1**. From the design patterns which have been studied along the different lectures, which would be the most appropriate in order to facilitate to the class AICommon to choose the specie, and then create a specific number of units to be managed in every object?**

We have to choose a creational bc we want to create speific number of units

We can use Abstract Factory because you want to create different families of object (units) we dont want to mix them together

We can define a family by specie: human or orc

This pattern allows to disengage “the need of creating a new object” from“the application of the operator new over a specific class”. So, the object which create the units would be a different object from the one which needs and uses them after.

Moreover, the Abstract Factory is the creational pattern to assure that after establishing the specie (the family), the factory of objects will generate exclusively products of its family

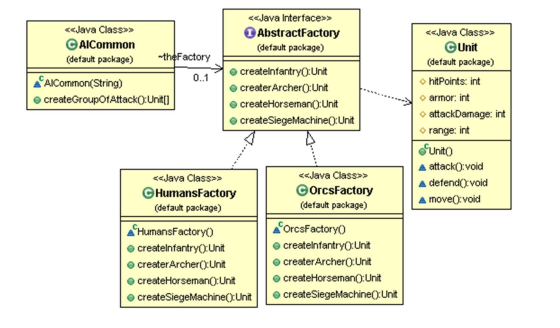
2. **After applying this pattern, could it be possible to generate units for both species in the same attack group?**

No, because we want to be separating the groups. With abstract fatories we are not allowed to do that

The specific factory will generate human units or orc units. Therefore, each specific factory will generate only units of a concrete specie.

3. **Which structure of classes has to be added, how will it integrate with the classes which already exist? Draw the diagram. Indicate the function for every new class.**

4 units so we need 4 products for each species



4. **Implement the class AICommon with its new constructor and its method createGroupOfAttack()**

// Artifitial Intelligence for Orcs

**package** wc1.base;

**public** **class** AICommon {

AbstractFactory factory;

**public** AICommon (String species) {

// We create the specific factory according to the specie

**if**(species.equals("orcs"))

factory = **new** FactoryOrcs();

**else**

factory = **new** FactoryHuman();

}

/\*\*

\* Generation of a group of attack for A.I.

\* **@return** An attack group with Orcs

\*/

**public** Unit[] createGroupOfAttack()

{

// Array of Orcs Units

Unit[] groupOfAttack = **new** Unit[10];

// 4 x infantry

**for**(**int** x = 0; x < 4; x++)

groupOfAttack[x] = factory.createInfantry();

// 3 x archers

**for**(**int** x = 4; x < 7; x++)

groupOfAttack[x] = factory.createArcher();

// 2 x horseman

groupOfAttack[7] = factory.createHorseman();

groupOfAttack[8] = factory.createHorseman();

// 1 x machine of siege

groupOfAttack[9] = factory.createSiegeMachine();

**return** groupOfAttack;

}

}