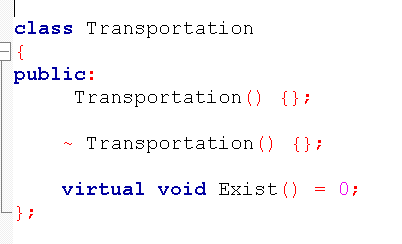
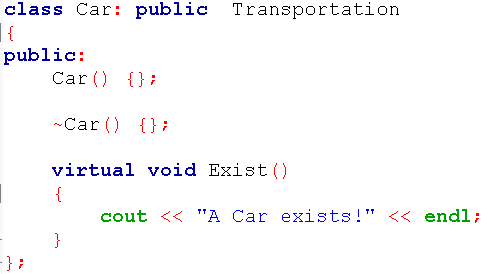
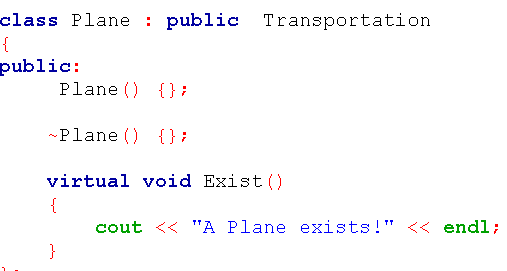
**FACTORY METHOD**

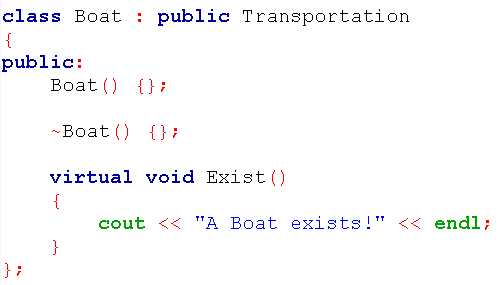
1. **Example relate to Factory Method:**

- A transportation company with many types of means:









1. **Solve without FDP:**

Whenever we need to instantiate an object, we use the new operator as usual:

Transportation \*p1=new Car;

Transportation \*p2=new Plane;

Transportation \*p3=new Boat;

1. **Difficulty of the above way:** difficult to manage initialize the object.

We need an object that can handle the initialization of these objects, just pass the request, then that object will return the corresponding object. To do that we need to use a Factory object.

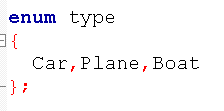
1. **Factory Method [ OverView]**

Factory method is a [creational design pattern](https://www.geeksforgeeks.org/design-patterns-set-1-introduction/), i.e., related to object creation. In Factory pattern, we create object without exposing the creation logic to client and the client use the same common interface to create new type of object.  
The idea is to use a static member-function (static factory method) which creates & returns instances, hiding the details of class modules from user.

1. **Solve above problem with Factory Method**

Source code:

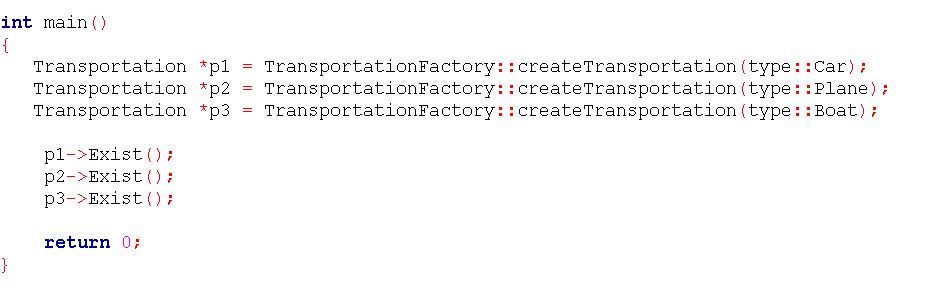
*Step 1: Create enum to save type*



*Step 2: Build Factory*



*Step 3: Using Factory:*



1. **Other problems solved with Factory Method:**

Ex01: ‘Drawing’ system, depending on user’s input, different pictures like square, rectangle, circle can be drawn. Here we can use factory method to create instances depending on user’s input.

Ex02:

If a Restaurant serves Veg food, Non-Veg food, and Italian food.

Treat VegFood, NonVegFood, and ItalianFood as three classes whose superclass is Food.

If a customer asks "Veg" then Factory method would return the "VegFood" class.

1. **Advantages and disadvantages**

*+ Advantages:*

\* Allows you to hide implementation of an application seam (the core interfaces that make up your application).

\* Allows you to easily test the seam of an application (that is to mock/stub) certain parts of your application so you can build and test the other parts

\* Allows you to change the design of your application more readily, this is known as loose coupling.

\* It makes the application more customizable.

*+ Disadvantages:*

\* Makes code more difficult to read as all of your code is behind an abstraction that may in turn hide abstractions.

\* Reduced readability due to increased abstraction.

\* Applicable only for families of classes.

**@REFERENCE:**

<http://uttakarshtikku.blogspot.com>

<https://www.quora.com/>

<https://www.geeksforgeeks.org/>

<https://www.stdio.vn/>