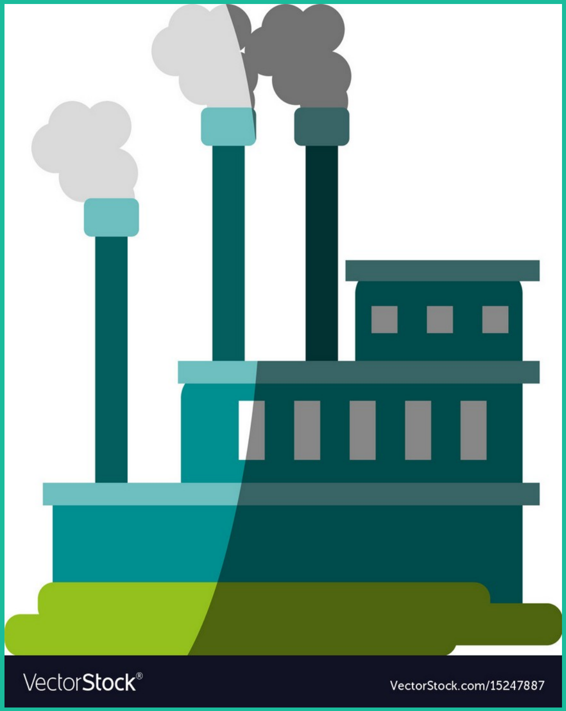
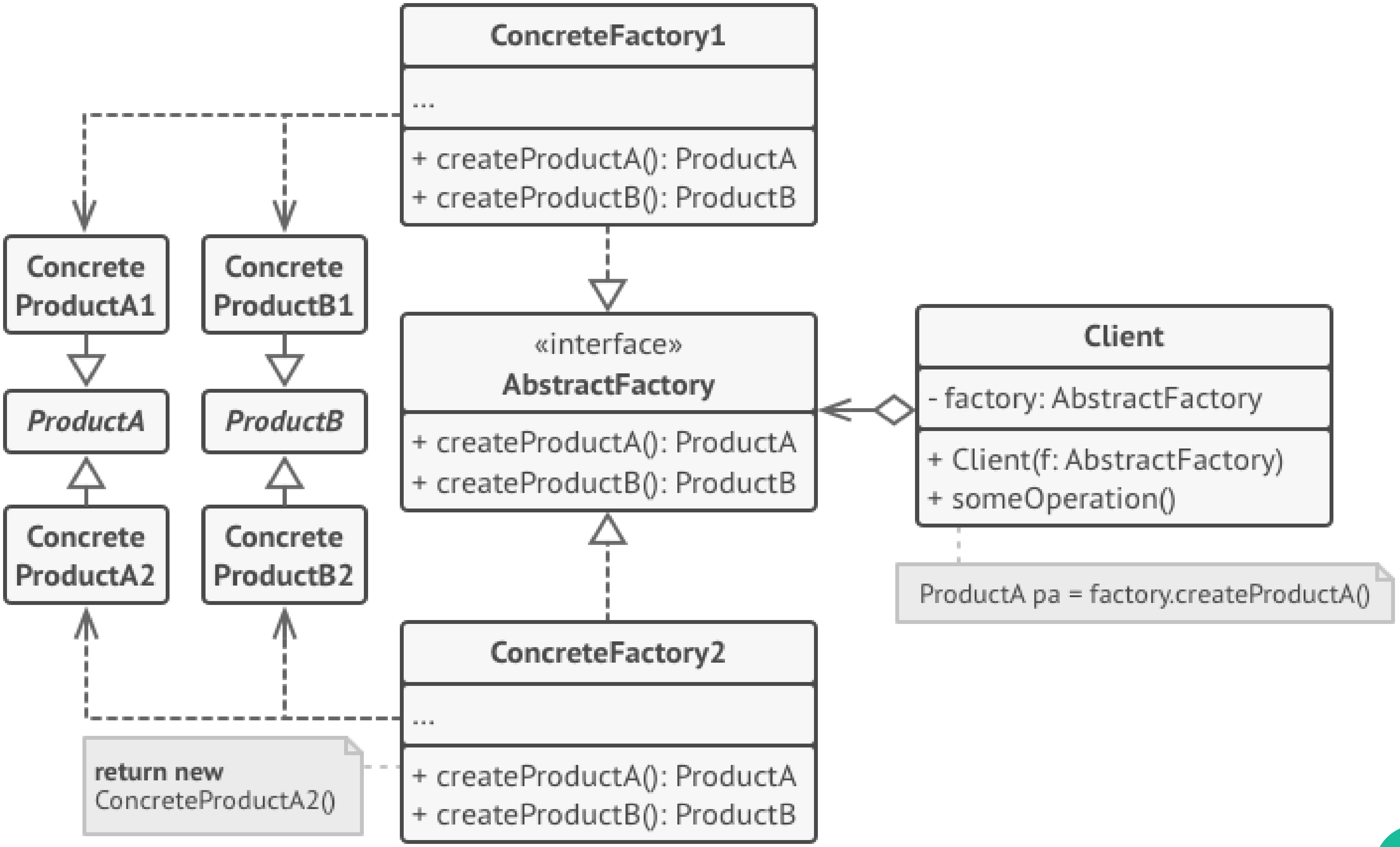
Abstract Factory pattern



Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.

If you have several object with similar properties but need different behavior then Abstract Factory is for you.
And if you are client then you will have to know only properties but not the differences.

Abstract Factory


ConcreteFactory1
...
+ createProductA(): ProductA
+ createProductB(): ProductB
Concrete ProductA1
Concrete ProductB1
ProductA
ProductB
Concrete ProductA2
Concrete ProductB2
«interface»
AbstractFactory
+ createProductA(): ProductA
+ createProductB(): ProductB
Client
- factory: AbstractFactory
+ Client(f: AbstractFactory)
+ someOperation()
ProductA pa = factory.createProductA()
ConcreteFactory2
...
+ createProductA(): ProductA
+ createProductB(): ProductB
return new ConcreteProductA2()

Abstract Factory code

```
abstract class Weapon
{
    public abstract void Hit();
}
```

```
class Bow : Weapon
{
    public override void Hit()
    {
        Console.WriteLine("Shoot with a bow");
    }
}
```

```
class Sword : Weapon
{
    public override void Hit()
    {
        Console.WriteLine("Hit with a sowrd");
    }
}
```

```
abstract class Movement
{
    public abstract void Move();
}
```

```
class FlyMovement : Movement
{
    public override void Move()
    {
        Console.WriteLine("Flying");
    }
}
```

```
class RunMovement : Movement
{
    public override void Move()
    {
        Console.WriteLine("Running");
    }
}
```

Abstract Factory code

```
abstract class HeroFactory
{
    public abstract Movement CreateMovement();
    public abstract Weapon CreateWeapon();
}
```

```
class ElfFactory : HeroFactory
{
    public override Movement CreateMovement()
    {
        return new FlyMovement();
    }

    public override Weapon CreateWeapon()
    {
        return new Bow();
    }
}
```

```
class VoinFactory : HeroFactory
{
    public override Movement CreateMovement()
    {
        return new RunMovement();
    }

    public override Weapon CreateWeapon()
    {
        return new Sword();
    }
}
```

```
class Hero
{
    private Weapon _weapon;
    private Movement _movement;
    public Hero(HeroFactory factory)
    {
        _weapon = factory.CreateWeapon();
        _movement = factory.CreateMovement();
    }
    public void Run()
    {
        _movement.Move();
    }
    public void Hit()
    {
        _weapon.Hit();
    }
}
```

Abstract Factory code & result

```
static void Main(string[] args)
{
    Hero elf = new Hero(new ElfFactory());
    Hero hero = new Hero(new VoinFactory());

    elf.Hit();
    elf.Run();
    Console.WriteLine("------------------");
    hero.Hit();
    hero.Run();

    Console.ReadKey();
}
```

```
Shoot with a bow
Flying
------------------
Hit with a sowrd
Running
```

Thanks everyone!

Code aviable at:

https://github.com/SeniorSSS/DesignPatterns/tree/master/AbstractFactory

Author: Janis Strazdins