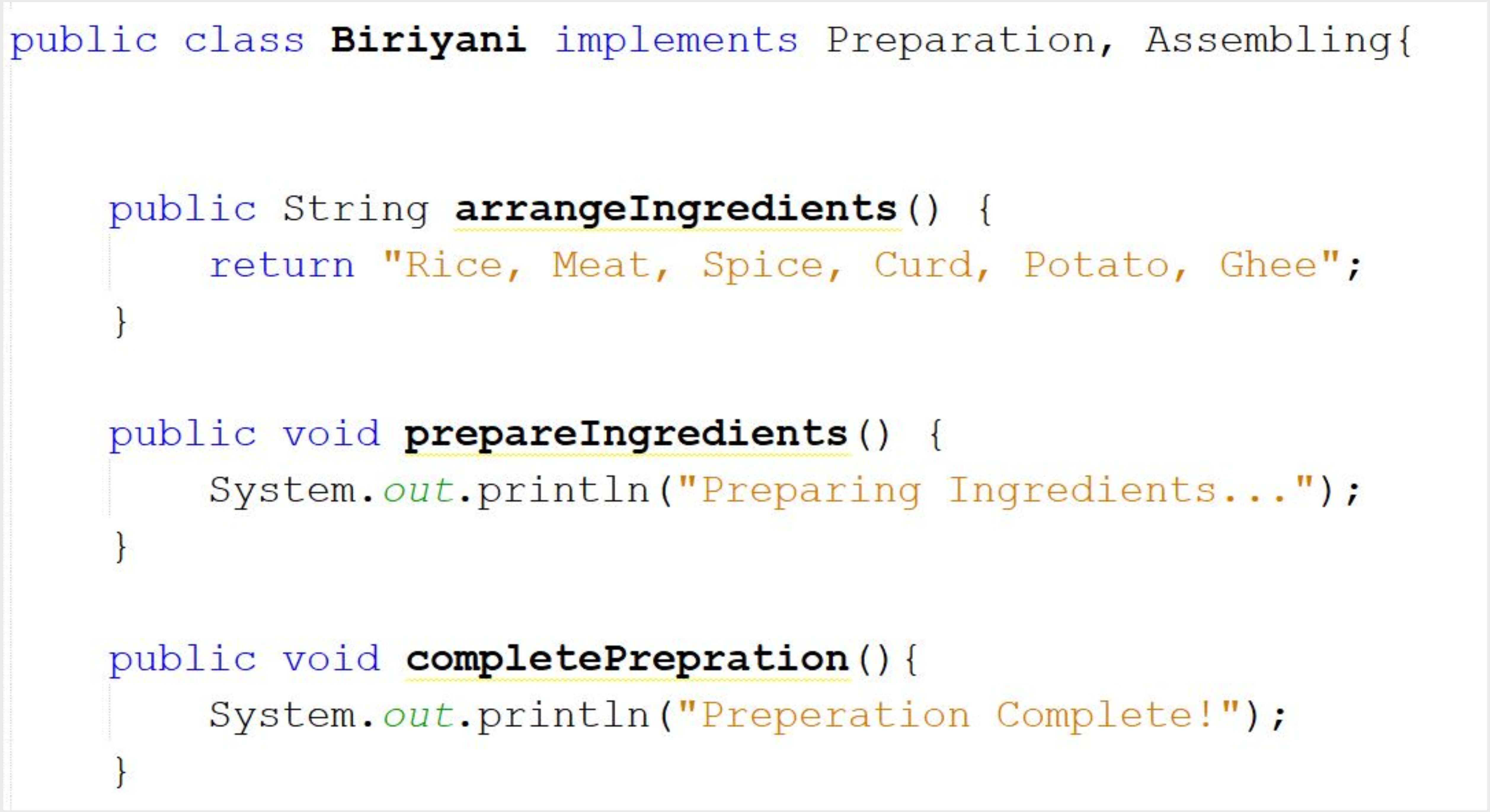
Basic Activities

- Assembling
  - assembleItems
  - cookItems
  - finishCooking

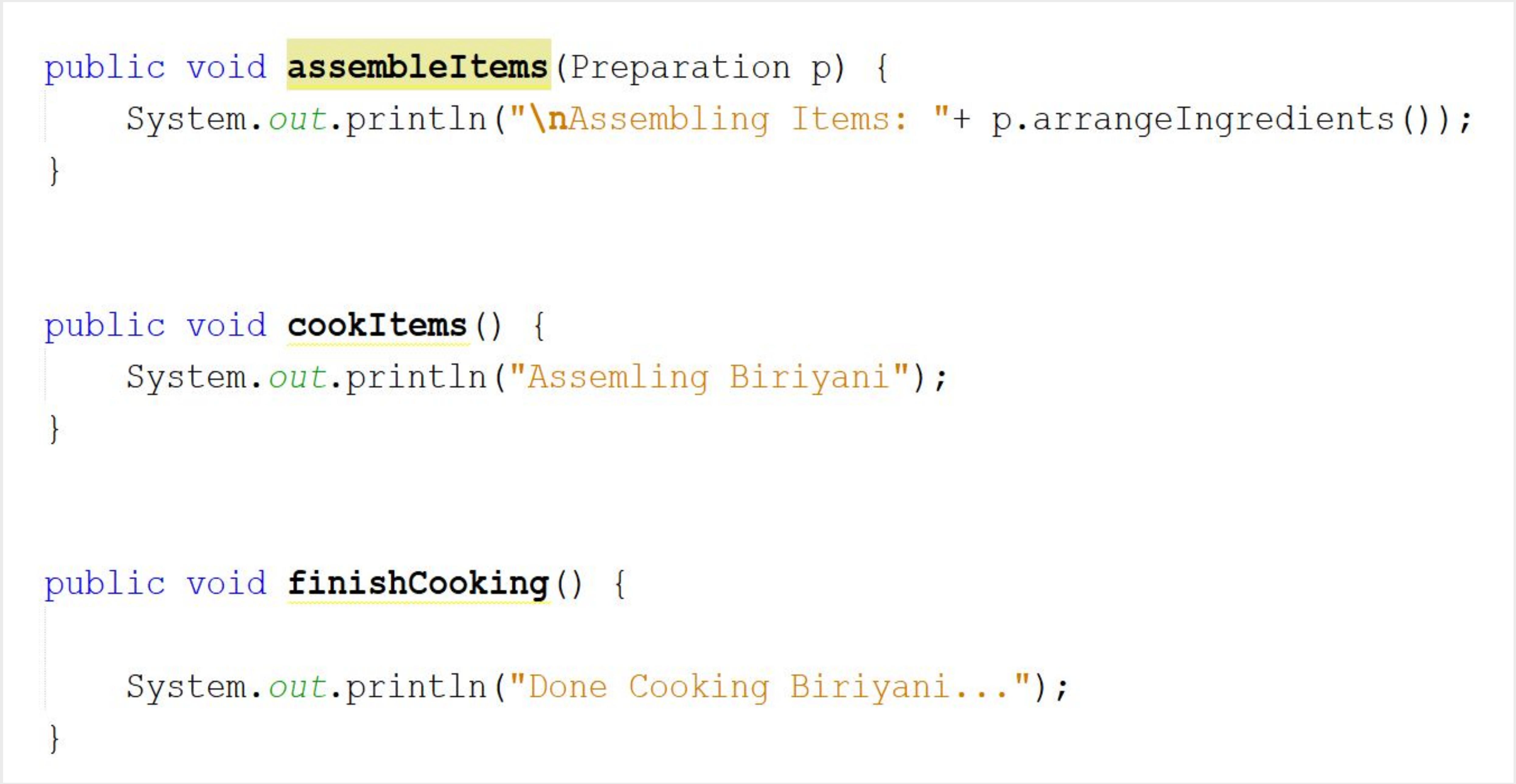
Create an Interface for the 2nd Activity.



```
public interface Assembling {


    void assembleItems(Preparation p);
    void cookItems();
    void finishCooking();
}
```



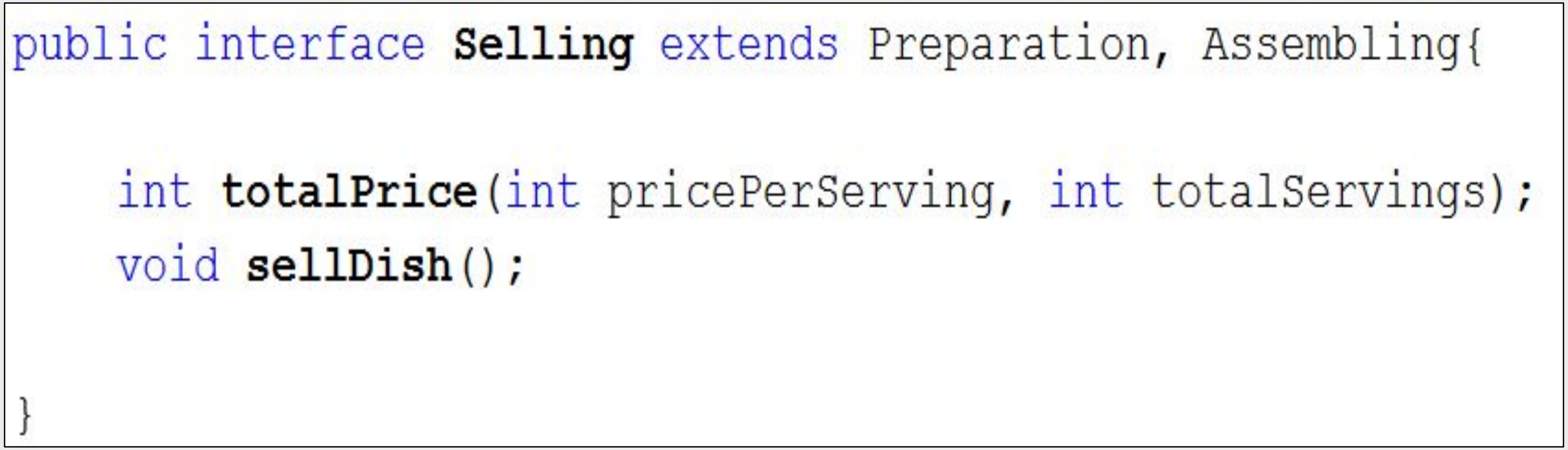
```java
public class Biriyani implements Preparation, Assembling{


    public String arrangeIngredients() {
        return "Rice, Meat, Spice, Curd, Potato, Ghee";
    }

    public void prepareIngredients() {
        System.out.println("Preparing Ingredients...");
    }

    public void completePrepration(){
        System.out.println("Preperation Complete!");
    }
```



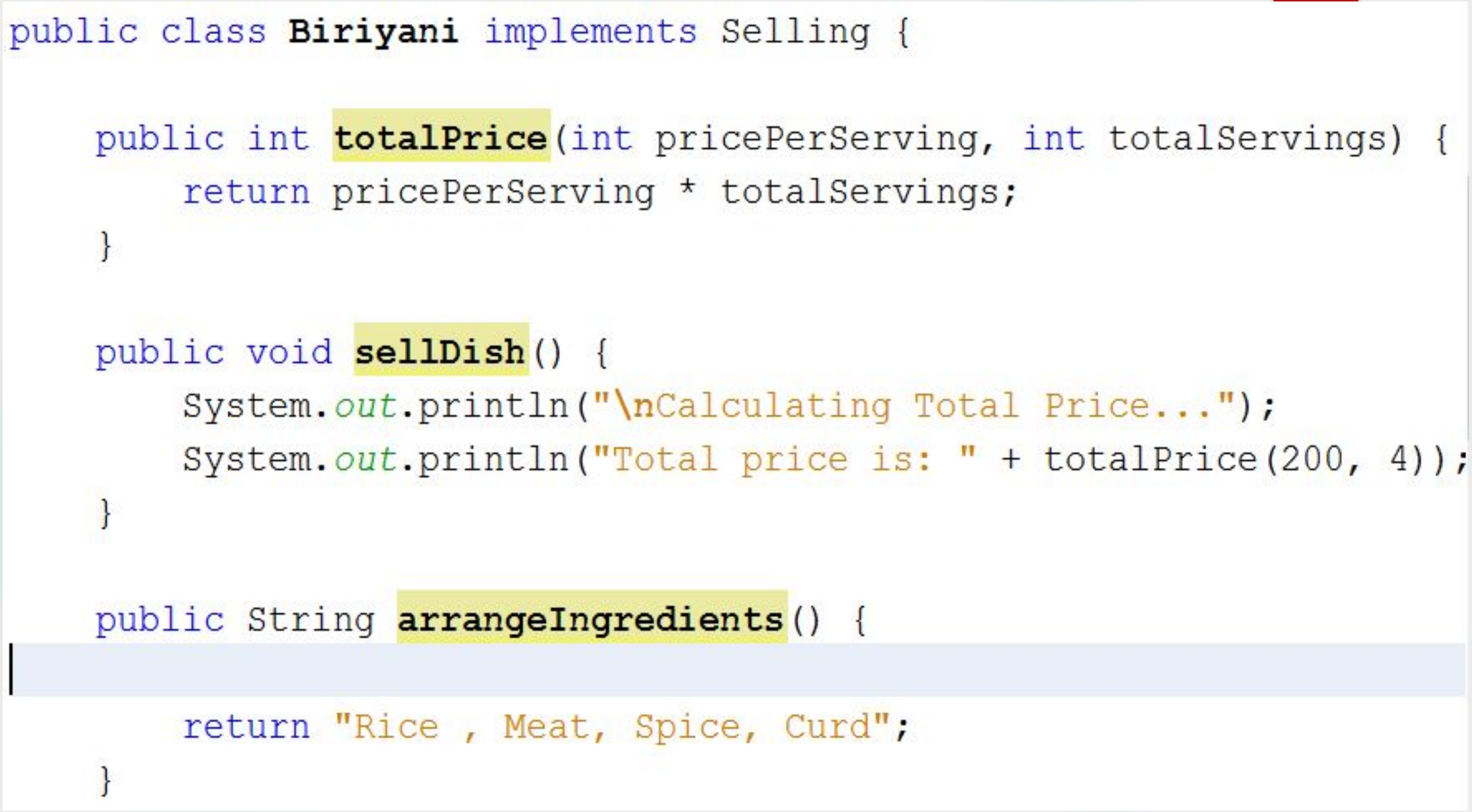
```java
public void assembleItems(Preparation p) {
    System.out.println("\nAssembling Items: "+ p.arrangeIngredients());
}



public void cookItems() {
    System.out.println("Assemling Biriyani");
}



public void finishCooking() {

    System.out.println("Done Cooking Biriyani...");
}
```

Suppose now we are designing in such a way that in order to sell the product must be prepared and assembled first.

Then we can extend the 2 interfaces to Selling interface.



```java
public interface Selling extends Preparation, Assembling{


    int totalPrice(int pricePerServing, int totalServings);
    void sellDish();


}
```

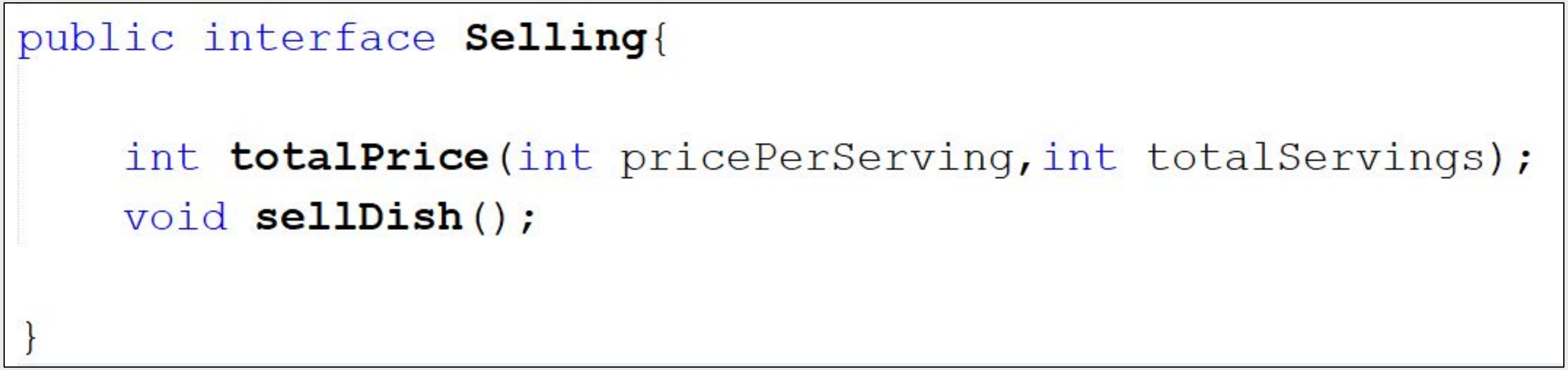


```java
public class Biriyani implements Selling {

    public int totalPrice(int pricePerServing, int totalServings) {
        return pricePerServing * totalServings;
    }

    public void sellDish() {
        System.out.println("\nCalculating Total Price...");
        System.out.println("Total price is: " + totalPrice(200, 4));
    }

    public String arrangeIngredients() {

        return "Rice , Meat, Spice, Curd";
    }
```

Now if I add another game where we have ready made candy, then will the previously designed Selling interface work?

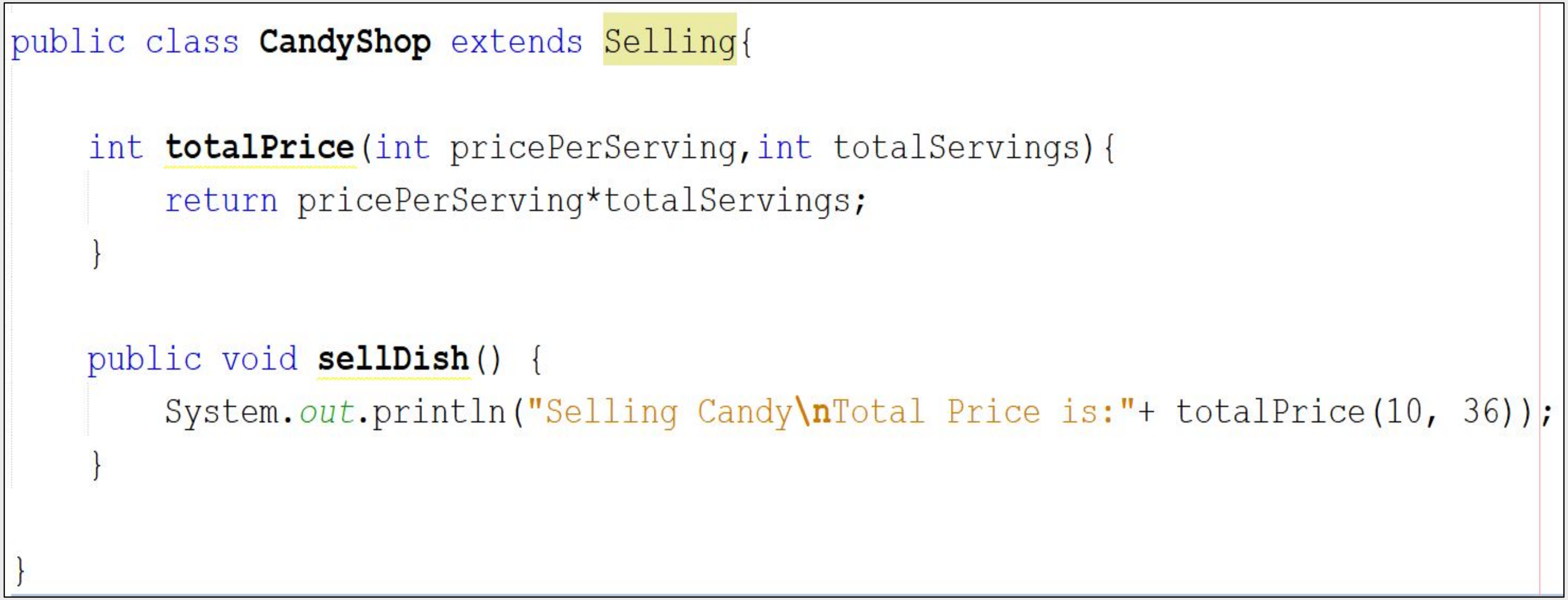
No, it will still force us to assemble and prepare ingredients. So this is a design flaw. Then we can simply delete the extends part.

Deleting the extends part and keeping it simple.



```java
public interface Selling{

    int totalPrice(int pricePerServing,int totalServings);
    void sellDish();

}
```

CandyShop can now implement only Selling



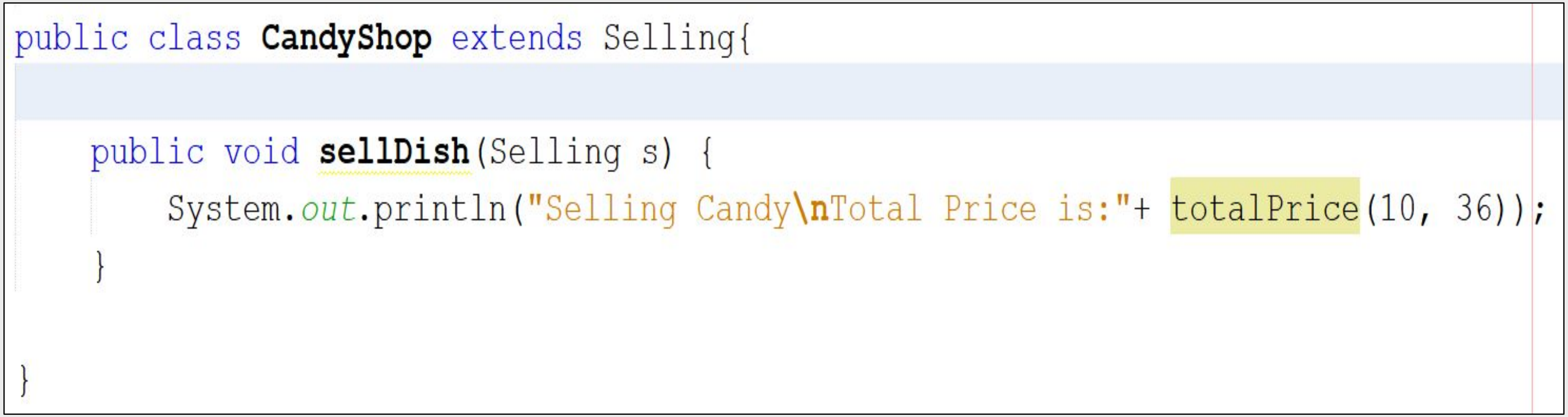
```java
public class CandyShop extends Selling{

    int totalPrice(int pricePerServing,int totalServings){
        return pricePerServing*totalServings;
    }

    public void sellDish() {
        System.out.println("Selling Candy\nTotal Price is:"+ totalPrice(10, 36));
    }


}
```

Did you notice that the totalPrice() method always return the same formula?

If so then wouldn’t it be better if we implement it only once?

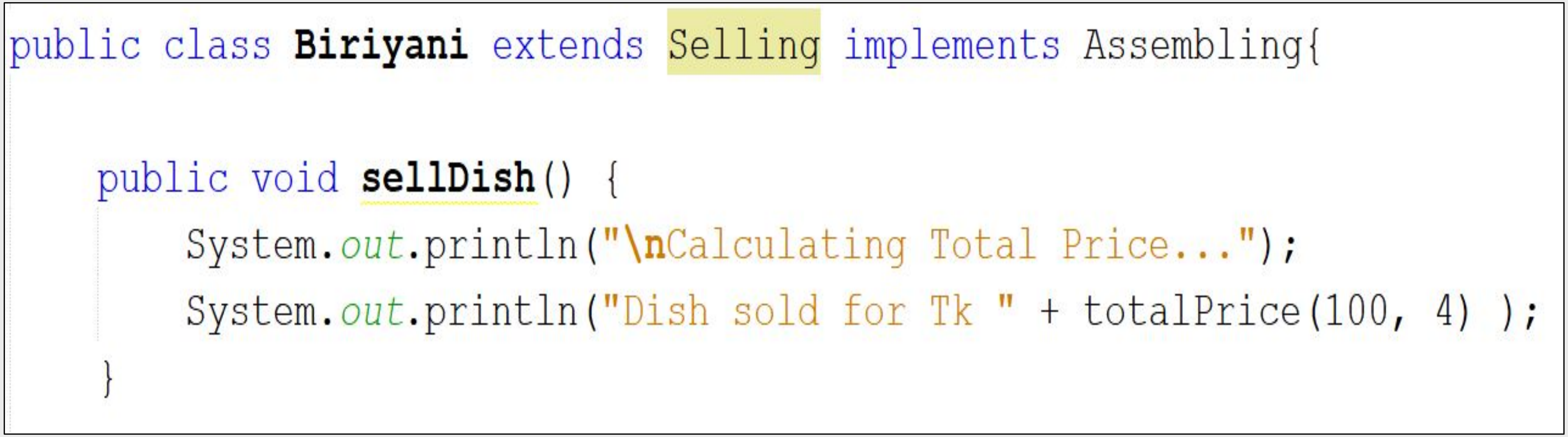
But Selling also has an unimplemented method. What can we do in this scenario?

We can simply convert Selling interface to an Abstract class.



```
public class CandyShop extends Selling{


    public void sellDish(Selling s) {
        System.out.println("Selling Candy\nTotal Price is:"+ totalPrice(10, 36));
    }


}
```

And the Biriyani Class becomes:



```
public class Biriyani extends Selling implements Assembling{


    public void sellDish() {
        System.out.println("\nCalculating Total Price...");
        System.out.println("Dish sold for Tk " + totalPrice(100, 4) );
    }
```