Programación 2 - TUDAI

Búsquedas

Farmacia

Una farmacia desea poder buscar aquellos medicamentos que son producidos por el laboratorio Bayer.

Cada medicamento posee un nombre, el laboratorio que lo produce y un precio.

Constantes en código no!! 🚩🚩🚩

Farmacia

- Buscar medicamentos de otro laboratorio!
- Buscar medicamentos por nombre
- Buscar medicamento por precio

Farmacia

- Como la búsqueda se realiza sobre los medicamentos almacenados en la clase Farmacia, ubicamos en ella los métodos de búsqueda

Farmacia

- Si observamos el código de los métodos buscarMedicamentoX... vemos que gran parte está repetido 🚩🚩🚩

```
public ArrayList<Medicamento> buscarMedicamentosXLab(String lab) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (medicamento.getLaboratorio().equals(lab))
      lista.add(medicamento);
  }
  return lista;
}
```

```
public ArrayList<Medicamento> buscarMedicamentosXNombre
                        (String nom) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (medicamento.getNombre().equals(nom))
      lista.add(medicamento);
   }
  return lista;
}
```

```
public ArrayList<Medicamento> buscarMedicamentosXPrecio(double precio) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (medicamento.getPrecio() == precio)
      lista.add(medicamento);
  }
  return lista;
}
```

Farmacia

- Solo cambia la condición del if que valida si un medicamento cumple con la condición de búsqueda

```
public ArrayList<Medicamento> buscarMedicamentosXLab(String lab) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (medicamento.getLaboratorio().equals(lab))
      lista.add(medicamento);
  }
  return lista;
}
```

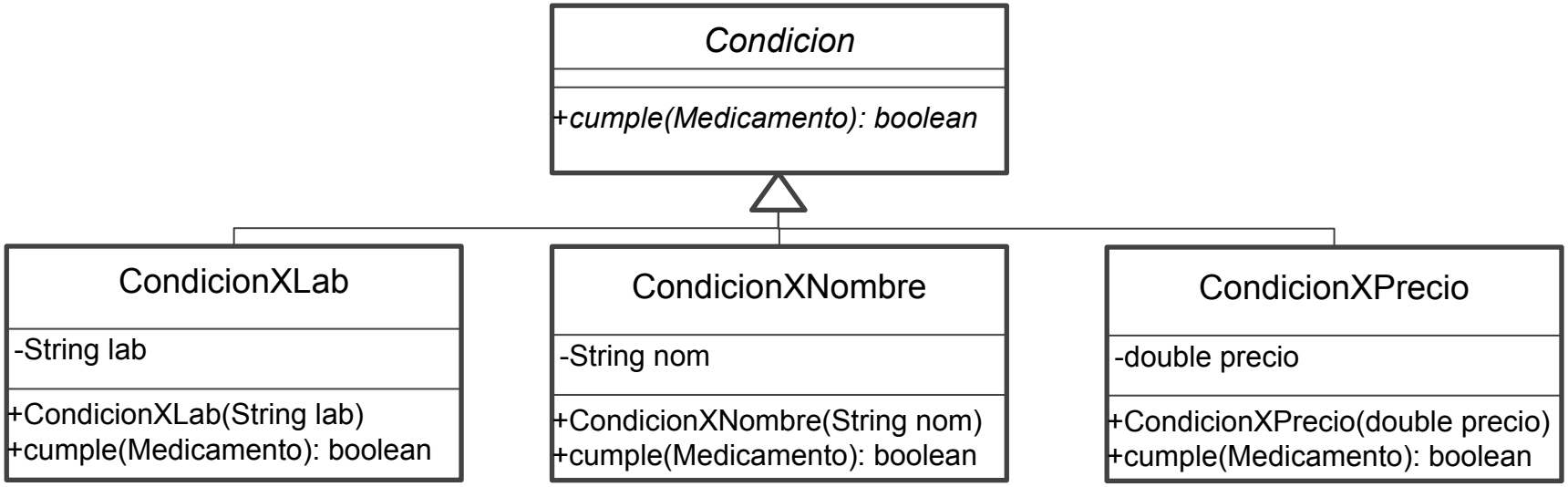
```
public ArrayList<Medicamento> buscarMedicamentosXNom
                              (String nom) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (medicamento.getNombre().equals(nom))
      lista.add(medicamento);
  }
  return lista;
}
```

```
public ArrayList<Medicamento> buscarMedicamentosXPrecio(double precio) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (medicamento.getPrecio() == precio)
      lista.add(medicamento);
  }
  return lista;
}
```

Por lo tanto, abstraemos la condición y las encapsulamos en una nueva jerarquía de clases

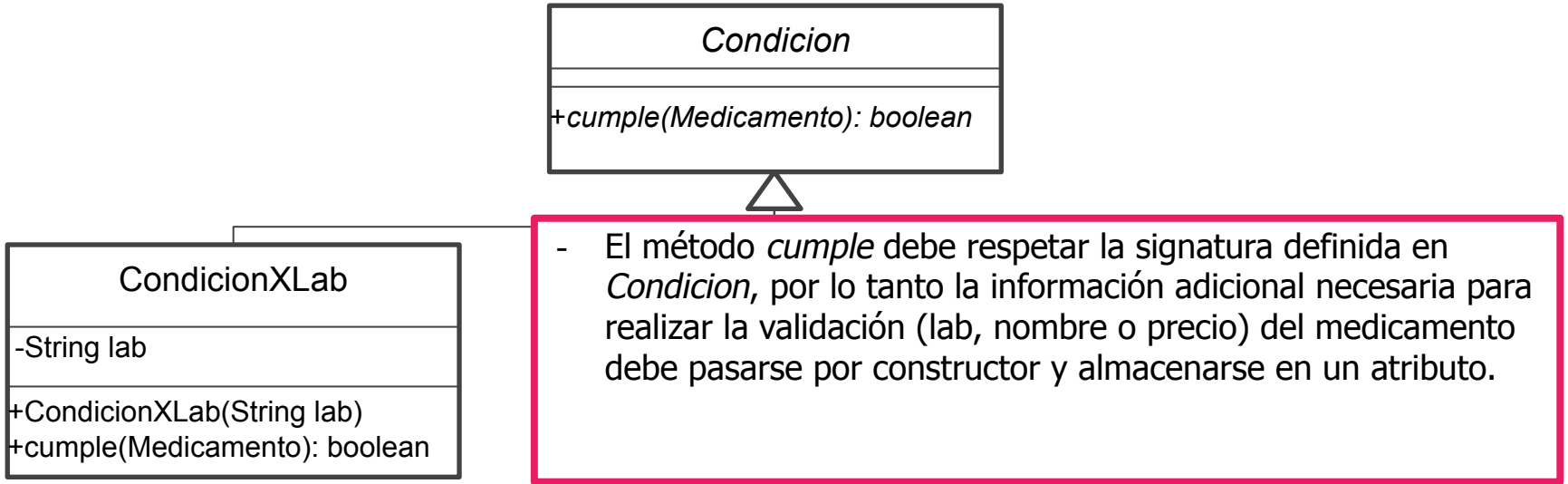
Farmacia - `Condicion`

- Defino una clase abstracta *Condicion* con un método abstracto que evalúa si el medicamento cumple o no con una determinada condición.
- De esta clase heredan las subclases concretas que representan cada una de las formas "simples" de búsqueda



Farmacia

- Defino una clase abstracta *Condicion* con un método abstracto que evalúa si el medicamento cumple o no con una determinada condición.
- De esta clase heredan las subclases concretas que representan cada una de las formas "simples" de búsqueda



Farmacia - `CondicionXLab`

- La implementación concreta de cada método *cumple* dependerá del tipo de condición que se quiera validar.

```
public class CondicionXLab extends Condicion {

  private String lab;

  public CondicionXLab(String lab) {
    this.lab = lab;
  }

  @Override
  public boolean cumple(Medicamento medicamento) {
    return medicamento.getLaboratorio().equals(this.lab);
  }
}
```

Farmacia - `CondicionXNombre`

- La implementación concreta de cada método *cumple* dependerá del tipo de condición que se quiera validar.

```
public class CondicionXNombre extends Condicion {

  private String nom;

  public CondicionXNombre(String nom) {
    this.nom = nom;
  }

  @Override
  public boolean cumple(Medicamento medicamento) {
    return medicamento.getNombre().equals(this.nom);
  }
}
```

ALTERNATIVA: `return medicamento.getNombre().contains(this.nom);`

Farmacia - `CondicionXPrecio`

- La implementación concreta de cada método *cumple* dependerá del tipo de condición que se quiera validar.

```
public class CondicionXPrecio extends Condicion {

  private double precio;

  public CondicionXPrecio(double precio) {
    this.precio = precio;
  }

  @Override
  public boolean cumple(Medicamento medicamento) {
    return medicamento.getPrecio() == this.precio;
  }
}
```

Farmacia - `buscarMedicamentos(Condicion)`

- Reformulamos el método de búsqueda en base a *Condición*

```
public ArrayList<Medicamento> buscarMedicamentos(Condicion c) {
  ArrayList<Medicamento> lista = new ArrayList<>();
  for (int i = 0; i < this.medicamentos.size(); i++) {
    Medicamento medicamento = this.medicamentos.get(i);
    if (c.cumple(medicamento))
      lista.add(medicamento);
  }
  return lista;
}
```

Si el medicamento cumple con la condición de búsqueda se agrega a la lista de resultados para retornar.

Ahora tenemos un solo método de búsqueda en Farmacia, que recibe una Condicion por parámetro. Como la condición cambia de búsqueda en búsqueda, no queremos instanciarla dentro del buscar! 🚩🚩🚩

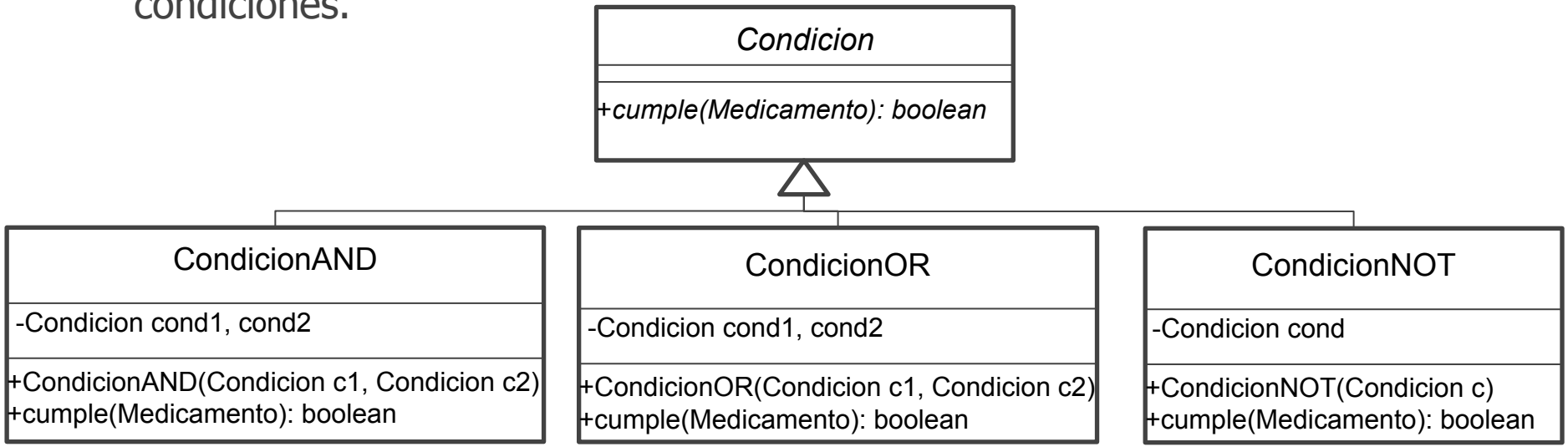
Farmacia
-ArrayList<Medicamento> medicamentos -…
+addMedicamento(Medicamento) +… +buscarMedicamentos(Condicion c): ArrayList<Medicamento>

Farmacia

- Combinaciones lógicas de los anteriores
    - Medicamentos de Bayer con precio igual a 100

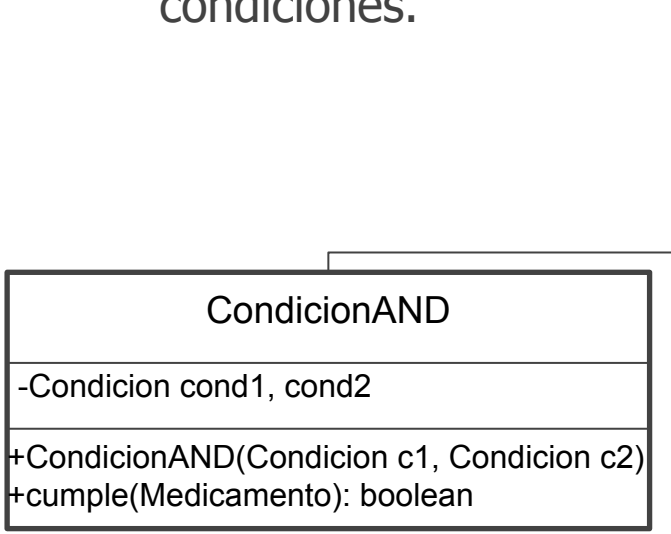
Farmacia – Combinaciones lógicas

- Las combinaciones lógicas AND, OR y NOT se modelan también como subclases de *Condicion*, y delegan parte de la responsabilidad en otras condiciones.



Farmacia – Combinaciones lógicas

- Las combinaciones lógicas AND, OR y NOT se modelan también como subclases de *Condicion*, y delegan parte de la responsabilidad en otras condiciones.



- Al igual que ocurre con las condiciones simples, las condiciones que componen cada parte del AND, OR y NOT son pasadas como parámetros en el constructor y almacenadas como atributos.
- Notar que cond1 y cond2 son de tipo *Condicion* para que pueda recibir instancias de cualquier subclase de ella.

Farmacia - `CondicionAND`

- La condición AND delega la responsabilidad de validar el Medicamento en las condiciones que la componen

```
public class CondicionAND extends Condicion {

  private Condicion cond1, cond2;

  public CondicionAND(Condicion c1, Condicion c2) {
    this.cond1 = c1;
    this.cond2 = c2;
  }

  @Override
  public boolean cumple(Medicamento medicamento) {
    return this.cond1.cumple(medicamento) && this.cond2.cumple(medicamento);
  }
}
```

Farmacia - `CondicionOR`

- La condición OR delega la responsabilidad de validar el Medicamento en las condiciones que la componen

```
public class CondicionOR extends Condicion {

  private Condicion cond1, cond2;

  public CondicionOR(Condicion c1, Condicion c2) {
    this.cond1 = c1;
    this.cond2 = c2;
  }

  @Override
  public boolean cumple(Medicamento medicamento) {
    return this.cond1.cumple(medicamento) || this.cond2.cumple(medicamento);
  }
}
```

Farmacia - `CondicionNOT`

- La condición NOT delega la responsabilidad de validar el Medicamento en la condición que la compone

```
public class CondicionNOT extends Condicion {

  private Condicion cond;

  public CondicionNOT(Condicion c) {
    this.cond = c;
  }

  @Override
  public boolean cumple(Medicamento medicamento) {
    return !this.cond.cumple(medicamento);
  }
}
```

Farmacia – Instanciación de las condiciones

- La instanciación de las condiciones se realiza fuera de las clases que representan el modelo. Ejemplos:

```
// Para buscar todos los medicamentos que contengan “ina” (según versión alternativa)
Condicion cond1 = new CondicionXNombre("ina");

// Para buscar todos los medicamentos del lab “Bayer”
Condicion cond2 = new CondicionXLab("Bayer");

// Para buscar todos los medicamentos que contengan “ina” y sean del lab “Bayer”
Condicion cond3 = new CondicionAND(cond1, cond2);

// Para buscar todos los medicamentos que contengan “ina” O sean del lab “Bayer”
Condicion cond4 = new CondicionOR(cond1, cond2);

// Para buscar todos los medicamentos que NO contengan “ina” y no sean del lab “Bayer”
Condicion cond5 = new CondicionNOT(cond3);
```