Singleton Pattern

Software Design

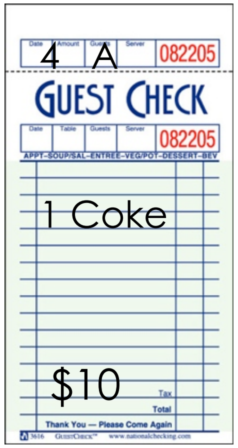
Check please!

- In a very busy restaurant you can be attended by several different waiters
- This restaurant has implemented a process to always assign the orders to a table number
- At the end, they take all of the orders for the same table and create the check

A dinner in this restaurant

- Lupita and Perico seat in table 4
- Lupita is thirsty and orders a coke immediately to ‘Waiter A’, Perico has not yet decided what to drink
- ‘Waiter B’ brings Lupita’s coke and Perico orders a coke as well
- ‘Waiter C’ bring Perico’s coke and Perico orders 2 enchiladas to start, Lupita is still thinking
- Later, Lupita orders a chicken pozole to ‘Waiter B’
- Perico gets his enchiladas and orders a ‘tostada de pata’ to ‘Waiter D’
- Lupita gets her pozole, and Perico his tostada, and now Perico wants a Quesadilla de chicharron. Waiter A takes the order
- Finally Lupita and Perico ask for the check

Orders on table 4



Date Amount Guests Server 4 A 082205

GUEST CHECK

Date Table Guests Server 082205

APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

1 Coke

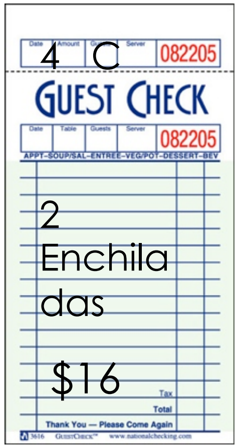
$10

Tax

Total

Thank You — Please Come Again

3616 GUESTCHECK™ www.nationalchecking.com



Date Amount Guests Server 4 C 082205

GUEST CHECK

Date Table Guests Server 082205

APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

2 Enchiladas

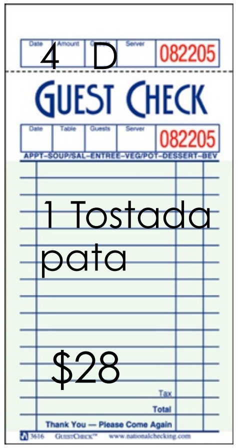
$16

Tax

Total

Thank You — Please Come Again

3616 GUESTCHECK™ www.nationalchecking.com



Date Amount Guests Server 4 D 082205

GUEST CHECK

Date Table Guests Server 082205

APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

1 Tostada pata

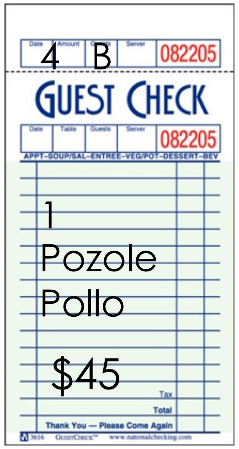
$28

Tax

Total

Thank You — Please Come Again

3616 GUESTCHECK™ www.nationalchecking.com



Date Amount Guests Server 4 B 082205

GUEST CHECK

Date Table Guests Server 082205

APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

1 Pozole Pollo

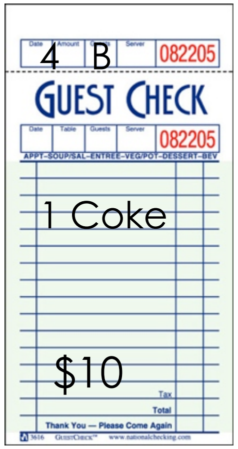
$45

Tax

Total

Thank You — Please Come Again

3616 GUESTCHECK™ www.nationalchecking.com



Date Amount Guests Server 4 B 082205

GUEST CHECK

Date Table Guests Server 082205

APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

1 Coke

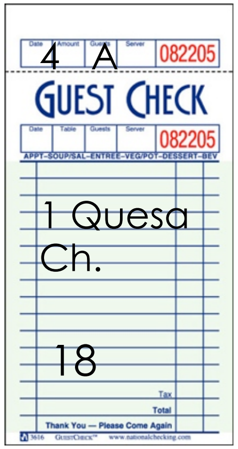
$10

Tax

Total

Thank You — Please Come Again

3616 GUESTCHECK™ www.nationalchecking.com



Date Amount Guests Server 4 A 082205

GUEST CHECK

Date Table Guests Server 082205

APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

1 Quesa Ch.

18

Tax

Total

Thank You — Please Come Again

3616 GUESTCHECK™ www.nationalchecking.com

Imagine this situation in code

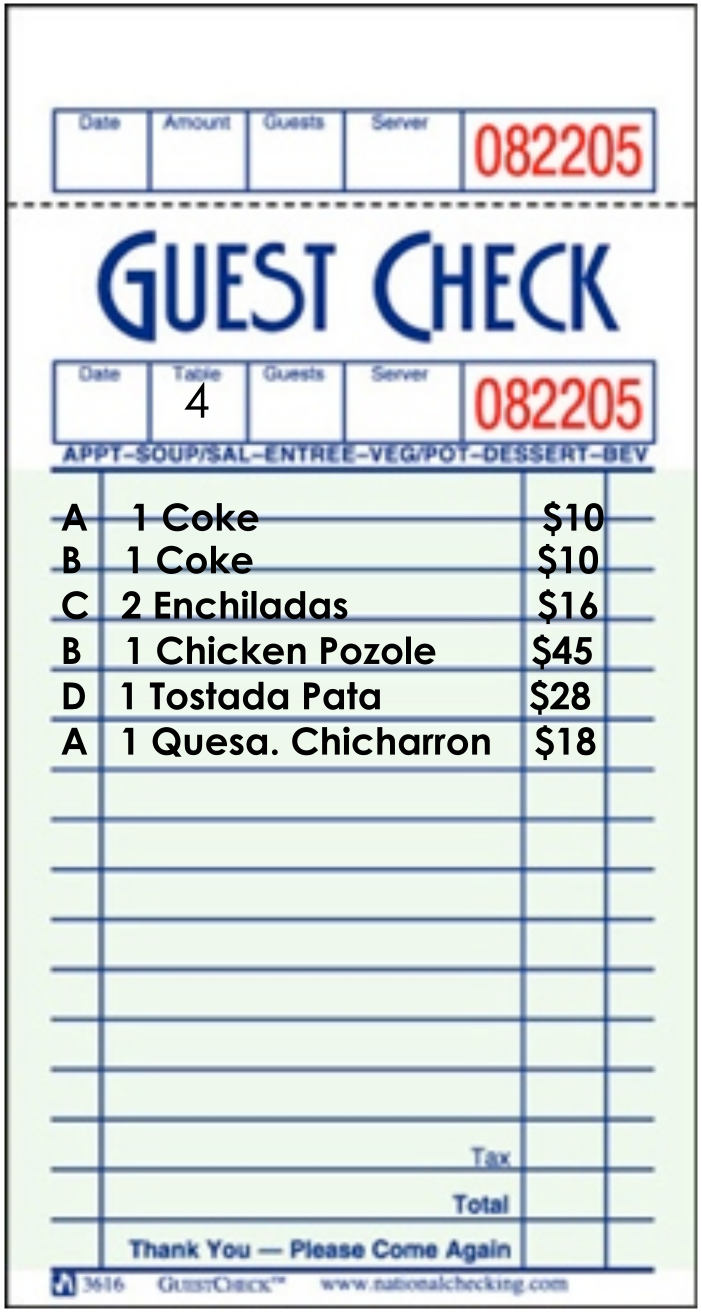
```java
TableOrder order1 = new TableOrder();
order1.setTable(4);
order1.setWaiter("Waiter A");
Drink d1 = new Drink();
d1.setDescription("Coke");
order1.addDrink(d1);

TableOrder order2 = new TableOrder();
order2.setTable(4);
order2.setWaiter("Waiter B");
Drink d2 = new Drink();
d2.setDescription("Coke");
order2.addDrink(d2);

TableOrder order3 = new TableOrder();
order3.setTable(4);
order3.setWaiter("Waiter C");
Dish ds1 = new Dish();
ds1.setDescription("Enchilada");
Dish ds2 = new Dish();
ds2.setDescription("Enchilada");
order1.addDish(ds1);
order1.addDish(ds2);
```

Analyzing situation

- We create a new TableOrder every time a customer asks for something
- We can lose or misplace one of those TabeOrders
- Creating the final check could be a nightmare

Solution

- Add all orders to a single TableOrder
- Ensure we have only one TableOrder per customer



Date	Amount	Guests	Server	082205

GUEST CHECK

Date	Table	Guests	Server	082205
	4			

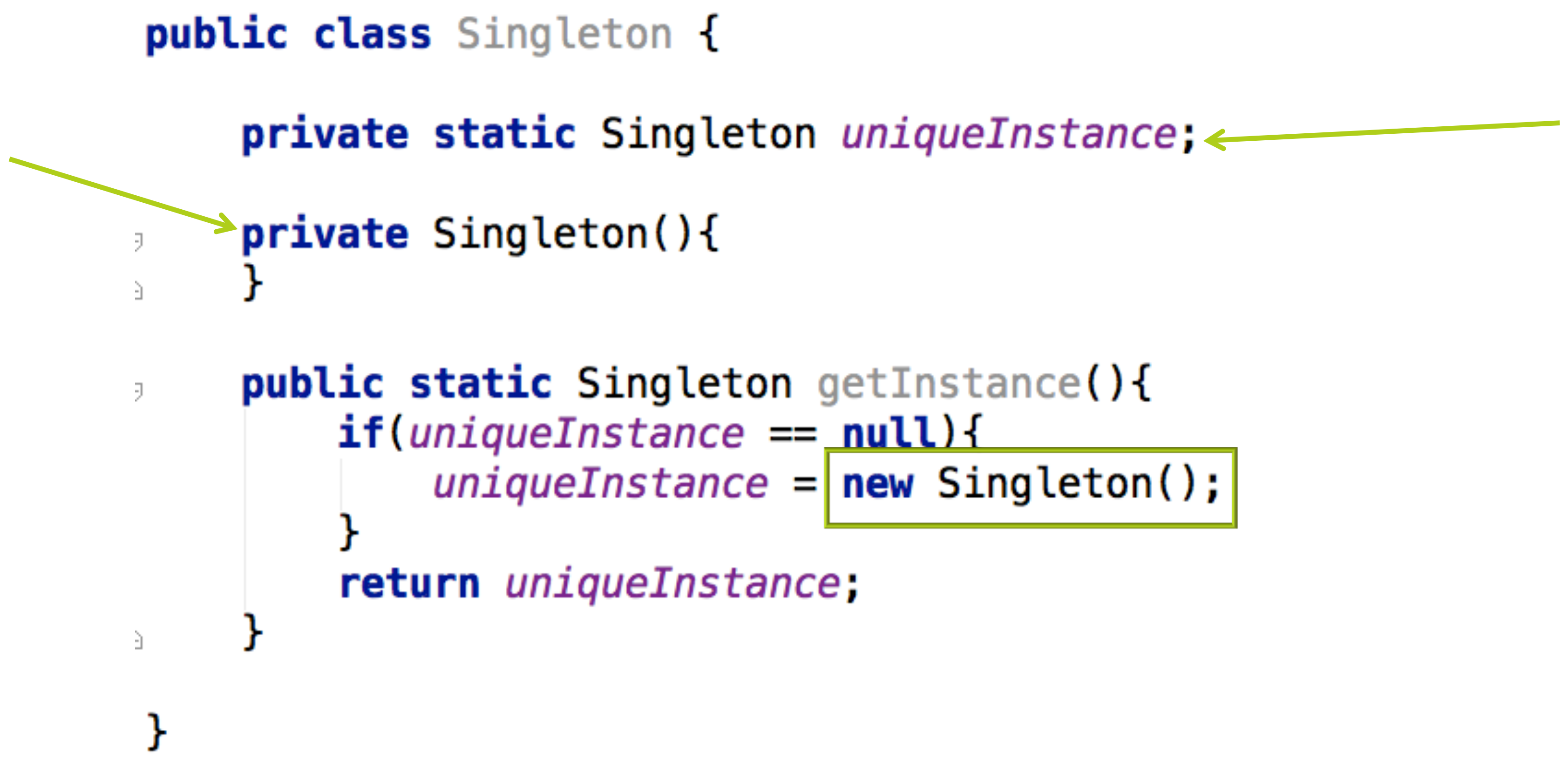
APPT–SOUP/SAL–ENTREE–VEG/POT–DESSERT–BEV

A	1 Coke	$10
B	1 Coke	$10
C	2 Enchiladas	$16
B	1 Chicken Pozole	$45
D	1 Tostada Pata	$28
A	1 Quesa. Chicharron	$18
	Tax	
	Total	

Thank You — Please Come Again

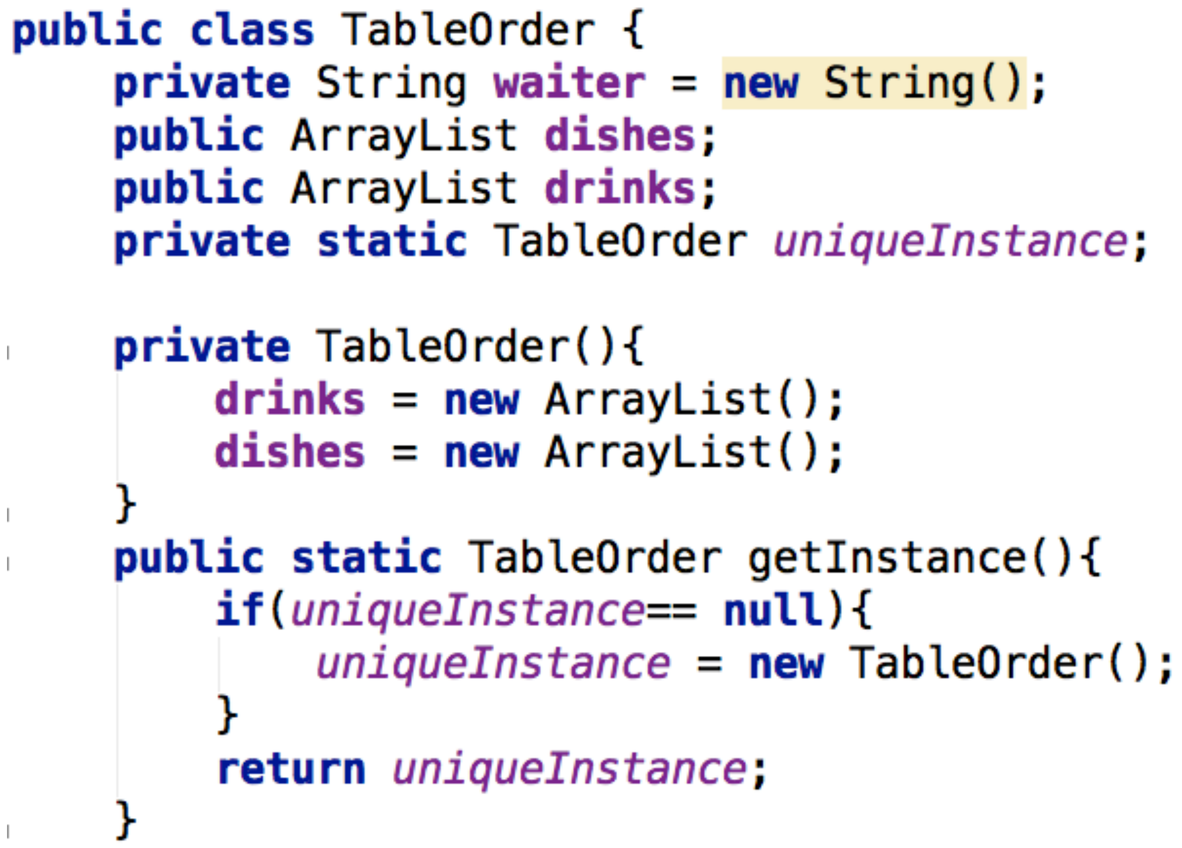
3616 GUESTCHECK™ www.nationalchecking.com

A private Constructor



```
public class Singleton {

    private static Singleton uniqueInstance;

    private Singleton(){
    }

    public static Singleton getInstance(){
        if(uniqueInstance == null){
            uniqueInstance = new Singleton();
        }
        return uniqueInstance;
    }

}
```

In our restaurant



```
public class TableOrder {
    private String waiter = new String();
    public ArrayList dishes;
    public ArrayList drinks;
    private static TableOrder uniqueInstance;

    private TableOrder(){
        drinks = new ArrayList();
        dishes = new ArrayList();
    }
    public static TableOrder getInstance(){
        if(uniqueInstance== null){
            uniqueInstance = new TableOrder();
        }
        return uniqueInstance;
    }
```

Implementation