Strategy (Strategie)

Rozšiřte aplikaci „Smart Café“ o několik možností platby (pro zjednodušení opět stačí vypisovat text do konzole).

Vytvořte abstraktní třídu nebo rozhraní `PaymentStrategy`, kde bude předepsána metoda `pay` se vstupními parametry `amout` a `table` a bude vracet `String`. Následně vytvořte alespoň dvě nové třídy, které z této třídy dědí. Každá třída zároveň přepisuje metodu `pay`.

V případě např. využití kreditní karty, může metoda pay vracet:

„Customer want to pay 400 using Credit Card (table 4)“

Následně vytvořte třídu `Checkout`, která v sobě bude ukládat používanou strategii. Konstruktoru třídy bude předána prvotní strategie placení. Dále implementujte metodu `setStrategy` (pomocí níž půjde měnit strategii placení) a metodu `processPayment`.

Ta bude sloužit pro volání momentálně zvolená strategie:

```
return strategy.pay(amount, table);
```

V hlavní třídě programu si následně vytvořte nový seznam osob, které budou přijímat notifikace o tom, že zákazník chce platit. Do tohoto seznamu přidejte číšníka.

Hlavní třída nyní může vypadat následovně:

```
OrderSubject order = new OrderSubject();

EmployeeObserver waiter = new EmployeeObserver("Waiter");
EmployeeObserver barista = new EmployeeObserver("Barista");

order.addObserver(barista);
order.addObserver(waiter);

CustomDrink customDrink = new
CustomDrink.Builder("coffee").milk().sugar().build();
order.notifyAll(customDrink +" in
"+CafeConfig.getInstance().getCafeName());

OrderSubject payment = new OrderSubject();
payment.addObserver(waiter);

Checkout checkout = new Checkout(new CreditCardPayment());
payment.notifyAll(checkout.processPayment(150, 4));
checkout.setStrategy(new CashPayment());
payment.notifyAll(checkout.processPayment(100, 2));
```

Pozn.: Hotová aplikace je taková, která není otestována pouze Vámi, ale především unit testy.