

# Strategy

دیزاین پترن هادر React

Behavioral Patterns

به زبان ساده همراه با مثال های عملی  
کاربرد دیزاین پترن ها برای برنامه نویسان React



# Behavioral Patterns

## Strategy

الگوی Strategy مجموعه‌ای از الگوریتم‌ها را تعریف می‌کند و این امکان را دهد که الگوریتم‌ها به صورت قابل تعویض و مستقل از استفاده کننده آن‌ها عمل کنند. می‌توان از این الگو برای مدیریت استراتژی‌های مختلف در یک کامپوننت استفاده کرد.

فرض کنید می‌خواهیم یک سیستم محاسبه تخفیف بر اساس حالت‌های زیر را پیاده‌سازی کنیم.

- ثابت: تخفیف در صد مشخصی از کل مبلغ کم می‌شود.
  - مشتریان وفادار: تخفیف به تعداد خریدهای قبلی بستگی دارد.
  - محدود زمانی: تخفیف در یک بازه زمانی خاص اعمال می‌شود.
- هر یک از این سه استراتژی منطق متفاوتی برای محاسبه تخفیف دارد.

# بخش DiscountStrategy

در ابتدا یک Interface به شکل زیر می سازیم.



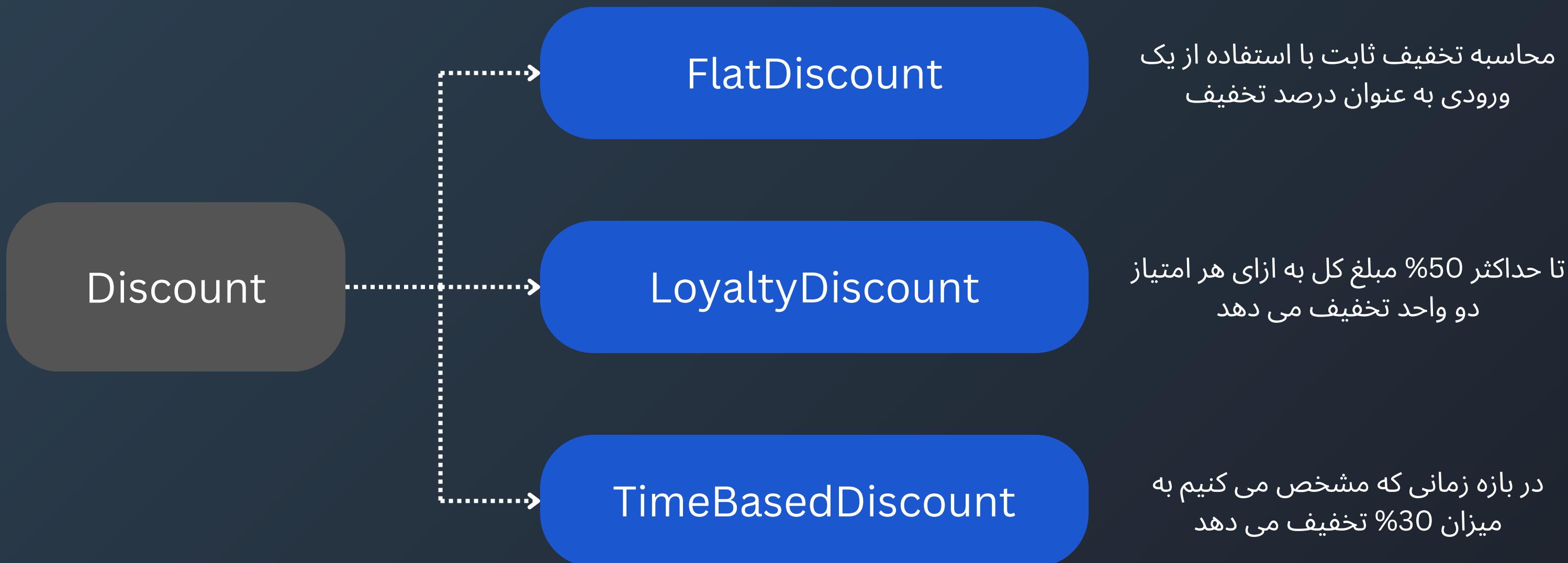
## DiscountStrategy



```
export default interface DiscountStrategy {  
  calculateDiscount(amount: number): number;  
}
```

# منطقه ها Strategy

همانطور که گفتیم سه متفاوت برای تخفیف ها به شکل زیر داریم.



# بخش FlatDiscount

در این Strategy یک ورودی به نام percentage داریم که بر اساس آن تخفیف محاسبه می شود.



```
import DiscountStrategy from '../discount-strategy.interface';

export default class FlatDiscount implements DiscountStrategy {
    private percentage: number;

    constructor(percentage: number) {
        this.percentage = percentage;
    }

    calculateDiscount(amount: number): number {
        return amount * (this.percentage / 100);
    }
}
```

# بخش LoyaltyDiscount

در این Strategy یک ورودی به نام loyaltyPoints داریم که به معنای امتیاز وفاداری می باشد و بر اساس آن و حداکثر تا 50 درصد تخفیف محاسبه می شود.



```
import DiscountStrategy from '../discount-strategy.interface';

export default class LoyaltyDiscount implements DiscountStrategy {
    private loyaltyPoints: number;

    constructor(loyaltyPoints: number) {
        this.loyaltyPoints = loyaltyPoints;
    }

    calculateDiscount(amount: number): number {
        const discount = Math.min(this.loyaltyPoints * 2, amount * 0.5);
        return discount;
    }
}
```

# بخش TimeBasedDiscount

در این Strategy دو ورودی به نام های startTime و endTime داریم که فقط در این بازه زمانی و تا سقف 30 درصد تخفیف محاسبه می شود.



## TimeBasedDiscount

```
import DiscountStrategy from '../discount-strategy.interface';

export default class LoyaltyDiscount implements DiscountStrategy {
  private loyaltyPoints: number;

  constructor(loyaltyPoints: number) {
    this.loyaltyPoints = loyaltyPoints;
  }

  calculateDiscount(amount: number): number {
    const discount = Math.min(this.loyaltyPoints * 2, amount * 0.5);
    return discount;
  }
}
```

# بخش DiscountCalculator

در این بخش یک Component به شکل زیر می سازیم و کاربرد می تواند یکی از سه تخفیف موجود را انتخاب نماید.



```
DiscountCalculator component code
```

```
import { useState } from 'react';
import DiscountStrategy from './pattern/discount-strategy.interface';
import FlatDiscount from './patternestrategies/flat-discount.strategy';
import LoyaltyDiscount from './patternestrategies/loyalty-discount.strategy';
import TimeBasedDiscount from './patternestrategies/time-based-discount.strategy';

interface DiscountCalculatorProps {
  amount: number;
}

const DiscountCalculator: React.FC<DiscountCalculatorProps> = ({ amount }) => {
  const [strategy, setStrategy] = useState<DiscountStrategy>(
    new FlatDiscount(10)
  );
  const [discount, setDiscount] = useState<number>(
    strategy.calculateDiscount(amount)
  );

  const changeStrategy = (newStrategy: DiscountStrategy) => {
    setStrategy(newStrategy);
    setDiscount(newStrategy.calculateDiscount(amount));
  };

  return (
    <div>
      <h2>Discount Calculator</h2>
      <p>Total Amount: {amount} units</p>
      <button onClick={() => changeStrategy(new FlatDiscount(10))}>
        Flat Discount (10%)
      </button>
      <button onClick={() => changeStrategy(new LoyaltyDiscount(180))}>
        Loyalty Discount (180 Points)
      </button>
      <button onClick={() => changeStrategy(
        new TimeBasedDiscount(
          new Date('2024-08-02'),
          new Date('2028-10-24')
        )
      )}>
        Time-Based Discount
      </button>
    </div>
  );
}

export default DiscountCalculator;
```

```
DiscountCalculator component code
```

```
<div>
  <h2>Discount Calculator</h2>
  <p>Total Amount: {amount} units</p>
  <div>
    <button onClick={() => changeStrategy(new FlatDiscount(10))}>
      Flat Discount (10%)
    </button>
    <button onClick={() => changeStrategy(new LoyaltyDiscount(180))}>
      Loyalty Discount (180 Points)
    </button>
    <button onClick={() => changeStrategy(
      new TimeBasedDiscount(
        new Date('2024-08-02'),
        new Date('2028-10-24')
      )
    )}>
      Time-Based Discount
    </button>
  </div>
  <div>
    <h3>Applied Discount:</h3>
    <p>{discount.toFixed(2)} units</p>
  </div>
  <div>
    <h3>Final Amount:</h3>
    <p>{(amount - discount).toFixed(2)} units</p>
  </div>
</div>
```

# طرز استفاده از DiscountCalculator

در پایان به شکل زیر از DiscountCalculator استفاده می کنیم.



## StrategyDesignPattern



```
import DiscountCalculator from './DiscountCalculator';

const StrategyDesignPattern = () => {
  return (
    <div>
      <DiscountCalculator amount={1000} />
    </div>
  );
};

export default StrategyDesignPattern;
```