ESKİŞEHİR TEKNİK ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

BIM492
Design Patterns

Homework 2

1 - Statement of Work

I will create an example of “**Abstract Factory Pattern**”. My project is creating Droids from 3 Droid Factories. The Abstract Factory Pattern provides an interface for creating families of related or dependent objects without specifying their concrete class.

I have a **DroidFactory** interface. A Droid Factory creates PilotDroid, BattleDroid, JumpDroid.

I have 3 concrete Factory Class like IronDroidFactory, PlasticDroidFactory, TitaniumDroidFactory and implements **DroidFactory**.

This Factories have own models. For example IronDroidFactory have IronBattleDroid, IronJumpDroid, IronPilotDroid.

That’s all!

2 - Design Pattern

So I will create a project works like this idea and I choose the **Abstact Factory** pattern for this idea.

Imagine that we are developing a framework for a GUI environment, were windows will be drawn on a display device and the user will interact with the GUI using a mouse and a keyboard.

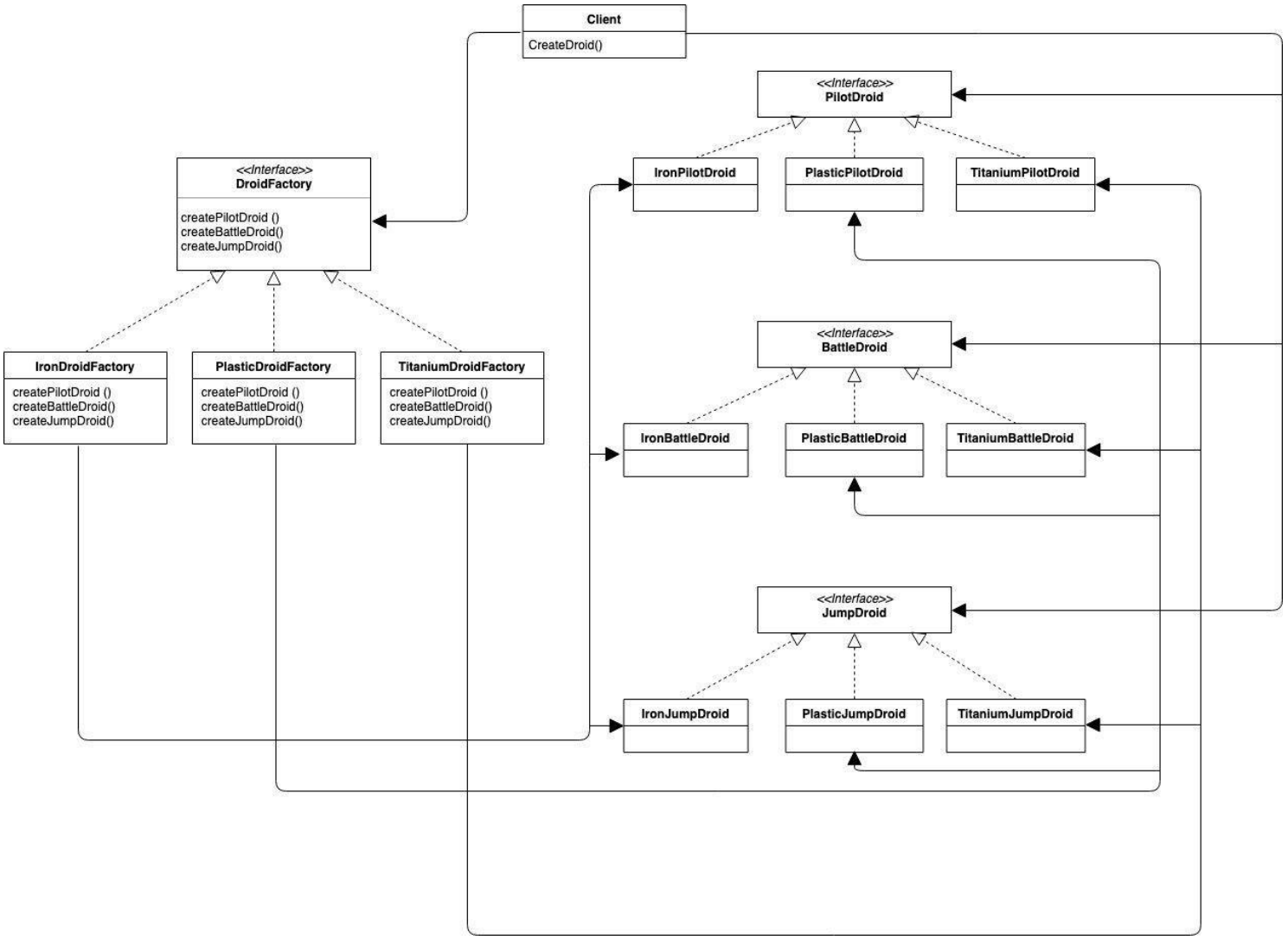
The first version of the framework will support Windows OS, so Factory method is used for creation of the graphical abstractions like Frame, Window, ScrollBar, etc.

In the next version, the framework will be extended to Linux OS. So, how should we extend our factory method?

One way would be to introduce **factory abstraction**, where each OS will have dedicated factory for creation of the graphical abstractions. The proposed solution is an example of the **Abstract Factory**.

The **Abstract Factory** is one level higher in abstraction than the **Factory Method**. The Factory Method abstracts the way objects are created, while the Abstract Factory also abstracts the way factories are created, which in turn abstracts the way objects are created.

3 - UML Diagram



4 - Research

There are two design patterns which are called “**factory**” patterns: **factory method** and **abstract factory**.

They are called factories because they produce something. The interesting thing is that there are other **creational** design patterns which are called differently, but they still produce something, so, just following this logic, all those patterns could be called factories.

The devil is, as always, in the details. Singleton is focused on the ability to reuse the same object. Prototype is focused on the ability to create a copy of an object. Factory patterns are focused on the mass production of new objects.

Still, what would be the real-life analogy for the abstract factory?

We are talking about the facility that can produce families of objects. It seems, this is what restaurants do.. Restaurants can produce breakfasts, launches, and dinners. What exactly you get for launch depends on which restaurant you go to. Those are the keys in this pattern:

– There is a **client** (you)

– The client is provided with a concrete implementation of the **abstract factory** (that particular restaurant around the corner)

– The client knows that that particular factory can **produce objects** of **specific type** (you can order breakfast, lunch, or dinner in that restaurant)

What’s the difference from the **abstract method pattern**?

There seem to be quite a bit of overlap in these two, however, where abstract factory is less specialized and can create many different things, abstract method is all about creating different versions of the same. In the example with the restaurants above, we would be talking about those restaurants that can serve lunch only. So, we would come in and say “I want launch special”. And, depending on the restaurant, we would get a particular version of that lunch.

5 - Implementation

Codes in the Github.