Taller 5 – Grupo 7

Integrantes:

Alejandro Charry Acevedo - ja.chary@uniandes.edu.co - 202111151

Santiago Rodriguez Mora - s.rodriguezm2@uniandes.edu.co - 202110332

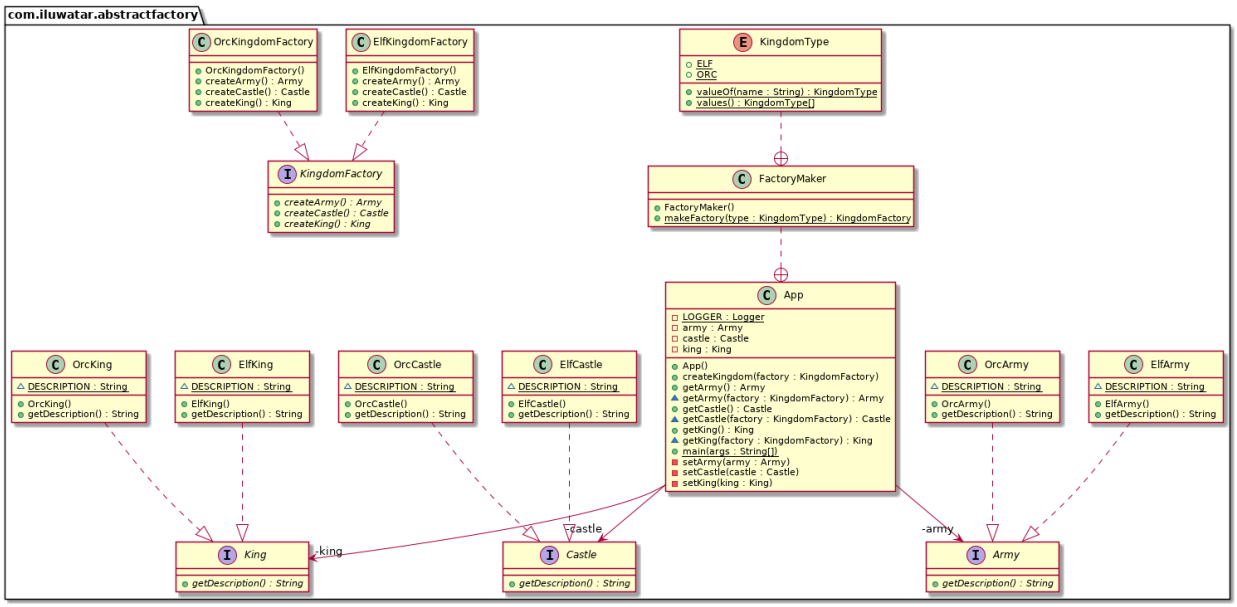
Valeria Torres Gomez - v.torresg23@uniandes.edu.co – 202110363

1. **Link al repositorio escogido:**
   https://github.com/iluwatar/java-design-patterns/tree/master/abstract-factory

2. **Información general del proyecto:**
   El repositorio escogido corresponde a un proyecto que modela un reino fantástico con elfos, ejércitos, castillos y el reino. El repositorio simula un reino completo y se basa en el patrón abstract factory para lograr optimizar sus acciones.

3. **Patrón escogido:**
   Abstract Factory: El abstract factory trata de solucionar el problema de crear diferentes familias de objetos. El patrón es aconsejado cuando se prevé la inclusión de nuevas familias de objetos, pero puede resultar contraproducente cuando se añaden nuevos objetos o cambian los existentes, puesto que afectaría a todas las familias creadas.

4. **Diagrama de clases del repositorio:**



5. **Código donde se implementa el patrón:**

```
public interface KingdomFactory {
  Castle createCastle();
  King createKing();
  Army createArmy();
}

public class ElfKingdomFactory implements KingdomFactory {

  @Override
  public Castle createCastle() {
    return new ElfCastle();
  }

  @Override
  public King createKing() {
    return new ElfKing();
  }

  @Override
  public Army createArmy() {
    return new ElfArmy();
  }
}

public class OrcKingdomFactory implements KingdomFactory {

  @Override
  public Castle createCastle() {
    return new OrcCastle();
  }

  @Override
  public King createKing() {
    return new OrcKing();
  }

  @Override
  public Army createArmy() {
    return new OrcArmy();
  } }
```

6. **Implementación del patrón dentro del código:**
   Para crear un reino necesitamos objetos con una estructura común. El reino elfo necesita un rey elfo, un castillo de elfos, y un ejército de elfos, mientras que el reino orco necesita un rey orco, un castillo orco y un ejército orco. Hay una dependencia entre los objetos en el reino.
   En otras palabras, el patrón ofrece una fábrica de fábricas; una fábrica que agrupa a las fábricas individuales pero relacionadas/dependientes sin especificar sus clases concretas. El patrón "abstract factory" proporciona una forma de encapsular un grupo de fábricas individuales que tienen un tema común sin especificar sus clases concretas.

7. **Ventajas y Desventajas del uso del patrón "abstract factory":**
   Ventajas:
   - La principal ventaja al usar "abstract factory" es que se pueden agrupar todas las fábricas de los objetos necesarios en una sola fábrica principal: "Kingdom Factory" dentro del proyecto. Si se necesitara algún otro objeto con la misma estructura que se tiene (reino, rey, ejército), se podría usar la misma "Kingdom Factory" que es el uso del patrón, ya que al ser abstracta podría manejar este nuevo tipo de objeto sin especificar la clase.
   - La ventaja inmediatamente anterior abarca también el hecho de que no se necesita estar escribiendo código específico para clase de cada objeto que pudimos encapsular en la misma fábrica abstracta. En otras palabras, "abstract factory" permite la implementación de código abstracto (sin especificación de su clase) para la creación de varios objetos que siguen una misma estructura.

   Desventajas:
   - Puede ser que el código sea de difícil comprensión para cualquier persona de un equipo que trabaje sobre el repositorio que no esté familiarizado con el patrón.

8. **Otras formas de solucionar el caso particular sin recurrir al patrón "abstract factory":**
   Otra manera en la que se puede solucionar el caso particular es a través de la creación de fábricas individuales para la creación de cada objeto según su clase. Sin embargo, esto no tendría mucho sentido de seguir, puesto que recurriría en demasiado código innecesario y repetitivo para cada objeto que siga la misma estructura anteriormente mencionada.