Factory Method Pattern

Software Design

PozoleSoft

- As the owner of Pozolerias “La tesorito” you want to have a system that could help you to create your different types of pozole

Creating pozoles

- At some point you end up having code like this:

```
public Pozole orderPozole() {
        Pozole pozole = new Pozole();

        pozole.prepare();
        pozole.serve();

        return pozole;
}
```

Different meats for pozole

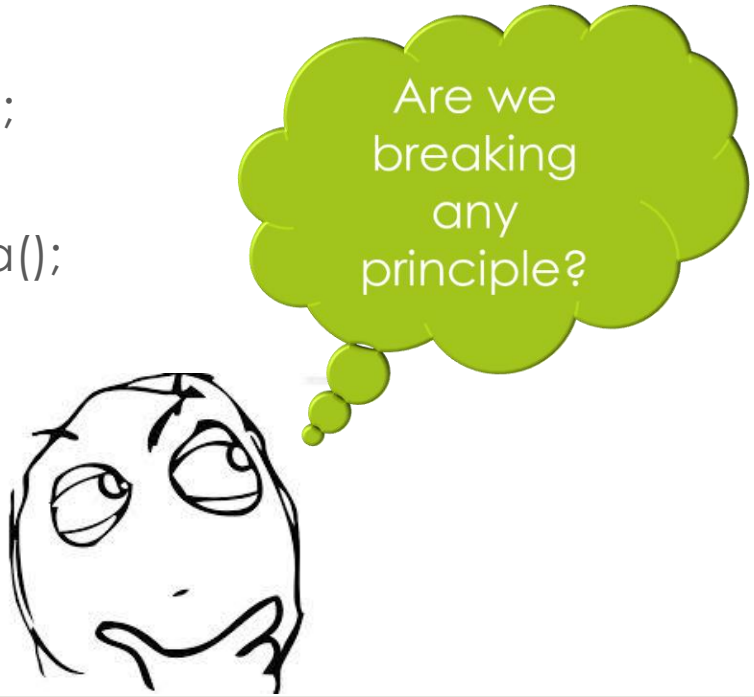
- Then we realize we sell different pozoles according to the meat:

```
public Pozole orderPozole( String meat ) {
        Pozole pozole;

        if(meat.equals("pollo"))
                pozole = new PozolePollo();
        else if(meat.equals("pierna"))
                pozole = new PozolePierna();
        else if(meat.equals("cachete"))
                pozole = new PozoleCachete();

        pozole.prepare();
        pozole.serve();

        return pozole;
}
```

What if we want to sell different meats

- We will need to add more else-if sentences:

```
…
else if(meat.equals("oreja"))
        pozole = new PozoleOreja();
else if(meat.equals("trompa"))
        pozole = new PozoleTrompa();
…
```



Get rid of “new”

- If we want to be **open for extension** (add different meats for the pozole) we need to alter the code each time
- If we have to do that we won’t be **closed for modification**
- We need to get rid of the **new** statement.
- **HOW?**

Lets encapsulate what varies

What stays the same

```
public Pozole orderPozole(
String meat ) {
        Pozole pozole;

        pozole.prepare();
        pozole.serve();

        return pozole;
}
```

What varies

```
if(meat.equals("pollo"))
        pozole = new PozolePollo();
else if(meat.equals("pierna"))
        pozole = new PozolePierna();
else if(meat.equals("cachete"))
        pozole = new PozoleCachete();
else if(meat.equals("oreja"))
        pozole = new PozoleOreja();
else if(meat.equals("trompa"))
        pozole = new PozoleTrompa();
```

Lets create the SimplePozoleFactory

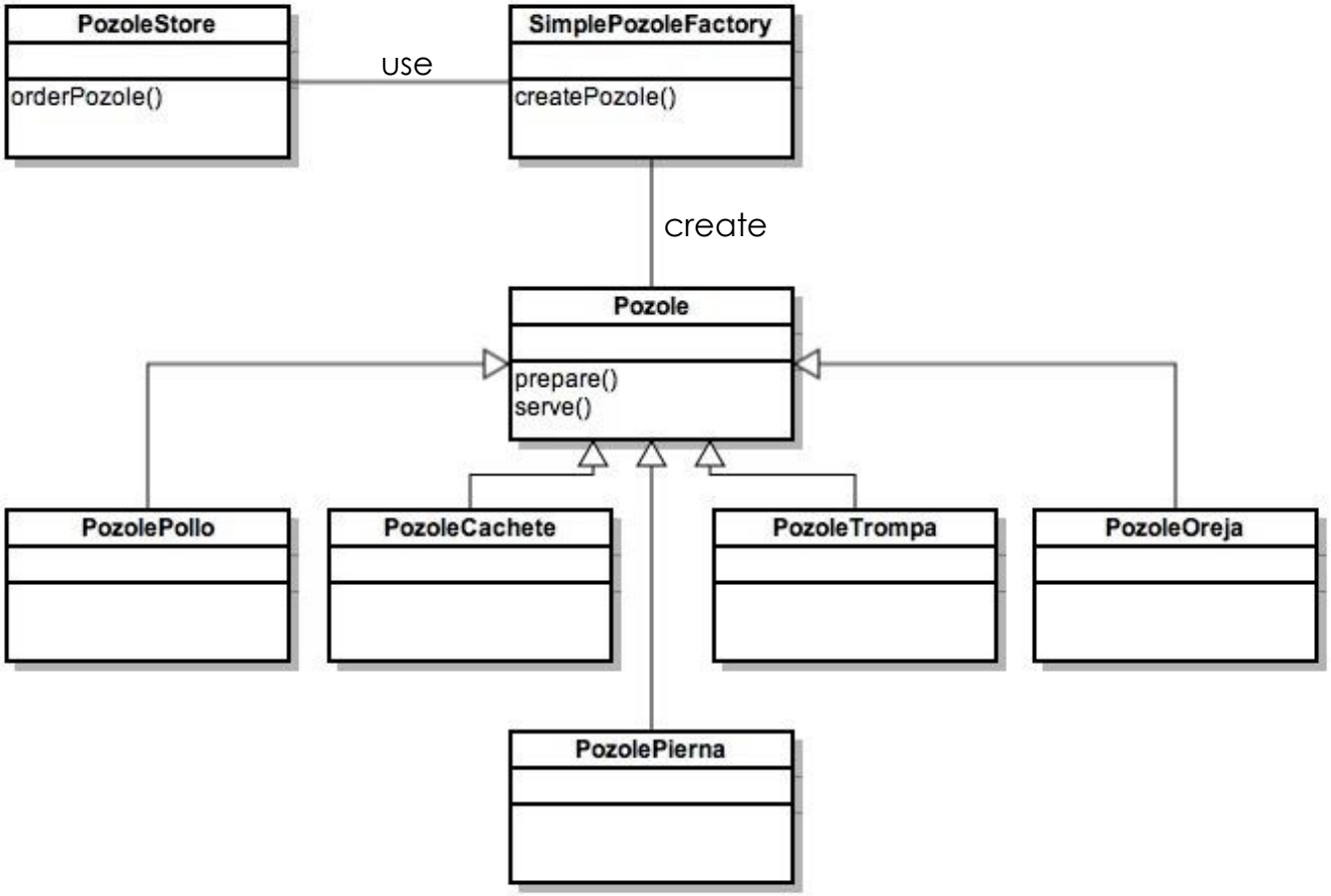
- Encapsulating the object creation for all pozoles:

```
public class SimplePozoleFactory {
    public Pozole createPozole (String meat) {
        Pozole pozole = null;

        if(meat.equals("pollo"))
            pozole = new PozolePollo();
        else if(meat.equals("pierna"))
            pozole = new PozolePierna();
        else if(meat.equals("cachete"))
            pozole = new PozoleCachete();
        else if(meat.equals("oreja"))
            pozole = new PozoleOreja();
        else if(meat.equals("trompa"))
            pozole = new PozoleTrompa();

        return pozole;
    }
}
```

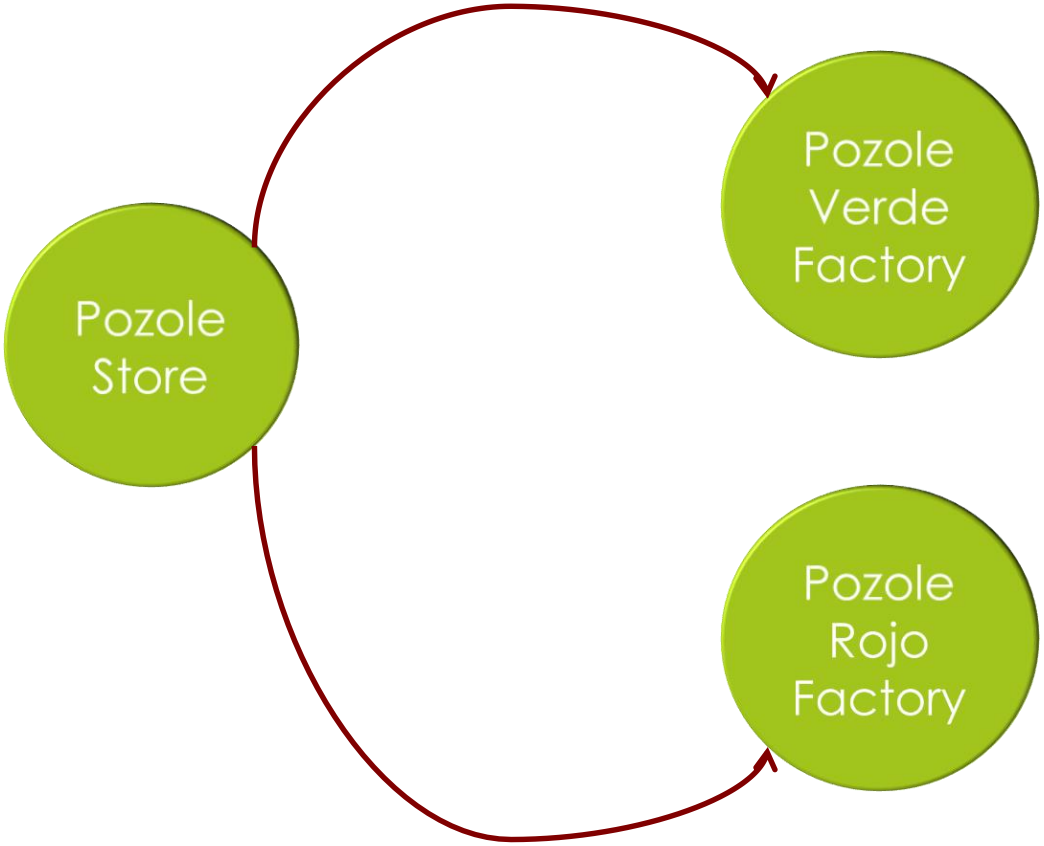
Reworking the PozoleStore class

```
public class PozoleStore {

    SimplePozoleFactory factory;

    public PozoleStore (SimplePozoleFactory factory){

        this.factory = factory;

    }
    public Pozole orderPozole( String meat ) {
        Pozole pozole;

        pozole = factory.createPozole( meat );

        pozole.prepare();
        pozole.serve();

        return pozole;
    }
}
```

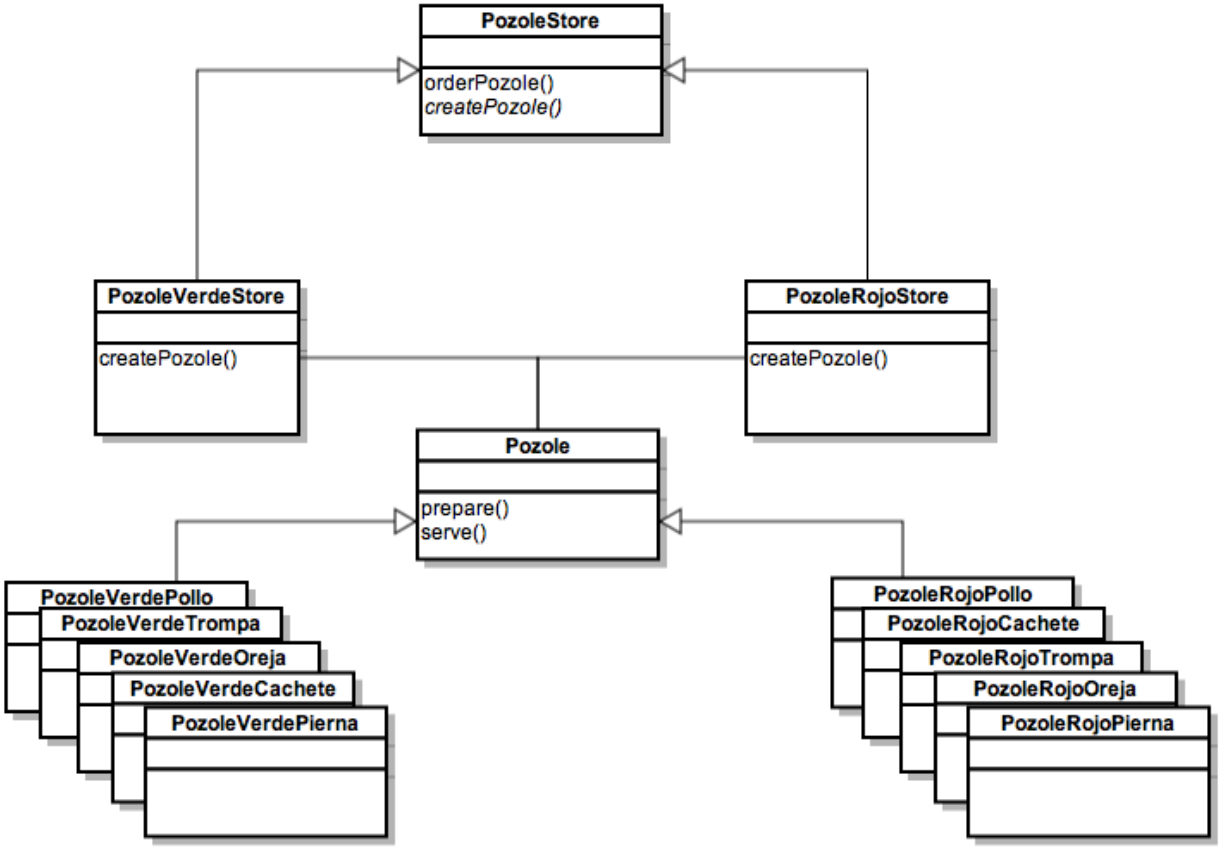
Simple Factory



Franchising the Pozole store



Pozole Franchise



Implementation

Factory Method Pattern

- Defines an interface for creating an object, but lets subclasses decide which class to instantiate.
- Factory method lets a class defer instantiation to subclasses

Factory Method class diagram