El uso de Common Reuse Principle o CRP a nivel de principios de ingeniería de Software es muy habitual cuando nos encontramos trabajando con Packages . La mayor parte de las veces cuando hablamos de Principios de Ingeniería de Software estos están muy ligados al manejo de clases e interfaces pero no al manejo de packages . Sin embargo existen principios que se aplican a estos elementos. Uno de ellos es el Principio CRP o *Principio Commun de Reutilización q*ue hace referencia a cómo debemos organizar las clases dentro de nuestros packages .

Common Reuse Principle (CRP)

El Principio en este caso dice

Las clases que se vayan a usar juntas deben empaquetarse juntas

Es algo muy sencillo ,vamos a ver un ejemplo de este principio.Imaginemonos que tenemos el interface Pago que tiene un método importePago()

```
package com.arquitecturajava.pagos;

public interface Pago {

  public void importePago(double importe);
}
```

Es un ejemplo trivial . Para ese interface tenemos tres implementaciones diferentes pago por Paypal , pago por Visa y pago por Cuenta Corriente

```
package com.arquitecturajava.pagos;

public class PagoCuenta implements Pago {

  public void importePago(double importe) {
```

```
    System.out.println("el pago por cuenta es :" +importe);
  }

}
```

```
package com.arquitecturajava.pagos;

public class PagoPaypal implements Pago {

  public void importePago(double importe) {
    System.out.println("El pago con paypal es:"+ importe);
  }

}
```

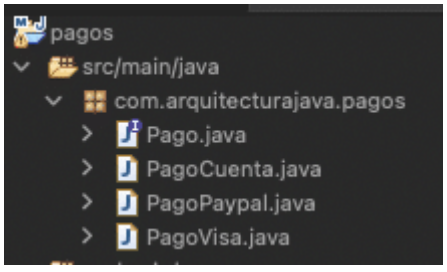
```
package com.arquitecturajava.pagos;

public class PagoVisa implements Pago {

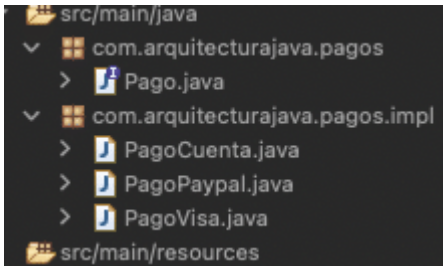
  public void importePago(double importe) {
    System.out.println("el pago con visa es :" +importe);
  }

}
```

En estos momentos todas las clases se encuentra ubicadas en el mismo package:

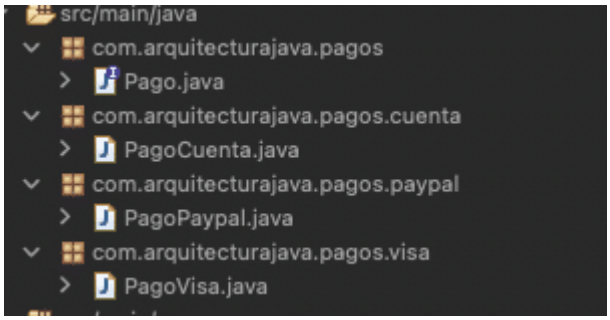


¿Es esto lo correcto? . Bueno la realidad es que depende mucho de como nuestro programa vaya a funcionar . Suele ser bastante habitual separar interfaces de alto nivel de su implementación . Así que podriamos encontrarnos con un nuevo diseño en el cual :

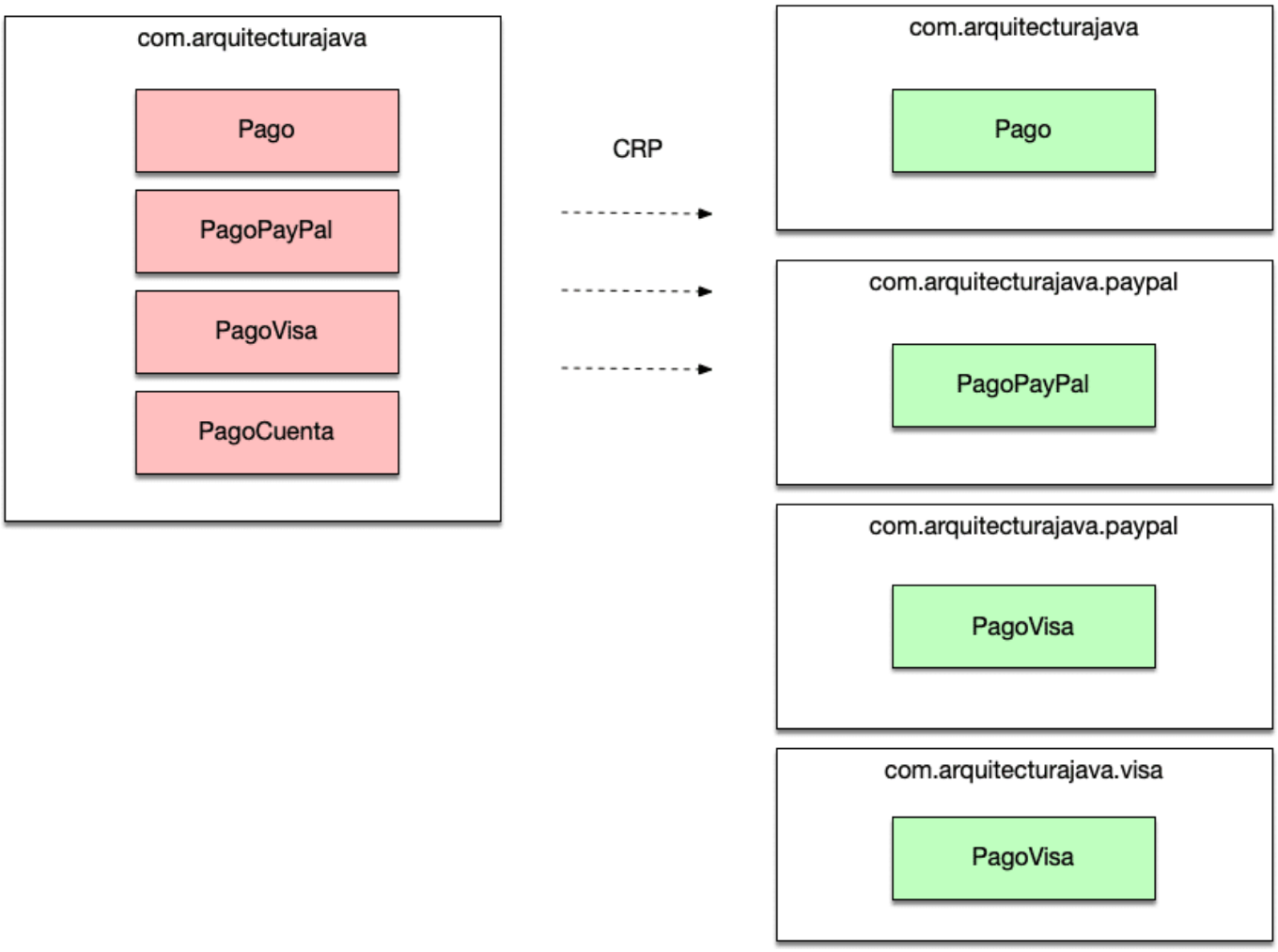


¿Cómo organizarnos?

La pregunta clave en nuestro caso es : ¿ Deben de ir todas las implementaciones de los pagos juntas? . En principio parece que sí ya que por el principio CRP todas las clases que se usen juntas deben empaquetarse juntas. Lamentablemente no es tan sencillo ya que el principio también se puede entender que todas las clases que se usen por separado deben ir en paquetes separados. En nuestro caso la pregunta es: ¿ Cuando a nivel de librerías usamos los métodos de pago se usan por separado PayPal , Visa y Cuenta corriente?. En caso de que por ejemplo la respuesta sea que sí que son 3 implementaciones excluyentes entre sí o se paga con PayPal , con Visa o con Cuenta corriente . En este caso cada una de las clases debería ir asociada a su propio package para facilitar más adelante los despliegues de forma individual.



Tengámoslo siempre en cuenta cuando desarrollemos.



Otros artículos relacionados

1. Command Pattern en Java y la gestion de tareas
2. El patrón MVC , arquitectura cliente vs servidor
3. JPA Proxy y su funcionamiento
4. Java Packages