Team #3

Design Pattern

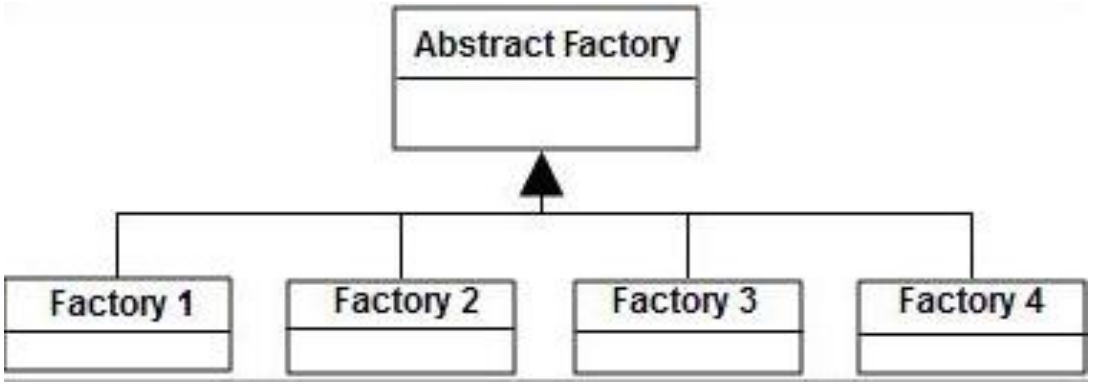
Abstract Factory

- **Bill Capps**
- **Doug Hoskisson**
- **Rakan Alanazi**

UMKC

Abstract Factory Pattern

An abstract factory is a factory that returns factories, that can be used to create sets of related objects .



UMKC

Abstract Factory Pattern

- Abstract Factory Pattern provides a way to encapsulate a group of individual factories that have a common theme.

- Abstract Factory is a way to decouple our clients from concrete classes.

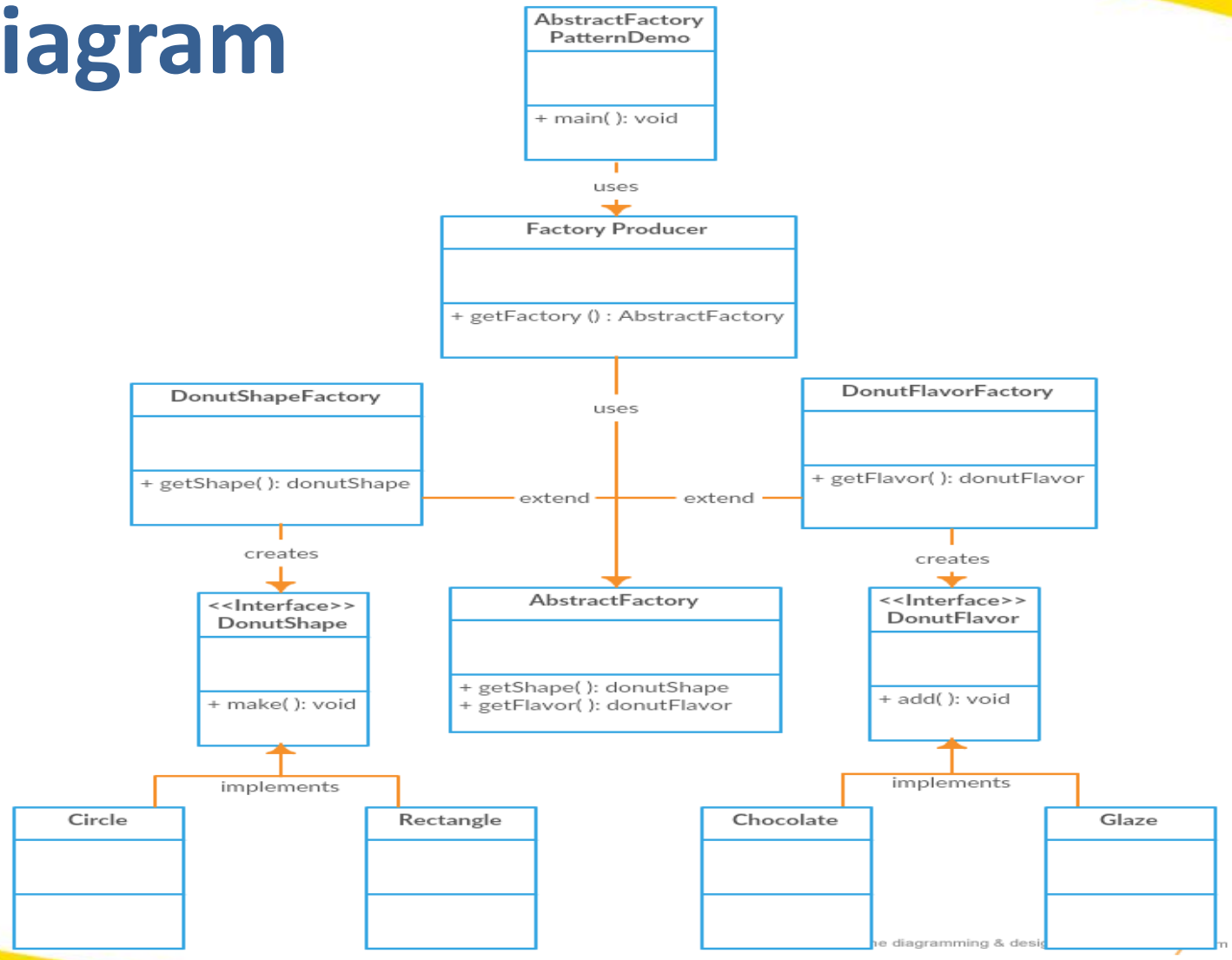
UMKC

Pros and Cons

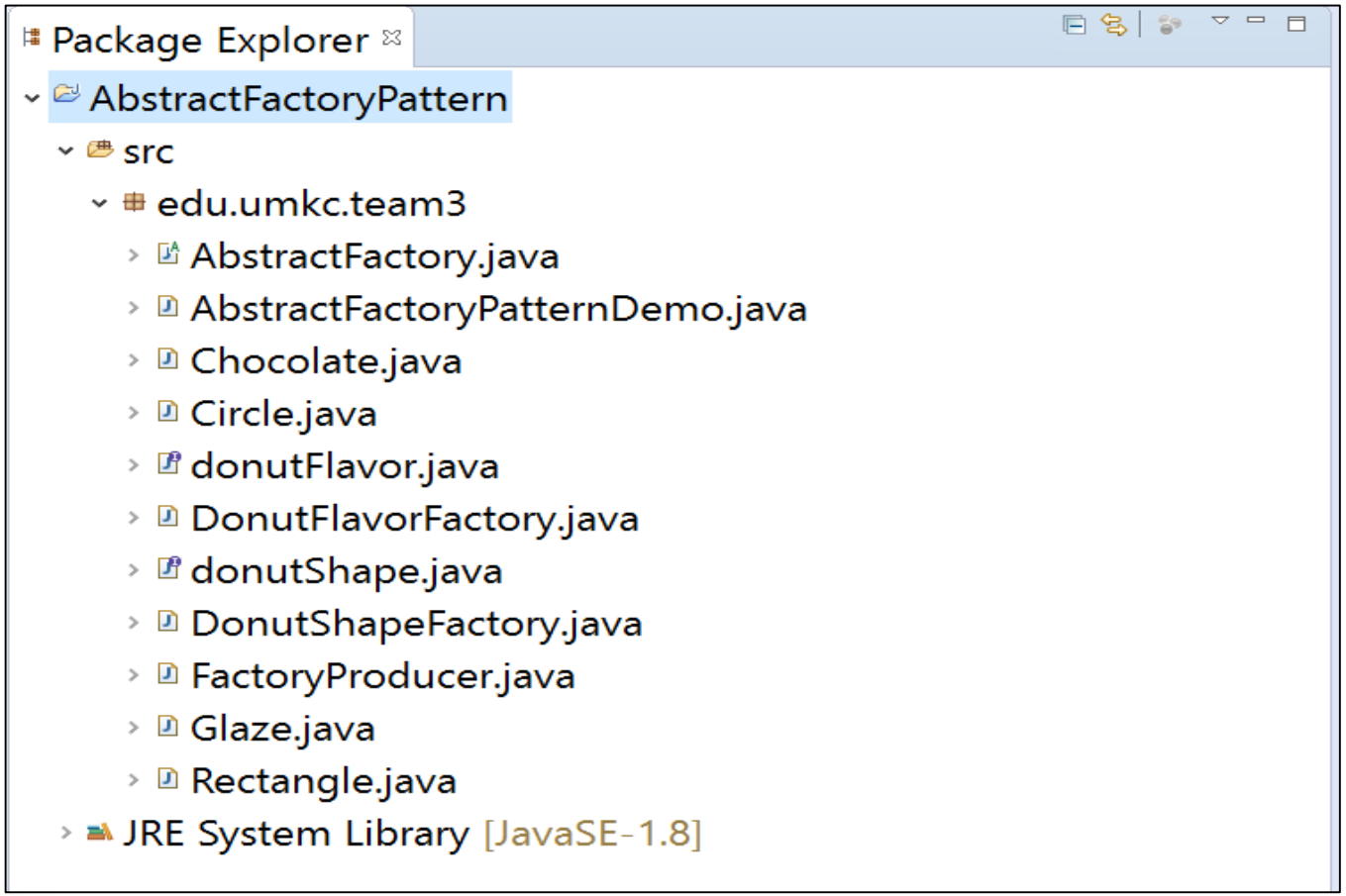
- **Pros**:
  It's possible to change one concrete class for another without changing the code that accesses them.
  Cons:
  There is more complexity in the beginning stages of writing code.

UMKC

UML Diagram


AbstractFactory PatternDemo
+ main( ): void
uses
Factory Producer
+ getFactory () : AbstractFactory
uses
DonutShapeFactory
+ getShape( ): donutShape
extend
extend
DonutFlavorFactory
+ getFlavor( ): donutFlavor
creates
creates
<<Interface>> DonutShape
+ make( ): void
AbstractFactory
+ getShape( ): donutShape
+ getFlavor( ): donutFlavor
<<Interface>> DonutFlavor
+ add( ): void
implements
implements
Circle
Rectangle
Chocolate
Glaze

Implementation



DonutFlavorFactory Class

```java
package edu.umkc.team3;

public class DonutFlavorFactory extends AbstractFactory {

        @Override
        public donutShape getShape(String shapeType){
            return null;
        }

        @Override
        donutFlavor getFlavor(String flavor) {

            if(flavor == null){
                return null;
            }

            if(flavor.equalsIgnoreCase("CHOCOLATE")){
                return new Chocolate();

            }else if(flavor.equalsIgnoreCase("GLAZE")){
                return new Glaze();


            }

            return null;
        }
    }
```

DonutShapeFactory Class

```java
package edu.umkc.team3;

public class DonutShapeFactory extends AbstractFactory {

        @Override
        public donutShape getShape(String shapeType){

            if(shapeType == null){
                return null;
            }

            if(shapeType.equalsIgnoreCase("CIRCLE")){
                return new Circle();

            }else if(shapeType.equalsIgnoreCase("RECTANGLE")){
                return new Rectangle();

            }

            return null;
        }

        @Override
        donutFlavor getFlavor(String flavor) {
            return null;
        }
    }
```

Interfaces

```java
package edu.umkc.team3;

public interface donutFlavor {

            void add();

}
```

```java
package edu.umkc.team3;

public interface donutShape {


        void make();
}
```

Chocolate and Glaze Classes

```java
package edu.umkc.team3;

public class Chocolate implements donutFlavor {

        @Override
        public void add() {
            System.out.println(" Add Chocolate flavor");
        }
    }
```

```java
package edu.umkc.team3;

public class Glaze implements donutFlavor {

        @Override
        public void add() {
            System.out.println("add Glaze flavor");
        }
    }
```

Rectangle and Circle Classes

```java
package edu.umkc.team3;

    public class Rectangle implements donutShape {


            public void make() {
                System.out.println("make Rectangle donut");
            }
        }
```

```java
package edu.umkc.team3;

public class Circle implements donutShape {

        public void make() {
            System.out.println(" make circle donut");
        }
    }
```

AbstractFactory and FactoryProducer

```java
package edu.umkc.team3;


public abstract class AbstractFactory {
        abstract donutShape getShape(String shape);
        abstract donutFlavor getFlavor(String flavor) ;
    }
```

```java
package edu.umkc.team3;

public class FactoryProducer  {
    public static AbstractFactory getFactory(String choice){

            if(choice.equalsIgnoreCase("SHAPE")){
               return new DonutShapeFactory();

            }else if(choice.equalsIgnoreCase("FLAVOR")){
               return new  DonutFlavorFactory();
            }

            return null;
        }
    }
```

Demo Class

```java
package edu.umkc.team3;

public class AbstractFactoryPatternDemo {

	public static void main(String[] args) {

		//get donut shape factory
		AbstractFactory shapeFactory = FactoryProducer.getFactory("SHAPE");

		//get an object of Shape Circle
		donutShape shape1 = shapeFactory.getShape("CIRCLE");

		//call make method of Shape Circle
		shape1.make();

		//get an object of Shape Rectangle
		donutShape shape2 = shapeFactory.getShape("RECTANGLE");

		//call make method of Shape Rectangle
		shape2.make();


		//get donut flavor factory
		AbstractFactory flavorFactory = FactoryProducer.getFactory("FLAVOR");

		//get an object of Flavor Chocolate
		donutFlavor flavor1 = flavorFactory.getFlavor("CHOCOLATE");

		//call add method of chocolate
		flavor1.add();

		//get an object of Flavor Glaze
		donutFlavor flavor2 = flavorFactory.getFlavor("GLAZE");

		//call add method of glaze
		flavor2.add();

	}
}
```

The Output

Problems Javadoc Declaration **Console**

```
<terminated> AbstractFactoryPatternDemo [Java Application]
Make Circle Donut
Make Rectangle Donut
Add Chocolate Flavor
Add Glaze Flavor
```

UMKC