```
/**
 * In an e-commerce application, different payment methods are accepted
 * (e.g., credit card, PayPal, and cryptocurrency).
 * As the business expands, new payment methods may be introduced,
 * and existing ones may undergo changes.
 * The system should be designed to accommodate these changes
 * without modifying the existing codebase.
 *
 * Solve this problem using Strategy Pattern with the help of following starter code.
 */

// Define the PaymentStrategy interface
interface PaymentStrategy {
    void processPayment(double amount);
}

// Concrete implementations for different payment methods

class CreditCardPayment implements PaymentStrategy {
    @Override
    public void processPayment(double amount) {
        System.out.println("Processing credit card payment of BDT " + amount + "/=
Taka");
    }
}

class PayPalPayment implements PaymentStrategy {
    @Override
    public void processPayment(double amount) {
        System.out.println("Processing PayPal payment of BDT " + amount + "/= Taka");
    }
}

class CryptocurrencyPayment implements PaymentStrategy {
    @Override
    public void processPayment(double amount) {
        System.out.println("Processing cryptocurrency payment of BDT " + amount + "/=
Taka");
    }
}

// Add new class for Bkash Payment

// Context class that uses a payment strategy
class ShoppingCart {
    // add member variables and functions (if necessary)

    public void checkout(double totalAmount) {
        // add code to implement this method
    }
}

public class OnlineShopping {
    public static void main(String[] args) {
        ShoppingCart cart1 = new ShoppingCart();
        ShoppingCart cart2 = new ShoppingCart();

        // add code here (if necessary)

        cart1.checkout(100.0); // Output: Processing Bkash payment of BDT 100.0/= Taka
        cart2.checkout(50.0);  // Output: Processing PayPal payment of BDT 50.0/= Taka
    }
}

// Note: You are not allowed to change other part of this code.
```