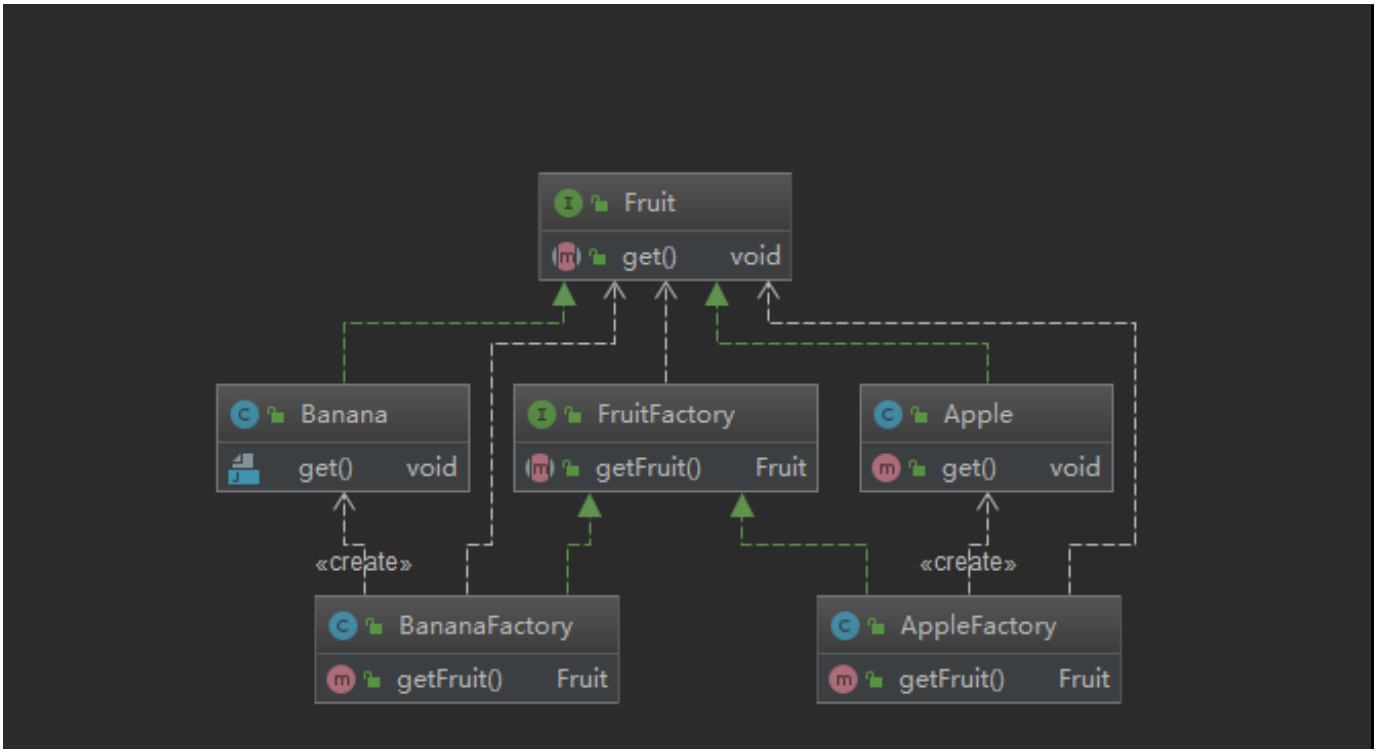
类图



为什么会需要工厂方法模式?

因为简单工厂模式违背了开闭原则。假如我需要添加一个采集西瓜的方法。在简单工厂中，我需要加入对西瓜的判断。

```
1   public static Fruit getFrult(String type) throws Exception {
2         if (type.equals("apple")) {
3             return Apple.class.newInstance();
4         } else if (type.equals("banana")) {
5             return Banana.class.newInstance();
6         } else if(type.equals("watermelon")){
7             return new Watermelon();
8         } else {
9             return null;
10        }
11    }
```

所以工厂方法模式就诞生了。他是简单工厂的升级。

具体写法：

1.抽象出来一个工厂

```
public interface FruitFactory {
    Fruit getFruit();
}
```

2.创建采集苹果的工厂

```
public class AppleFactory implements FruitFactory {
    @Override
    public Fruit getFruit() {
        return new Apple();
    }
}

```

3.创建采集香蕉的工厂

```
public class BananaFactory implements FruitFactory {
    @Override
    public Fruit getFruit() {
        return new Banana();
    }
}

```

4.如果需要扩展，就可以很方便的扩展。例如采集西瓜。

```
public class WatermelonFactory implements FruitFactory {
    @Override
    public Fruit getFruit() {
```

```
        return new Watermelon();
    }
}
```

5.测试

```
public class Test {
    public static void main(String[] args) throws Exception{

        AppleFactory appleFactory = new AppleFactory();
        Fruit fruit = appleFactory.getFruit();
        fruit.get();

        BananaFactory bananaFactory = new BananaFactory();
        Fruit fruit1 = bananaFactory.getFruit();
        fruit1.get();

        WatermelonFactory watermelonFactory = new
WatermelonFactory();
        Fruit fruit2 = watermelonFactory.getFruit();
        fruit2.get();

    }
}

结果:
采集苹果
采集香蕉
采集西瓜
```