수시고사 대비

전략 패턴 다이어그램

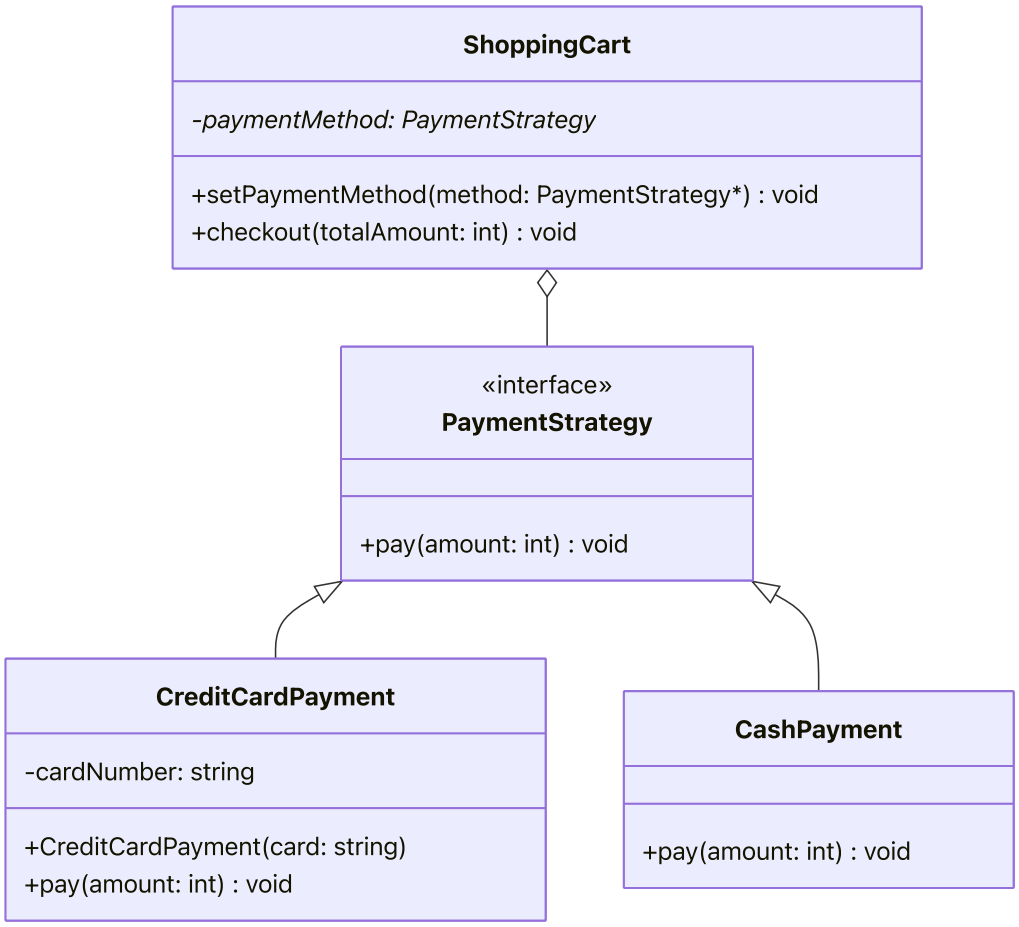
```
// 전략 인터페이스
class PaymentStrategy {
public:
    virtual ~PaymentStrategy() {}
    virtual void pay(int amount) = 0;
};

// 구체적 전략 1
class CreditCardPayment : public PaymentStrategy {
private:
    std::string cardNumber;
public:
    CreditCardPayment(const std::string& card) : cardNumber(card) {}
    void pay(int amount) override {
        std::cout << amount << "원을 카드 " << cardNumber << "로 결제" << std::endl;
    }
};

// 구체적 전략 2
class CashPayment : public PaymentStrategy {
public:
    void pay(int amount) override {
        std::cout << amount << "원을 현금으로 결제" << std::endl;
    }
};

// 컨텍스트
class ShoppingCart {
private:
    PaymentStrategy* paymentMethod;
public:
    void setPaymentMethod(PaymentStrategy* method) {
        paymentMethod = method;
    }

    void checkout(int totalAmount) {
        paymentMethod->pay(totalAmount);
    }
};
```



팩토리 메서드 패턴 다이어그램

```
// 제품 인터페이스
class Logger {
public:
    virtual ~Logger() {}
    virtual void log(const std::string& message) = 0;
};

// 구체적 제품 1
class FileLogger : public Logger {
public:
    void log(const std::string& message) override {
        std::cout << "파일에 로깅: " << message << std::endl;
    }
};

// 구체적 제품 2
class ConsoleLogger : public Logger {
public:
    void log(const std::string& message) override {
        std::cout << "콘솔에 로깅: " << message << std::endl;
    }
};

// 생성자 인터페이스
class LoggerFactory {
public:
    virtual ~LoggerFactory() {}
    virtual Logger* createLogger() = 0;
};
```

```cpp
// 구체적 생성자 1
class FileLoggerFactory : public LoggerFactory {
public:
    Logger* createLogger() override {
        return new FileLogger();
    }
};

// 구체적 생성자 2
class ConsoleLoggerFactory : public LoggerFactory {
public:
    Logger* createLogger() override {
        return new ConsoleLogger();
    }
};

// 클라이언트 코드
void application(LoggerFactory* factory) {
    Logger* logger = factory->createLogger();
    logger->log("애플리케이션 시작");
    //  ...
    delete logger;
}
```