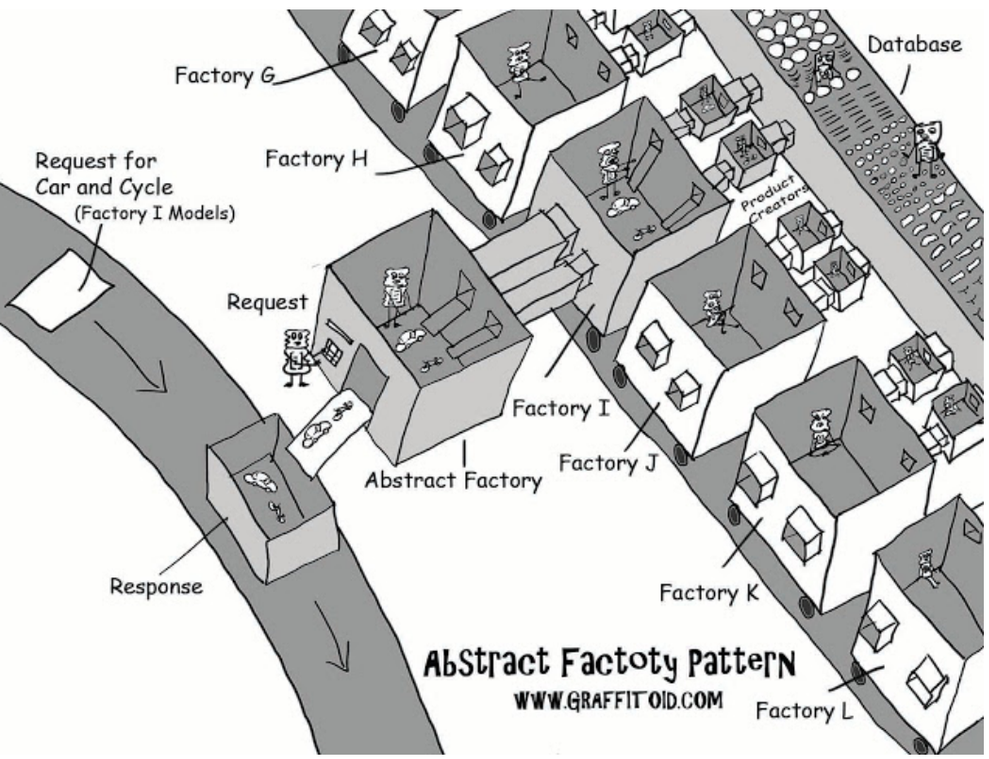


Tema 3: Disseny

Anna Puig

Enginyeria Informàtica

Facultat de Matemàtiques i Informàtica,

Universitat de Barcelona

Curs 2020/2021

UNIVERSITAT DE BARCELONA

Temari

1. Introducció al procés de desenvolupament del software
2. Anàlisi de requisits i especificació
3. Disseny
   - 3.1 Introducció
   - 3.2 Patrons arquitectònics
   - 3.3 Criteris de Disseny: G.R.A.S.P.
   - 3.4 Principis de Disseny: S.O.L.I.D.
   - 3.5 **Patrons de disseny**
4. Del disseny a la implementació
5. Ús de frameworks de testing

-

3.4. Patrons de disseny

Propòsit → **Àmbit ↓**	**CREACIÓ**	**ESTRUCTURA**	**COMPORTAMENT**
CLASSE	• Factory method	• class Adapter	• Interpreter • Template method
OBJECTE	• **Abstract Factory** • Builder • Prototype • Singleton • Object pool	• Object Adapter • Bridge • Composite • Decorator • Facade • Flyweight • Proxy	• Chain of Responsability • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor

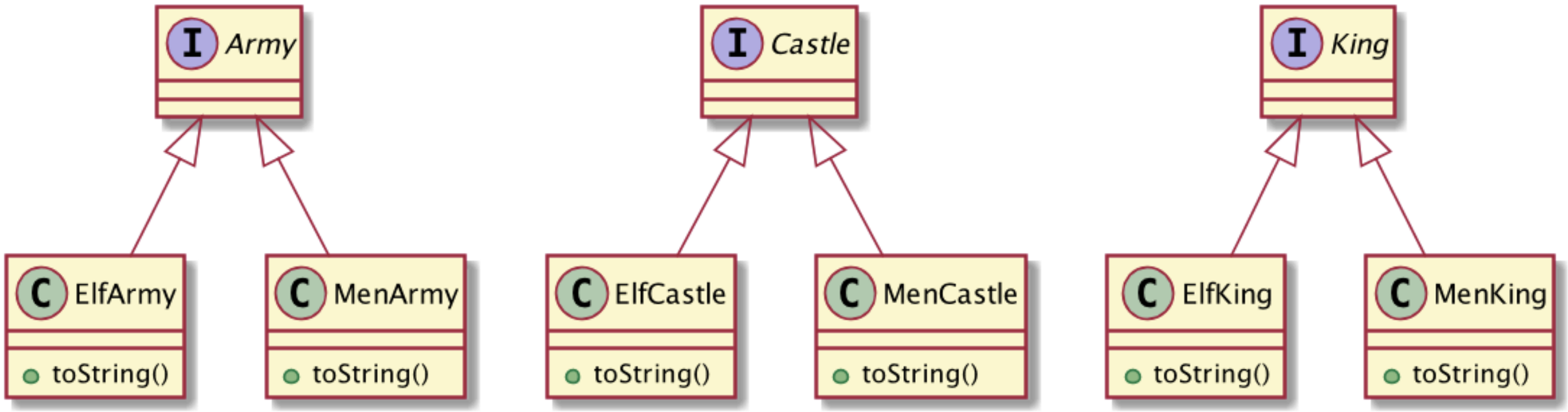
Patrons Factory

- **Factory Method** – Defineix una classe abstracte per crear objectes, però deixa a les subclasses decidir quina classe ha d'instanciar i consulta el nou objecte creat a través d'una interfície comú dels objectes creats

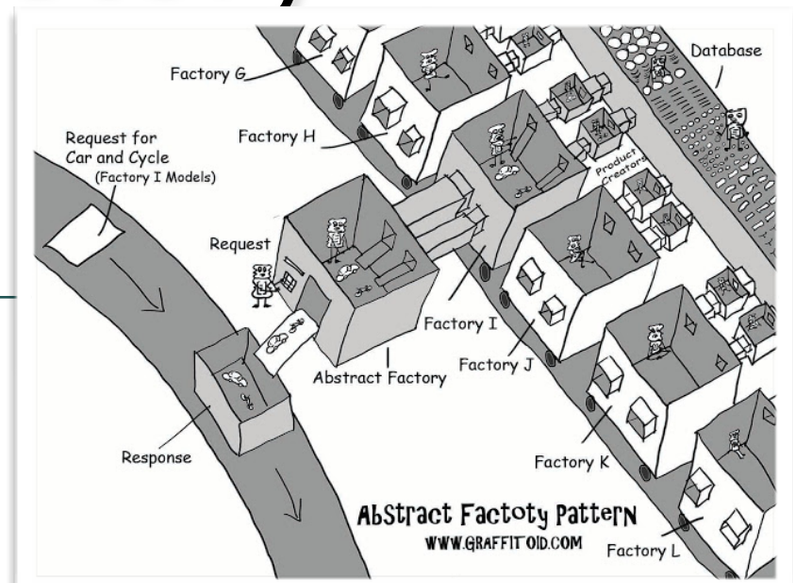
- **Abstract Factory** – Ofereix una interfície per crear una **família d'objectes** relacionats, sense explícitament especificar les seves classes

Exemple Patró Abstract Factory

- Volem crear dos regnes (els dels elfs i els dels homes). Cada regne té un castell, un rei i una armada. Per a cada un dels elements d'un regne es dissenya una interfície
- Com solucionem la seva creació "coordinada"?



Patró Abstract Factory



Nom del patró: **Abstract Factory**

Context: **Creació**

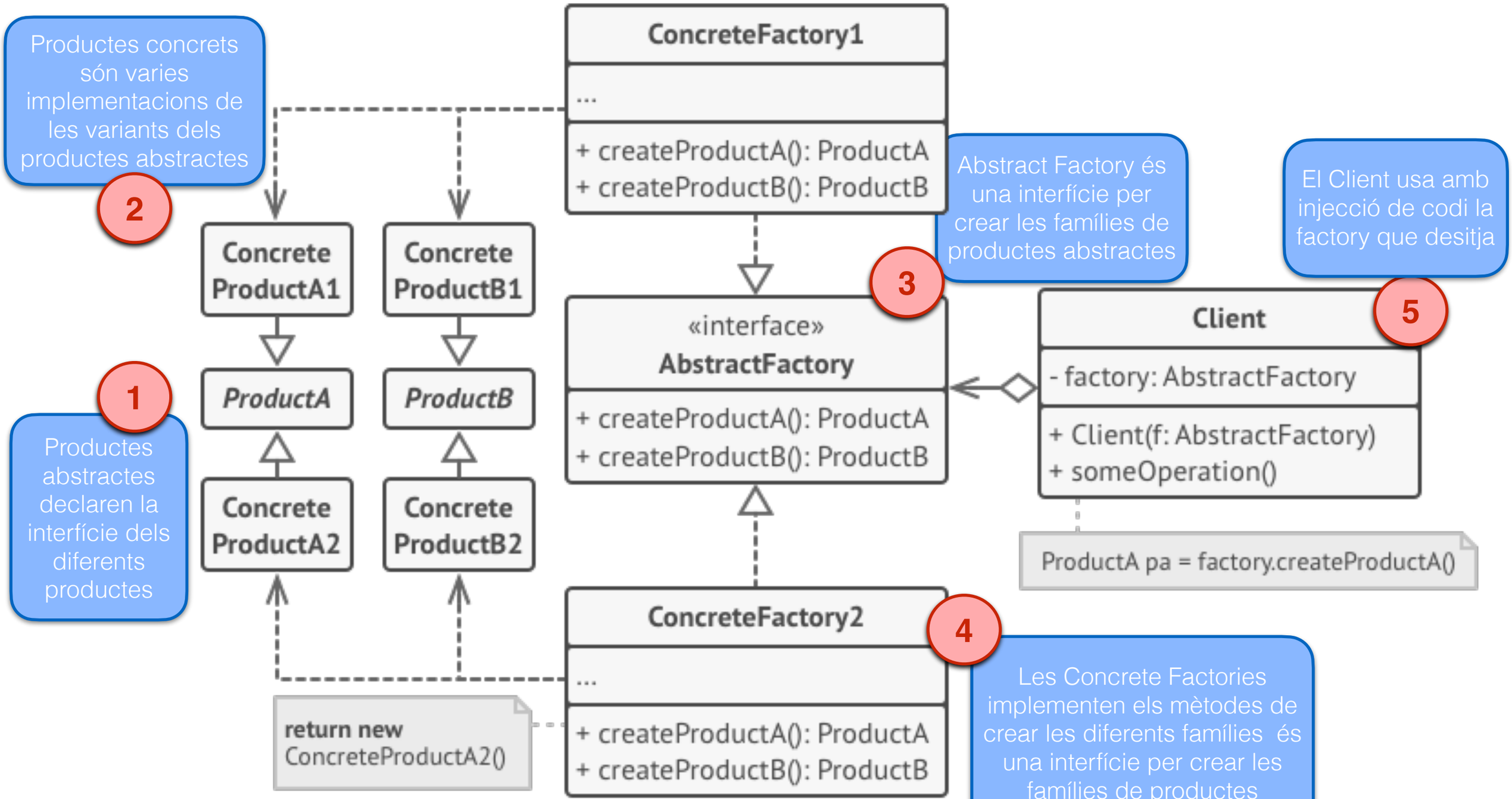
Problema:

- Crear **famílies d'objectes relacionats** o dependents sense especificar les classes concretes

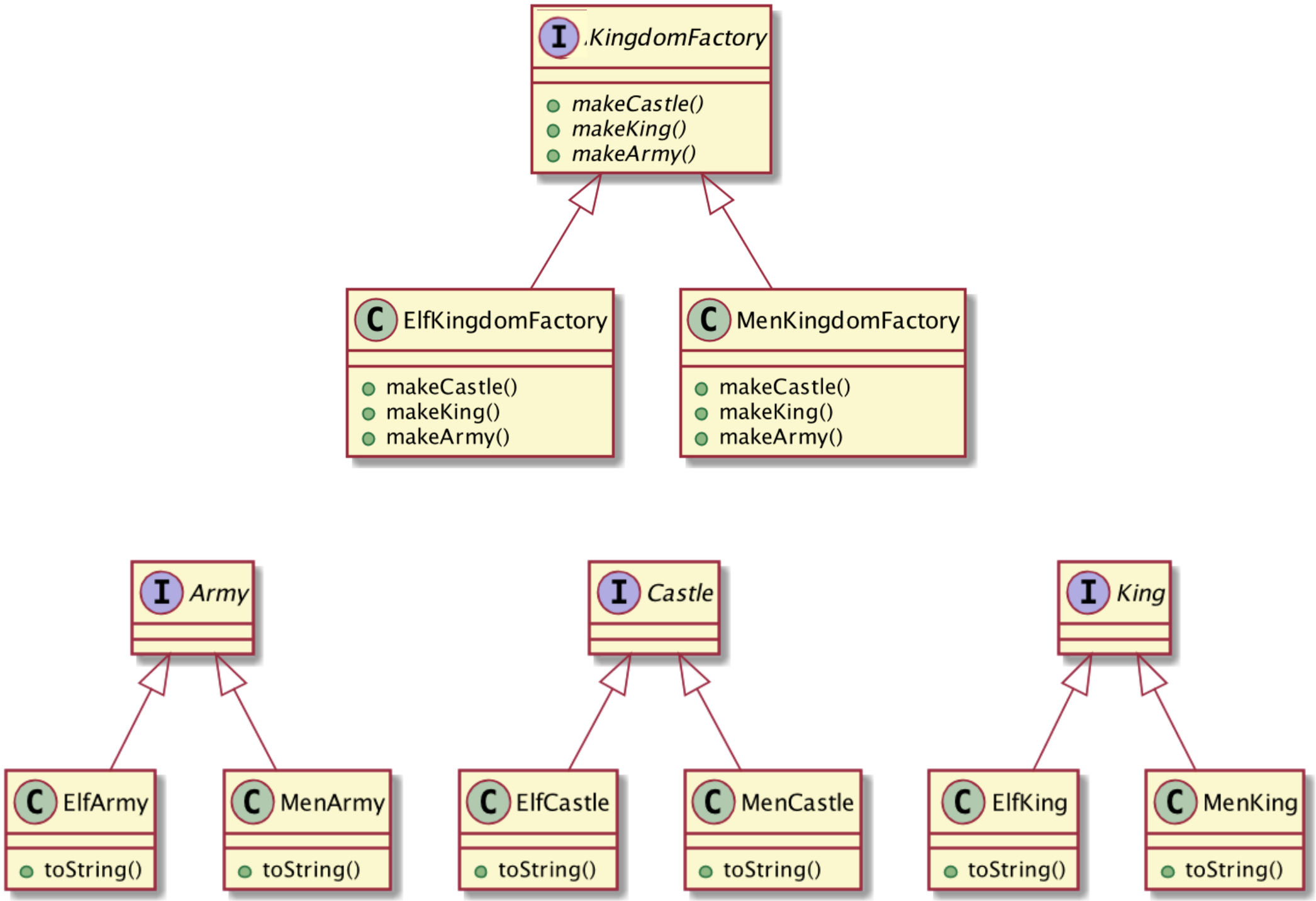
Solució:

- Es fa una interfície que és responsable de crear factories d'objectes relacionats sense explicitar directament les seves classes

Patró Abstract Factory



Exemple Patró Abstract Factory



Exemple Patró Abstract Factory

```
public class KingdomTestDrive {

    public static void main(String[] args) {
        createKingdom(new ElfKingdomFactory());
        createKingdom(new MenKingdomFactory());

    }

    public static void createKingdom(KingdomFactory factory) {
        King king = factory.makeKing();
        Castle castle = factory.makeCastle();
        Army army = factory.makeArmy();
        System.out.println("The kingdom was created: ");
        System.out.println(king);
        System.out.println(castle);
        System.out.println(army);
    }

}
```

Patró Abstract Factory

Nom del patró: **Abstract Factory**
Context: **Creació**
Pros:

- Centralització en la creació d'objectes
- Facilita l'escalabilitat del sistema
- Serveix per definir sistemes que poden configurar-se amb una de vàries famílies de productes
  - Per exemple, definir una interfície que soporti diferents sistemes de finestres (e.g. Windows, OpenView, Motif, ...)
- L'usuari s'abstrau de la/les instància/es a crear

Permet proporcionar una llibreria de classes on només es permet mostrar les seves interfícies i no les seves implementacions

Cons:

- Els codi esdevé més complexe ja que el patró introdueix un nombre addicional de classes