**Implementación del Patrón de Estrategia**

Utilizando el patrón de estrategia, podemos encapsular la lógica de cada método de pago en sus propias clases, de tal manera que el sistema principal se comunique con ellas a través de una interfaz común. Esto nos permitirá:

* **Separación de responsabilidades**: Cada método de pago se encargará de su propia lógica sin interferir con otros.
* **Facilidad de extensión**: Para agregar un nuevo método de pago, simplemente se implementa una nueva clase que siga la interfaz común, sin modificar el código existente.
* **Consistencia en la interfaz de usuario**: La tienda puede ofrecer una experiencia uniforme al cliente al interactuar con la interfaz común.

**Clases que Interactúan en el Sistema**

1. **PaymentProcessor**: Clase responsable de gestionar el flujo de pago y seleccionar la estrategia de pago.
2. **PaymentStrategy**: Interfaz común para todas las estrategias de pago.
3. **CreditCardPayment**: Clase que implementa el proceso de pago con tarjeta de crédito.
4. **CryptocurrencyPayment**: Clase que implementa el proceso de pago con criptomonedas.
5. **BankTransferPayment**: Clase que implementa el proceso de pago mediante transferencia bancaria.
6. **PaymentDetails**: Clase que representa la información básica del pago (como el monto y el método de pago).

**Diagrama de clases**

