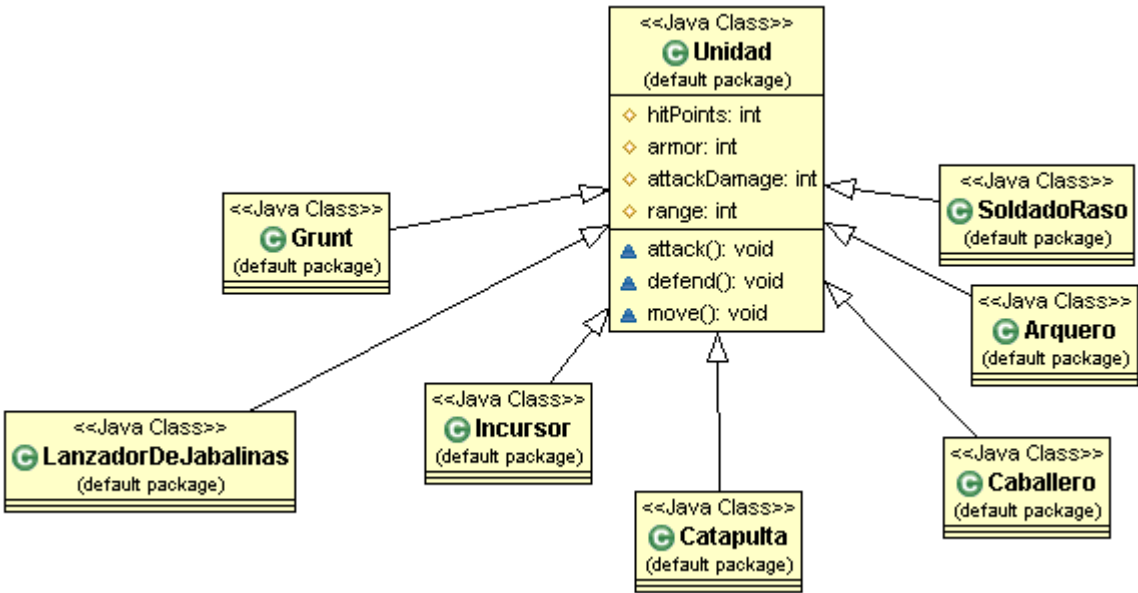


El juego Warcraft: Humanos vs Orcos® es un juego de estrategia donde dos especies combaten entre sí: los humanos y los orcos. Cada Especie dispone de los mismos tipos de unidades militares, aunque particularizada para cada especie. Podemos verlo de forma resumida en la siguiente tabla:

Tipo	Especie			
	Humanos		Orcos	
Infanteria		Soldado Raso		Grunt
Arquero		Arquero		Lanzador de Jabalinas
Jinete		Caballeros		Incursores
MaquinaAsedio		Catapulta		Catapulta

Para representar en memoria estos personajes se ha decidido utilizar la siguiente jerarquía de clases:



A partir de esa jerarquía se ha modelado una inteligencia artificial (I.A.) para que dirija a los Orcos. Un fragmento de esa clase `IAOrcos` es la de la derecha, donde podemos ver un método que crea una unidad de combate formada por varias unidades de cada tipo.

Los programadores al comprobar lo bien que funciona esta I.A. han decidido que sirva tanto para Orcos como para Humanos.

Para ello, se renombrará la clase como `IAComun`. Y se creará un constructor `IAComun(String Especie)`, para que en función de esa cadena se creen unidades orcas (“orc”) o unidades humanas (“humanos”). Aplica el *patrón de diseño* adecuado para evitar tener que crear una sentencia condicional con cada `new` a la hora de crear un tipo de unidad concreto.

```
// Inteligencia artificial de los orcos
public class IAOrcos {

    /**
     * Generación de un grupo de ataque de la I.A.
     * @return Un grupo de ataque orco
     */
    public Unidad[] creaGrupoDeAtaque()
    {
        // Array de Unidades Orcas
        Unidad[] grupoDeAtaque = new Unidad[10];

        // 4 x infanteria
        for(int x = 0; x < 4; x++)
            grupoDeAtaque[x] = new Grunt();
        // 3 x arqueros
        for(int x = 4; x < 7; x++)
            grupoDeAtaque[x] = new LanzadorDeJabalinas();
        // 2 x jinetes
        grupoDeAtaque[7] = new Incursor();
        grupoDeAtaque[8] = new Incursor();
        // 1 x máquina de asedio
        grupoDeAtaque[9] = new Catapulta();

        return grupoDeAtaque;
    }
}
```

1. De los patrones de diseño vistos en clase, ¿cuál resulta más adecuado para que esta clase sirva para poder escoger fácilmente la especie de las unidades a crear y manipular en cada objeto **IAComun**? **Justifica tu respuesta. (1,5 puntos)**

*El patrón debería ser de tipo creacional ya que nos estamos ocupando precisamente de la creación de objetos. El patrón que mejor se amolda a este problema es la **Factoría Abstracta (Abstract Factory)**. Ya que debemos construir distintas familias de objetos (objetos Unidad en el problema). Definiendo una familia por cada especie: humano u orco.*

*Este patrón permite desacoplar “la necesidad de crear un objeto nuevo”, de “la aplicación del operador **new** sobre una clase concreta”. De forma que el objeto que crea las unidades será un objeto distinto del que las necesita y luego utiliza.*

Además Abstract Factory es el patrón creacional que nos asegura que una vez fijada la especie (la familia), la factoría de objetos va a generar exclusivamente productos de su familia.

2. Una vez aplicado este patrón ¿se podría dar el caso de que en un mismo grupo de ataque se generen unidades de ambas especies? **Justifica tu respuesta. (1,5 puntos)**

No, no se puede dar esta situación. La factoría concreta escogida generará o bien unidades humanas o bien unidades orcas. Cada factoría concreta sólo generará unidades de una especie.

3. ¿Qué estructura de clases deberíamos añadir, y cómo se integrarían con las ya existentes? Dibuja el diagrama. Indica la función de cada nueva clase. **(2,5 puntos)**

*Deberíamos crear una interfaz **FactoriaAbstracta** con la declaración de las operaciones que devuelven unidades de cada tipo: **Unidad creaInfanteria()**; **Unidad creaArquero()**; **Unidad creaJinete()**; **Unidad creaMaquinaAsedio()**; ¹*

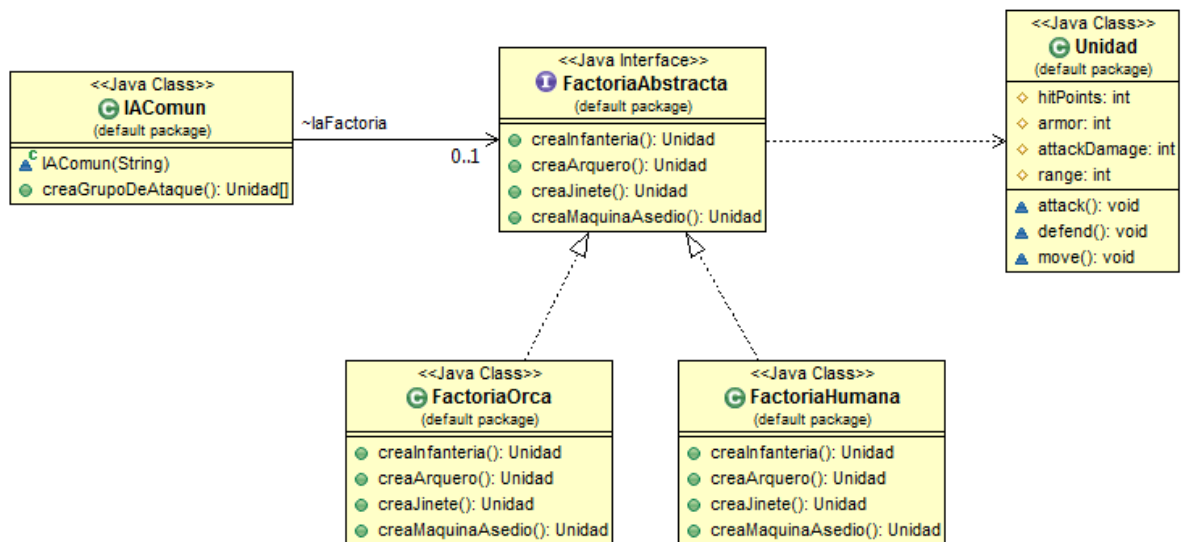
*Esta interfaz la implementarían las clases **FactoriaHumana** y **FactoriaOrca**. Una factoría concreta por cada especie. Cada factoría concreta implementará las operaciones anteriores, creando vía el operador **new** unidades de su especie. A modo de ejemplo en **FactoriaHumana**:*

```
Unidad creaInfanteria() {  
    return new SoldadoRaso();  
}
```

*Finalmente la clase **IAComun** utilizará **FactoriaAbstracta** por composición para generar sus unidades.*

*De esta manera, la clase que necesita unidades y las utiliza (**IAComun**) es distinta de la clase que crea finalmente los objetos (**FactoriaOrca** o **FactoriaHumana**).*

¹ De acuerdo con la arquitectura general se podría definir una interfaz abstracta por cada tipo de producto: **IInfanteria**, **IArquero**, **IJinete**, **IMaquinaAsedio**, y cada tipo de unidad concreto implementaría su interfaz. De forma que, por ejemplo **creaJinete()** devolvería un objeto **IJinete**. Pero en nuestro caso no es necesario ya que todas las unidades tienen una clase base común: **Unidad**.



4. Implementa la clase IComun con su nuevo constructor y su método creaGrupoDeAtaque(). (2,5 puntos)

```

// Inteligencia artificial común
public class IComun {

    FactoriaAbstracta laFactoria;

    IComun(String especie){
        // Creamos la factoria concreta adecuada a la especie
        if(especie.equals("orcocos"))
            laFactoria = new FactoriaOrcos();
        else
            laFactoria = new FactoriaHumanos();
    }

    /**
     * Generación de un grupo de ataque de la I.A.
     * @return Un grupo de ataque orco o humano dependiendo de su factoria
     */
    public Unidad[] creaGrupoDeAtaque()
    {
        // Array de Unidades Orcas
        Unidad[] grupoDeAtaque = new Unidad[10];

        // 4 x infanteria
        for(int x = 0; x < 4; x++)
            grupoDeAtaque[x] = laFactoria.creaInfanteria();
        // 3 x arqueros
        for(int x = 4; x < 7; x++)
            grupoDeAtaque[x] = laFactoria.creaArquero();
        // 2 x jinetes
        grupoDeAtaque[7] = laFactoria.creaJinete();
        grupoDeAtaque[8] = laFactoria.creaJinete();
        // 1 x máquina de asedio
        grupoDeAtaque[9] = laFactoria.creaMaquinaAsedio();

        return grupoDeAtaque;
    }
}

```

5. ¿Resultaría muy costoso añadir una tercera especie al juego? Por ejemplo:

Especie No-Muertos (Esqueleto/LanceroZombi/JineteSinCabeza/CañonDeLaPlaga)


¿Qué cambios habría que hacer? (1 punto)

No resultaría muy costoso. En el sentido de que no necesitamos modificar ninguna de las factorías (clases) ya creadas. Simplemente supone crear clases nuevas.

1º/ Definiríamos estas nuevas unidades como subclases de Unidad.

*2º/ Crearíamos una nueva factoría concreta **FactoriaNoMuertos**, y que su operación **creaInfanteria()** creara objetos de la clase **Esqueleto**, y de forma similar con el resto de tipos de unidades.*

6. ¿Resultaría muy costoso añadir un nuevo tipo de unidades a cada especie? Por ejemplo:

Tipo	Especie					
	Humanos		Orcos		No-Muertos	
Mago		Conjurador		Hechicero		Nigromante

¿Qué modificaciones habría que realizar? (1 punto)

Resultaría algo más costoso que el apartado anterior, al tener que modificar todas las factorías concretas que ya habíamos definido.

1º/ Crearíamos las nuevas unidades como subclase de Unidad.

*2º/ Habría que crear en la factoría abstracta un nuevo método: **Unidad creaMago()** ;*

3º/ Implementar esta operación en cada factoría concreta.

*Por ejemplo, para la **FactoriaHumana**:*

```
Unidad creaMago() {  
    return new Conjurador();  
}
```