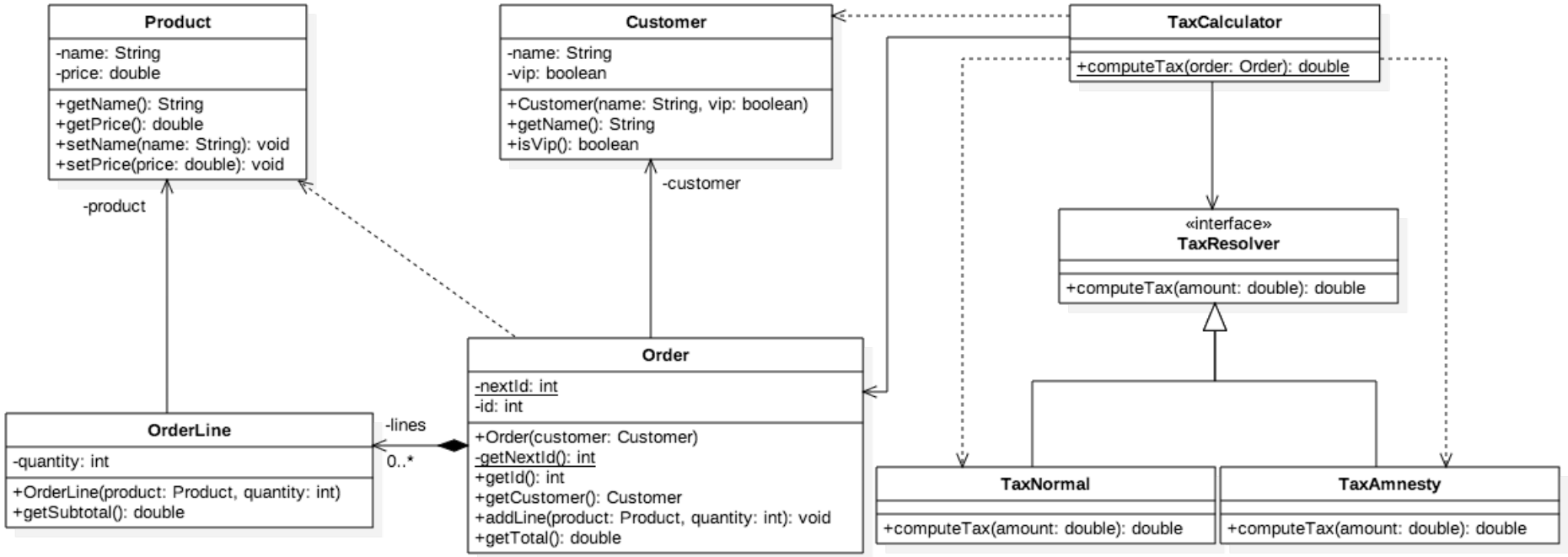
Diseño de Sistemas Software – Solución examen julio 2018

Ejercicio 1

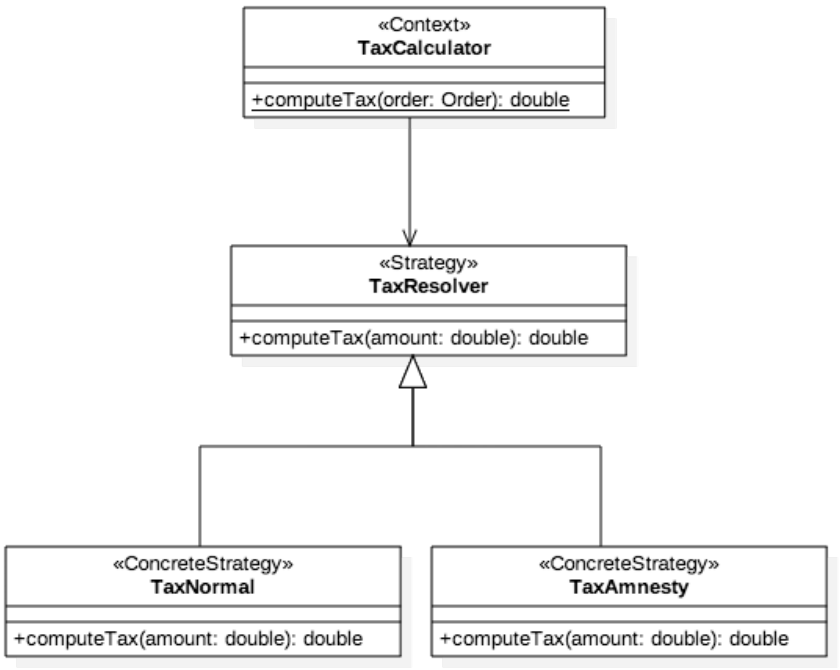


Diseño de Sistemas Software – Solución examen julio 2018

- Es importante reflejar todas las dependencias entre las clases del diagrama.
- Los métodos estáticos se indican con subrayado.

Ejercicio 2

El patrón empleado en el código es **Strategy**. La clase TaxCalculator (Context) necesita calcular los impuestos de distintas formas dependiendo del tipo de cliente (Customer). Para esto delega en las clases TaxNormal y TaxAmnesty (ConcreteStrategies) que implementan el interfaz TaxResolver (Strategy).



Ejercicio 3

La solución del problema requiere poder combinar distintos impuestos: el impuesto normal (21%), impuesto del agua (10%) e impuesto del pan (15%). La solución propuesta debe permitir hacerlo sin crear una clase para cada posible combinación, y sin afectar a la clase TaxCalculator, por lo que ésta debe recibir un único objeto que implemente el interfaz TaxResolver.

El patrón más adecuado para este problema es **Composite**, que permitiría crear un impuesto compuesto a partir de los impuestos detallados arriba, manteniendo el mismo interfaz que los impuestos sencillos. En la solución propuesta se añaden dos clases simples TaxWater y TaxBread y una clase compuesta TaxComposite que permite agregar distintos impuestos para calcular el valor acumulado de todos ellos.

```
public class TaxWater implements TaxResolver {
      public double computeTax(double amount) {
            return amount * 0.10;
      }
}

public class TaxBread implements TaxResolver {
      public double computeTax(double amount) {
            return amount * 0.15;
      }
}

public class TaxComposite implements TaxResolver {
      private ArrayList<TaxResolver> taxes;

      public void addTax(TaxResolver tax) {
            taxes.add(tax);
      }

      public double computeTax(double amount) {
            double taxAmount = 0.0;

            for (TaxResolver tax : taxes) {
                  taxAmount += tax.computeTax(amount);
            }

            return taxAmount;
      }

}
```

Con este nuevo diseño, el código cliente en TaxInspector quedaría así:

```
public class TaxInspector {
    public TaxResolver getTax(Order order) {
        TaxResolver tax;
        if (order.getCustomer().isVip())
            tax = new TaxAmnesty();
        else {
            TaxComposite taxComposite = new TaxComposite();
            taxComposite.addTax(new TaxNormal());
            if (order.containsWater()) taxComposite.addTax(new TaxWater());
            if (order.containsBread()) taxComposite.addTax(new TaxBread());
            tax = taxComposite;
        }
        return tax;
    }
}
```

Diseño de Sistemas Software – Solución examen julio 2018