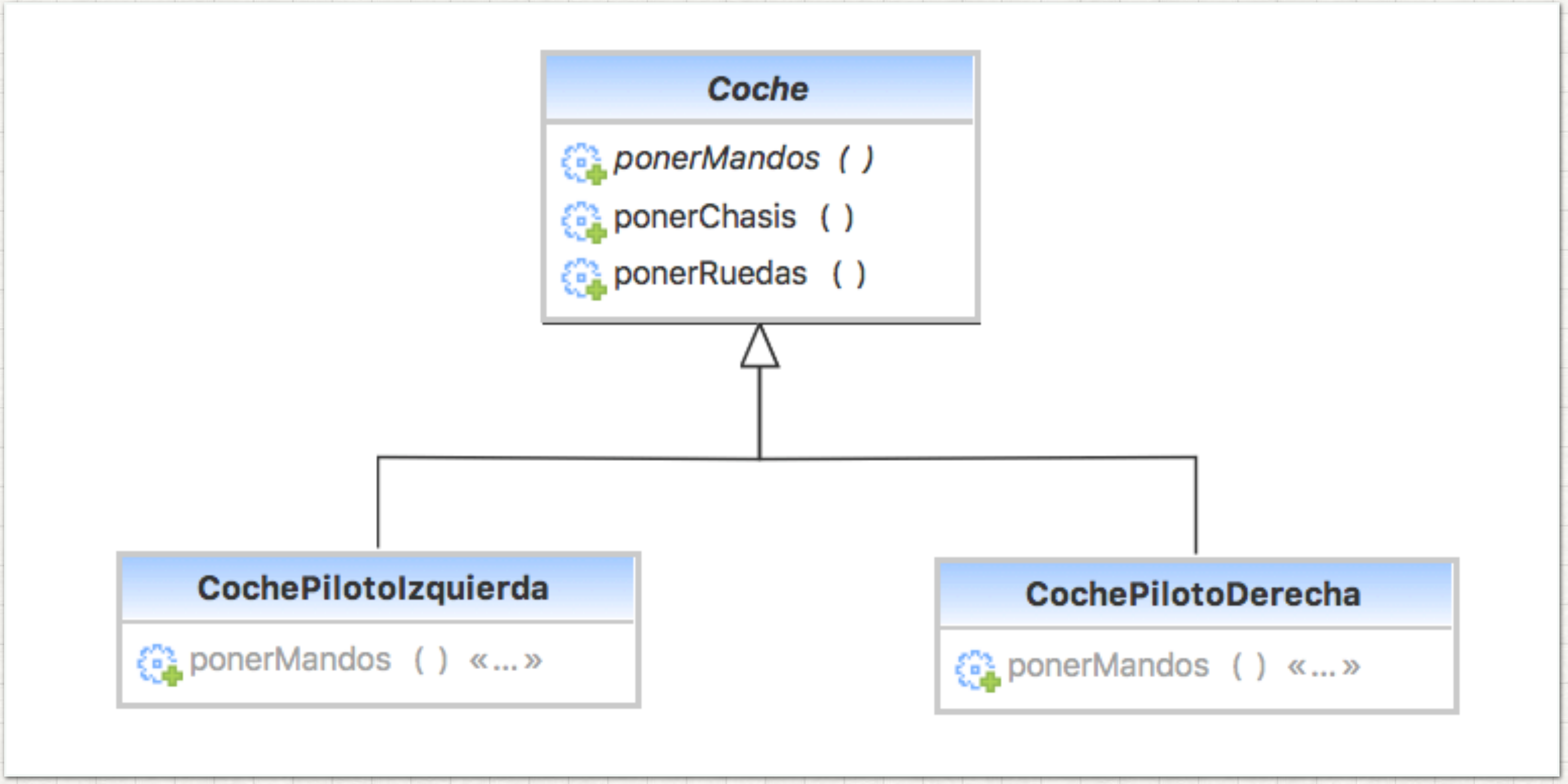
Factory method pattern

Disclaimer: no tengo ni idea de coches !   ;-)



```java
    // estamos en inglaterra. Sino usar piloto izquierdo
    Coche coche = new CochePilotoDerecha();

    coche.ponerChasis();
    coche.ponerMandos();
    coche.ponerRuedas();

    // usar coche
```

Problema: como encapsular la creación e inicialización del objecto, si este puede tener distintos tipos

“

Define an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation it uses to subclasses.

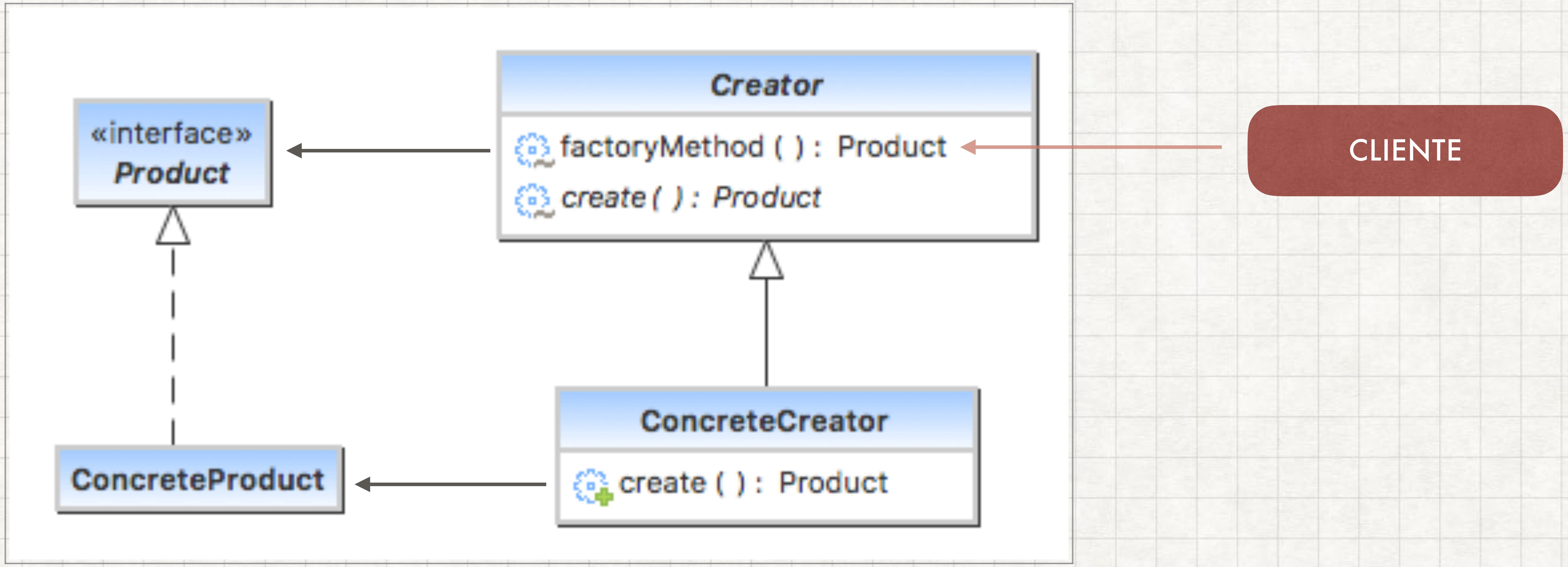
— Gang of Four

”

```java
public abstract class CocheFactory {

  public Coche crea () {
    Coche coche = crea();

    coche.ponerChasis();
    coche.ponerMandos();
    coche.ponerRuedas();

    return coche;
  }

  protected abstract Coche creaCoche();
}
```

```java
public class CocheInglesFactory extends CocheFactory {

  protected Coche creaCoche() {
    return new CochePilotoDerecha();
  }
}
```

```java
CocheFactory factory = new CocheInglesFactory();
Coche coche = factory.crea();
```


«interface»
Product
Creator
factoryMethod ( ) : Product
create ( ) : Product
ConcreteProduct
ConcreteCreator
create ( ) : Product
CLIENTE

```java
public abstract class MazeGame {
    private final List<Room> rooms = new ArrayList<>();

    public MazeGame() {
        Room room1 = makeRoom();
        Room room2 = makeRoom();
        room1.connect(room2);
        rooms.add(room1);
        rooms.add(room2);
    }

    abstract protected Room makeRoom();
}
```

```java
public class MagicMazeGame extends MazeGame {
    @Override
    protected Room makeRoom() {
        return new MagicRoom();
    }
}

public class OrdinaryMazeGame extends MazeGame {
    @Override
    protected Room makeRoom() {
        return new OrdinaryRoom();
    }
}

MazeGame ordinaryGame = new OrdinaryMazeGame();
MazeGame magicGame = new MagicMazeGame();
```

Factory method pattern