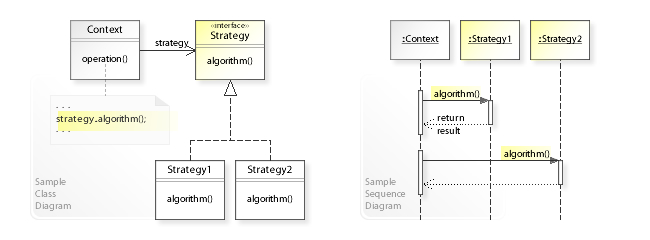
Statergy Pattern

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), the strategy pattern (also known as the policy pattern) is a [behavioral](https://en.wikipedia.org/wiki/Behavioral_design_pattern) [software design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) that enables selecting an [algorithm](https://en.wikipedia.org/wiki/Algorithm) at runtime. Instead of implementing a single algorithm directly, code receives run-time instructions as to which in a family of algorithms to use.



import java.util.ArrayList;

import java.util.List;

interface BillingStrategy {

*// Use a price in cents to avoid floating point round-off error*

int getActPrice(int rawPrice);

*// Normal billing strategy (unchanged price)*

static BillingStrategy normalStrategy() {

return rawPrice -> rawPrice;

}

*// Strategy for Happy hour (50% discount)*

static BillingStrategy happyHourStrategy() {

return rawPrice -> rawPrice / 2;

}

}

class CustomerBill {

private final List<Integer> drinks = new ArrayList<>();

private BillingStrategy strategy;

public CustomerBill(BillingStrategy strategy) {

this.strategy = strategy;

}

public void add(int price, int quantity) {

this.drinks.add(this.strategy.getActPrice(price\*quantity));

}

*// Payment of bill*

public void print() {

int sum = this.drinks.stream().mapToInt(v -> v).sum();

System.out.println("Total due: " + sum);

this.drinks.clear();

}

*// Set Strategy*

public void setStrategy(BillingStrategy strategy) {

this.strategy = strategy;

}

}

public class StrategyPattern {

public static void main(String[] arguments) {

*// Prepare strategies*

BillingStrategy normalStrategy = BillingStrategy.normalStrategy();

BillingStrategy happyHourStrategy = BillingStrategy.happyHourStrategy();

CustomerBill firstCustomer = new CustomerBill(normalStrategy);

*// Normal billing*

firstCustomer.add(100, 1);

*// Start Happy Hour*

firstCustomer.setStrategy(happyHourStrategy);

firstCustomer.add(100, 2);

*// New Customer*

CustomerBill secondCustomer = new CustomerBill(happyHourStrategy);

secondCustomer.add(80, 1);

*// The Customer pays*

firstCustomer.print();

*// End Happy Hour*

secondCustomer.setStrategy(normalStrategy);

secondCustomer.add(130, 2);

secondCustomer.add(250, 1);

secondCustomer.print();

}

}