Today we are going to discuss really exciting topic .which is Factory design pattern. In Software Development there are basically a lot of different patterns we can use.

The problem is people write code kind of procedurally top-down, write a bunch of methods and functions and classes .. the code works. But then down the line may be their product owner somebody can be tell us that they need to make a change and now all of sudden because there has been all this class dependency within their code things have broken and they make one change in one place and it doesn’t necessarily sync(sink) up with all of the other places that they wrote that same exact code. Because there has code duplication and really just inconsistencies with the way. they are writing their code and instantiating classes.

so today I am present the **factory design pattern** and there is kind of this jargon filled definition by the **Gang of Four** which are the original four authors of the classic design patterns elements of reusable object-oriented software book. And they essentially **define the factory design pattern** as an interface for creating an objection which you let the subclasses decide which class to instantiate (instensciet) and then the factory allows us to basically defer (diför) class instantiation to our respective subclasses.

Design pattern is nothing but way of design our classes, relationship between our classes in order to solve our business problems in accordance with the object oriented design principles.

Developers have solved the problems while adhering to (bagli kalarak) oob design principle are called as oop design patterns.

Till now was the beginning of my presentation. I want to finish to introduction part of Design Pattern subject. Now

Let me show you some example in the practice.

Wenn I first develop a tennis complex application.

They was one member. Member will booking using one mode of payment for one cord. When you are speak to director of tennis club. He would say you that they don’t have a simple member they have lot of membership. To develop this functionality in the book and play application for a club.

Here *member* class became abstract class. Because there was no simple *member.* Member had methods like *register notify*

So there were specific types of members, lifetime member, annual member and a temporary member. There are different from each other. Membership duration. There are different discounts. Annual numbers will have a 50 percent off on regular tennis cord bucking. Life time members will have 90 percent off. Obviously there will membership fee.

We had one more class which was **membership manager** so this class was for creating new memberships changing membership types, or somebody else to cancel membership.

it has a method createMemberschip.

What we are suppose to do by following oop design principle some block changing in memberschip manager,

Created another Memberfactory class. It will have one method called as create member so this will return member. It will have some type argument.

We move instantiation logic. Where we don't know which specific kind of object we want to create. We move that logic in to class. so it is not to responsibility of memberschipManager to instantiate object for member type. so we move that in to member factory class. this type of classes was purpose is to carry of the responsibility of generating specific type of object base don't certain condition are called as Factory classes. Factory.

it will be a class was purpose will be to generate specific kind of object for a super type. base on certain conditions.

factory method which is actually generating (doing) that job of instantiating specific type for a super type this kind of methods are called as factory methods.

this is our factory class and this is the factory method inside it.

advantage is which I am going to say that

may be you need this factory in more than one application sometimes.

factory design: there are certain super types which are required in variety of operations in variety of programs so the instantiation of specific types is required multiple time in our program. so it avoids code duplication. because it is happing only one place you can make changes at one place that is the advantage.

In this case this (member) it will be called as product.

(Anmeldung oder abstruct class, ) the specific super type. These(tempmember,lifetimemember) are called as concrete products (student, Lehre, )

And this (membership Manager) class which is consuming the factory is called as client oft he factory.

Factory’s supposed to create specific types is called as product.

Concrete product the specific types which responsibility of creating moves lies upon the factory.

These are the terminologies regarding factory.

Factory are used to define design patterns.