FastFood

Exercise

You have been tasked with writing an ordering system for a fast food restaurant. To do this, you must write the following three Java classes:

1. Create a public class `Food`:

Constructor: Food(String, int)
Fields: name, price
Methods: getPrice(), display()

- The fields `name` and `price` must both be `protected`
- The constructor must set the fields
- `getPrice()` must return the price
- The `display()` method must print `price + " kr " + name`.

2. Create a subclass of `Food` called `Pizza`:

Constructor: Pizza()
Fields: toppings
Methods: ..., addTopping(String), setName(String)

- The field `toppings` must be a `List` of `String`, and it must be `protected`
- The constructor should not take any arguments. The constructor of the super class must be given the arguments `"Pizza"` and `45` (dkk). The list of toppings must be empty for new Pizza objects.
- The `addTopping(String topping)` method adds to the list `toppings`, and increases the price by 10 dkk.
- The class must also override the `display()` method of the super class to also print the toppings comma separated with curly brackets around. E.g., a Pizza with the three toppings `"Pepperoni"`, `"Pineapple"`, and `"Onion"` should be printed as

  ```
  75 kr Pizza { Pepperoni, Pineapple, Onion }
  ```

  or the empty pizza as

  ```
  45 kr Pizza { }
  ```

  - Hint: the following code might give you a hint how to only have comma between and not before or after:

    ```
    String[] message = { "hi", "how", "are", "you" };
    String separator = "";  // first, no space
    for (String word : message) {
        System.out.print(separator);
        System.out.print(word);
        separator = " ";    // after first iteration, a space between the words
    }
    ```
- `setName(String name)` must set the name of the pizza through the name field of Food.

3. Create a class `Order`, representing an order of several food items:

Constructor: Order()
Fields: ordered
Methods: total(), display(), payWith(CreditCard)

- The field `ordered` must be a `List` of `Food`, and it must be `protected`.
- The constructor must leave the list of `ordered` food empty
- The `addFood(Food food)` must add the given food to the list `ordered`
- The `total()` method must return the sum of prices for all food items in the order.
- The `display()` method must print out all the food ordered, followed by the total for the order. The output must follow this format:

  ```
  65 kr Burger
  25 kr Coke
  75 kr Pizza { Pepperoni, Pineapple, Onion }
  165 kr TOTAL
  ```
- `CreditCard` is an interface, which is attached at the bottom of the problem description. You do not need to upload it with your solution, but if you copy it into your source folder, your code can compile locally.
- The `payWith(CreditCard card)` must return a boolean. It must attempt to `withdraw` the current total of the order from the credit card. `payWith` should return the same value that `withdraw` returns, but in case it is false, it must also print `"ERROR: Payment failed"`.

Reminder: It is required to submit the submission Id in LearnIT before the deadline.

Show **sample test data**

Attached Files: CreditCard.java

Submit Solution

Language	Java (15)
Comment	
	Files / Source Code
Files	Click to select files or drop files here
	Submit