Abstract Factory

Linda Marshall and Vreda Pieterse

Department of Computer Science
University of Pretoria

13 August 2014

Overview

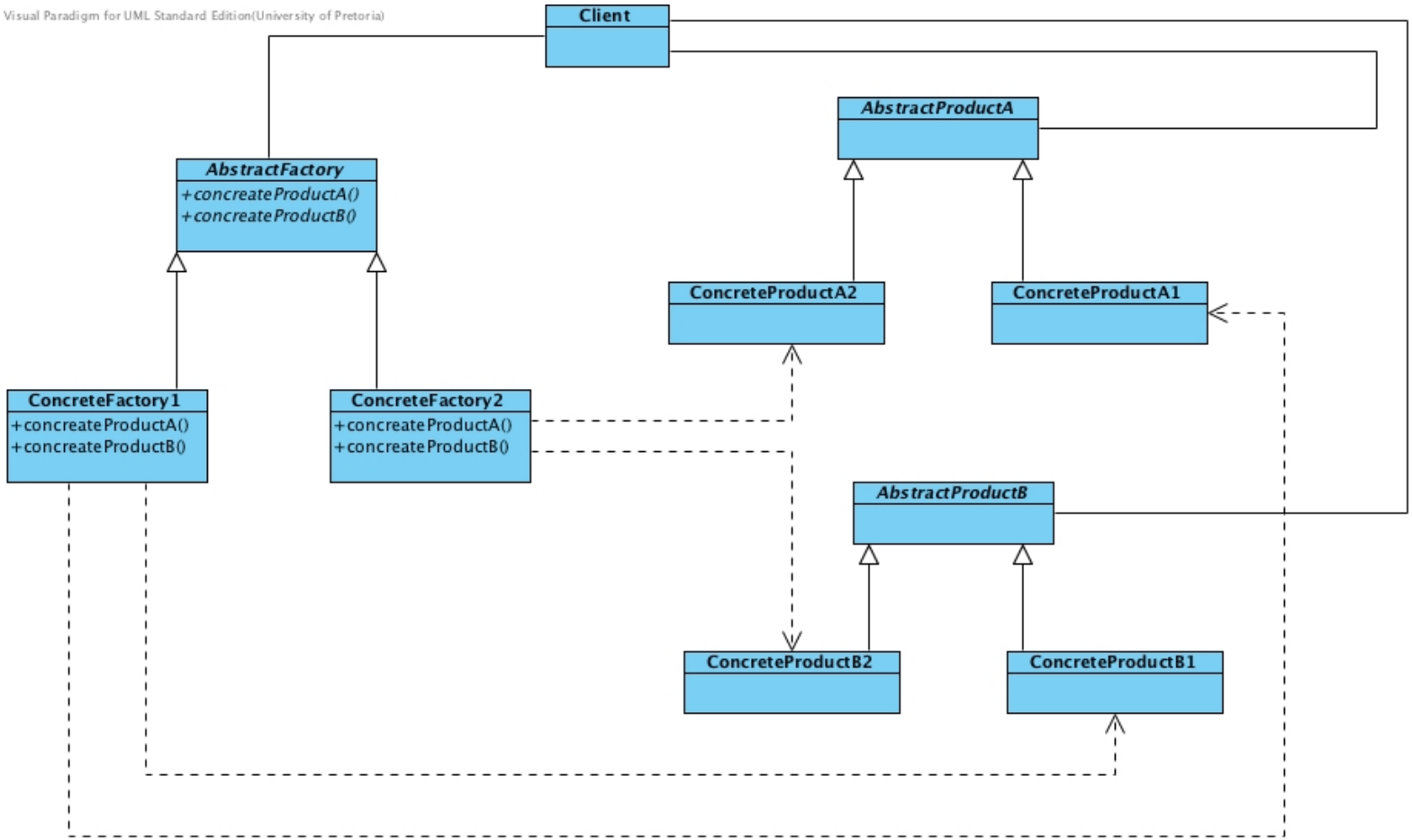
1. Identification
2. Structure
3. Participants
4. Related Patterns
5. Examples
   - Example 1 - Shapes
   - Example 2 - Sweets revisited
   - Example 3 - Surprise

Name and Classification:

Abstract Factory (Object Creational)

Intent:

"Provide an interface for creating families of related or dependent objects without specifying the concrete classes." GoF(87)


Visual Paradigm for UML Standard Edition(University of Pretoria)
Client
AbstractProductA
AbstractFactory
+concreateProductA()
+concreateProductB()
ConcreteProductA2
ConcreteProductA1
ConcreteFactory1
+concreateProductA()
+concreateProductB()
ConcreteFactory2
+concreateProductA()
+concreateProductB()
AbstractProductB
ConcreteProductB2
ConcreteProductB1

- Makes use of factory methods
- Abstract Factory creates a Concrete Factory and therefore does not directly create product
- Concrete Factory creates product

AbstractFactory

- provides an interface to produce abstract product objects

ConcreteFactory

- implements the abstract operations to produce concrete product objects

AbstractProduct

- provides an interface for product objects

ConcreteProduct

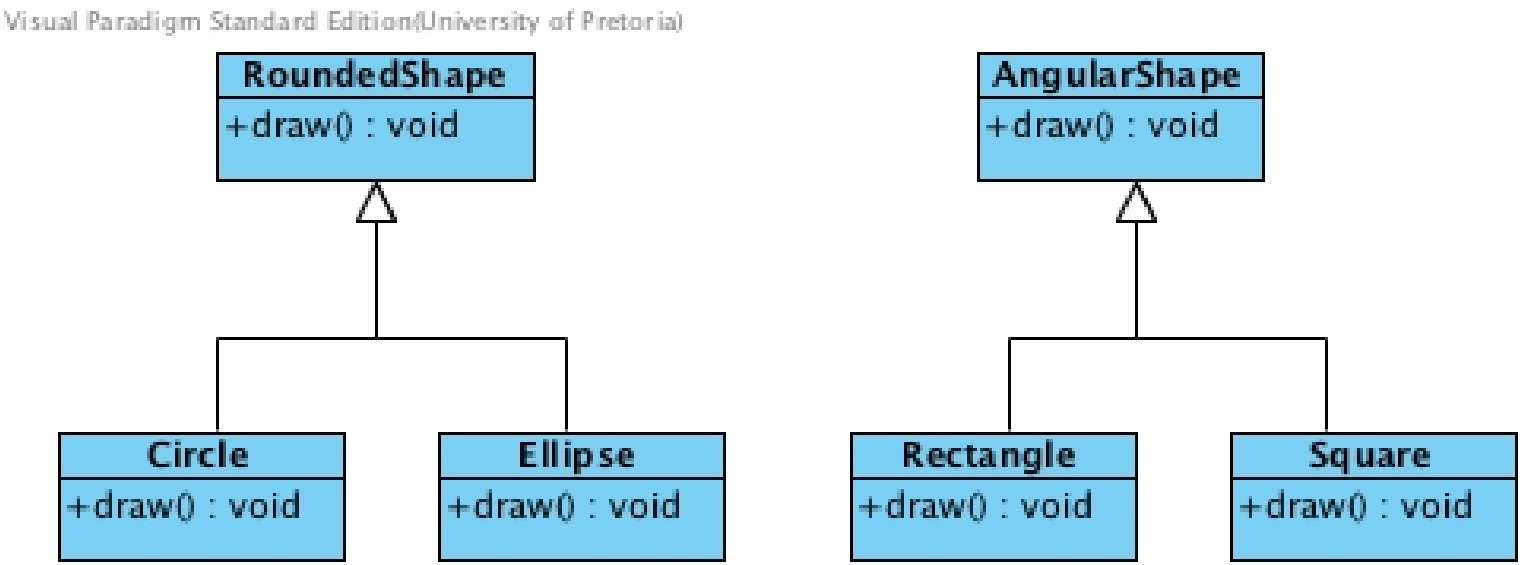
- implements the abstract operations that produce product objects that are created by the corresponding ConcreteFactory

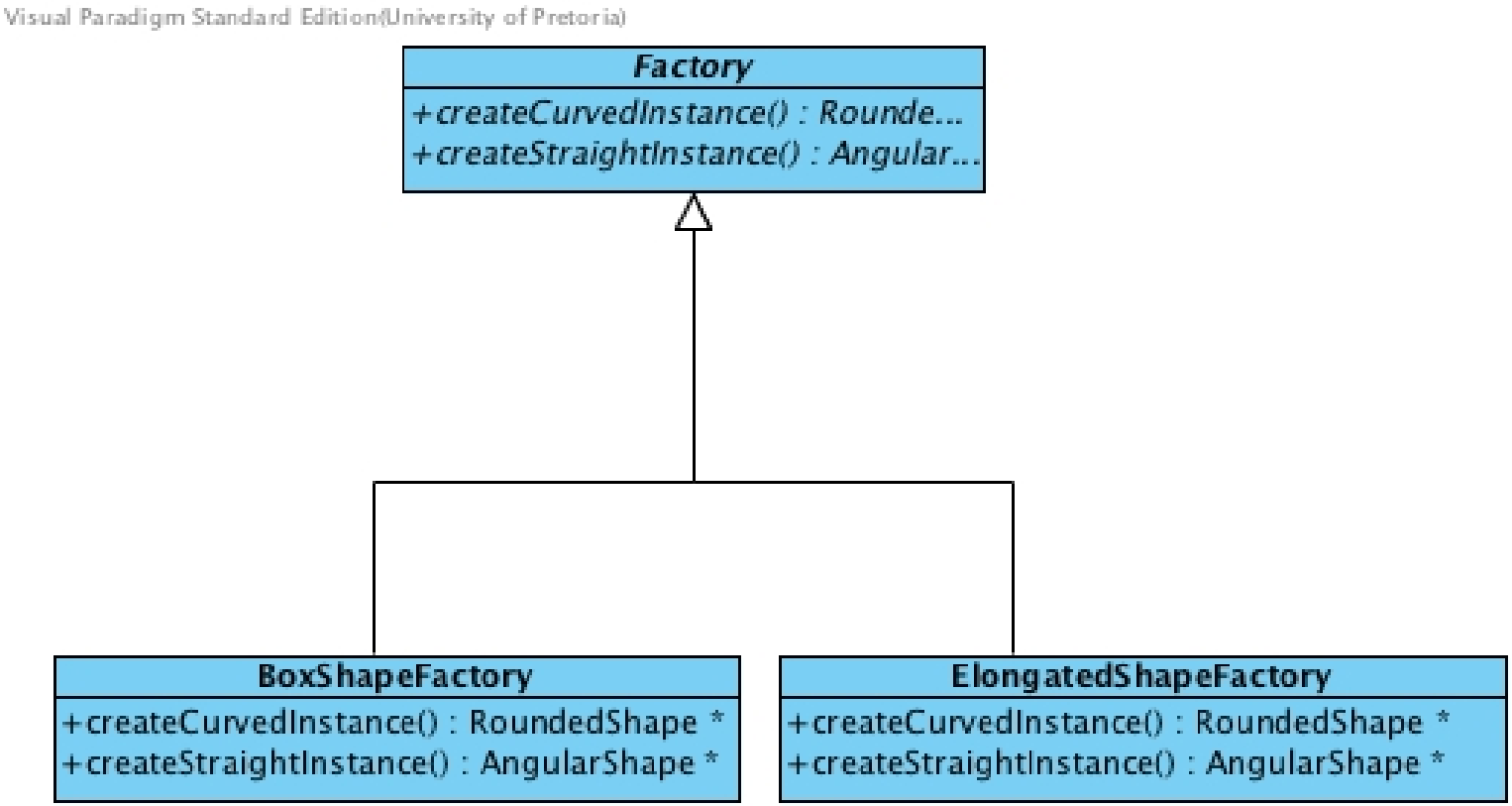
Client

- uses the interfaces defined by AbstractFactory and AbstractProduct

Related Patterns

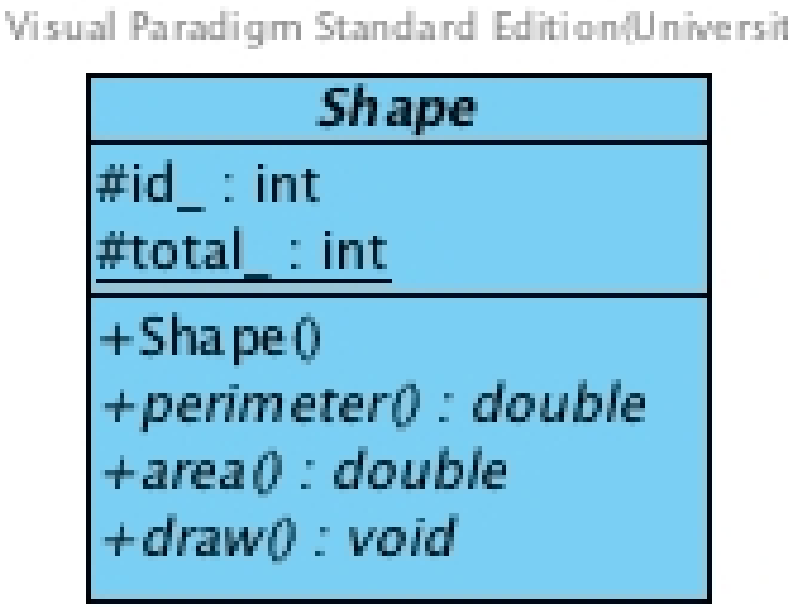
- **Factory Method** (107) and **Prototye**(117) : Are used in the implementation of the Abstract Factory.
- **Template Method** (325): May be used within the factory and product hierarchies.
- **Singleton**(127): Concrete factories may be implemented so.

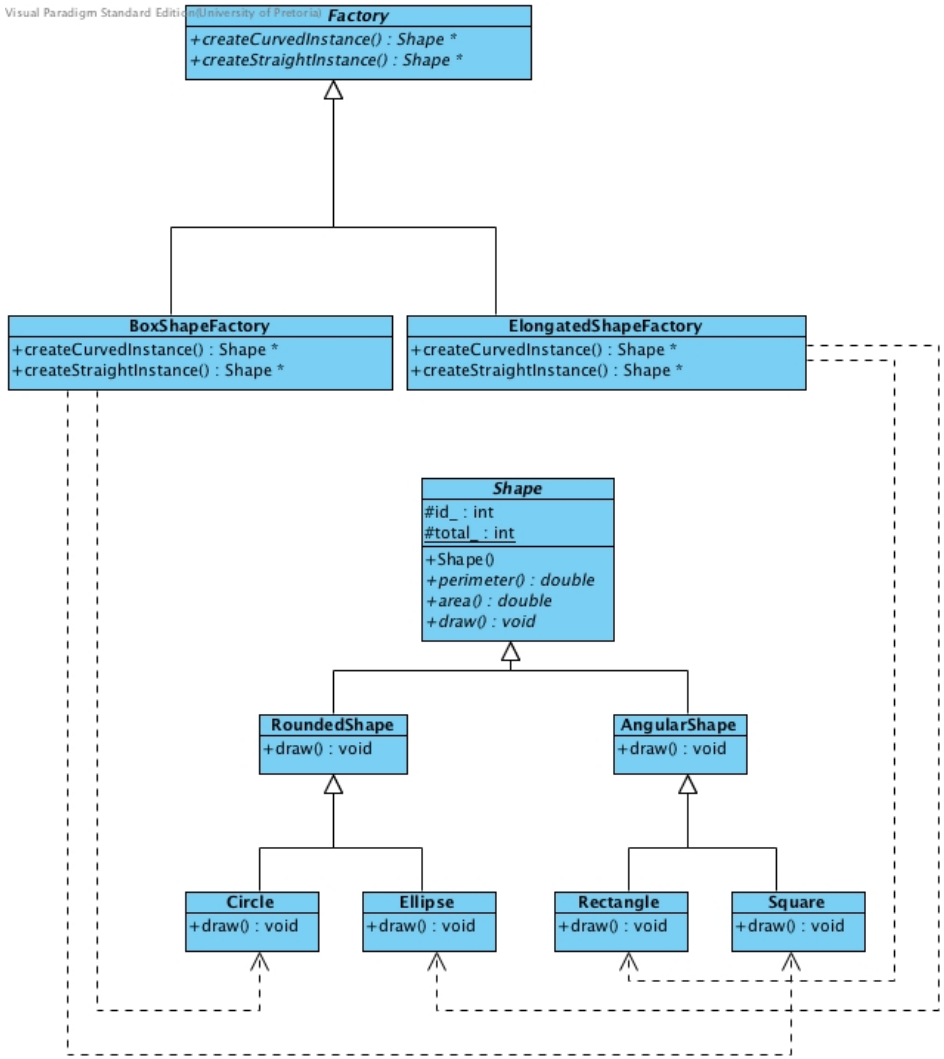

Visual Paradigm Standard Edition(University of Pretoria)
RoundedShape
+draw() : void
AngularShape
+draw() : void
Circle
+draw() : void
Ellipse
+draw() : void
Rectangle
+draw() : void
Square
+draw() : void


Visual Paradigm Standard Edition(University of Pretoria)
Factory
+createCurvedInstance() : Rounde...
+createStraightInstance() : Angular...
BoxShapeFactory
+createCurvedInstance() : RoundedShape *
+createStraightInstance() : AngularShape *
ElongatedShapeFactory
+createCurvedInstance() : RoundedShape *
+createStraightInstance() : AngularShape *

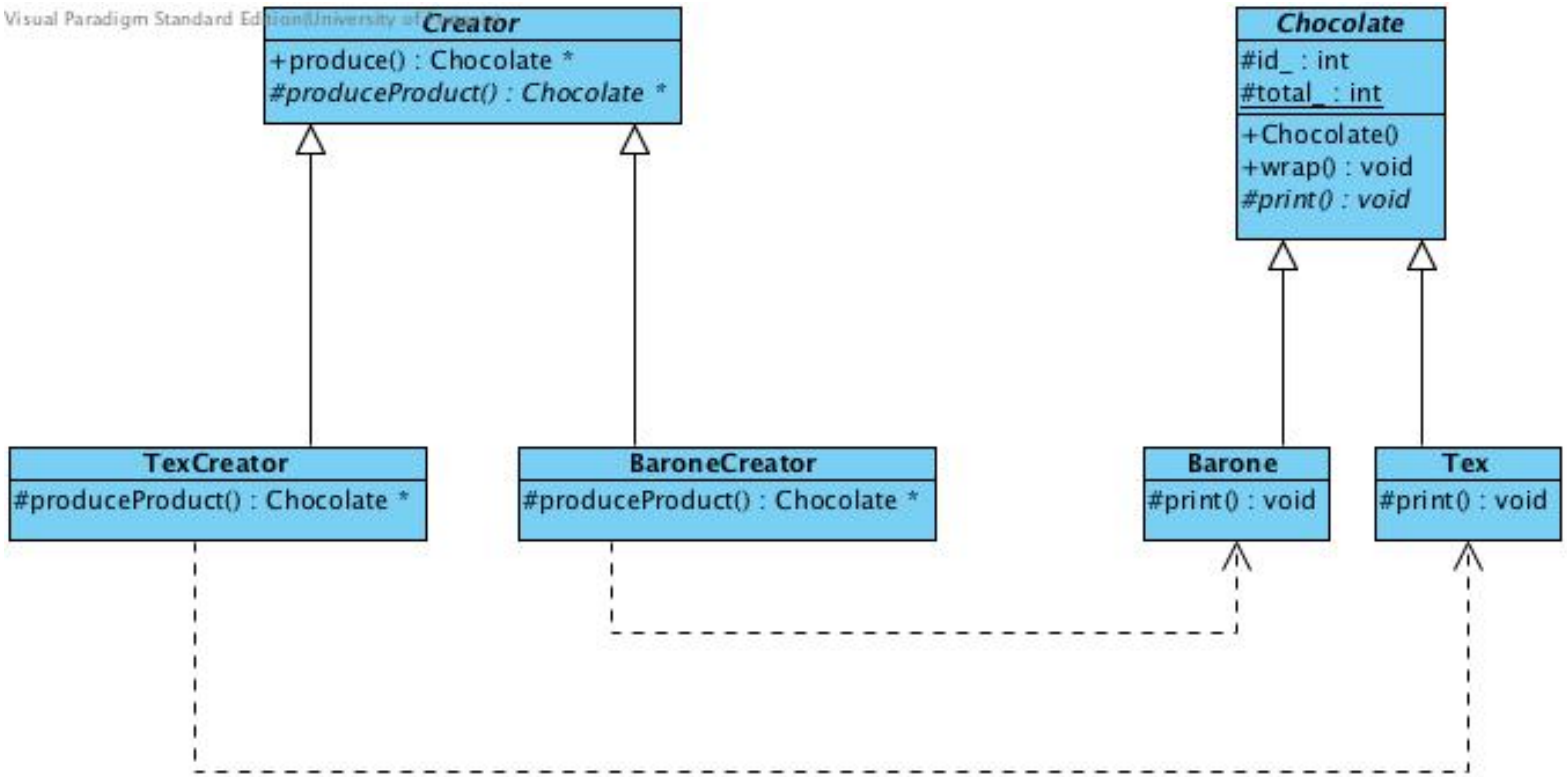
- All shapes have a `draw` function.
- What if we want to add an `area` and a `perimeter` function for each shape?

- Solution: Abstract a `Shape` class and let `RoundedShape` and `AngularShape` inherit from it.

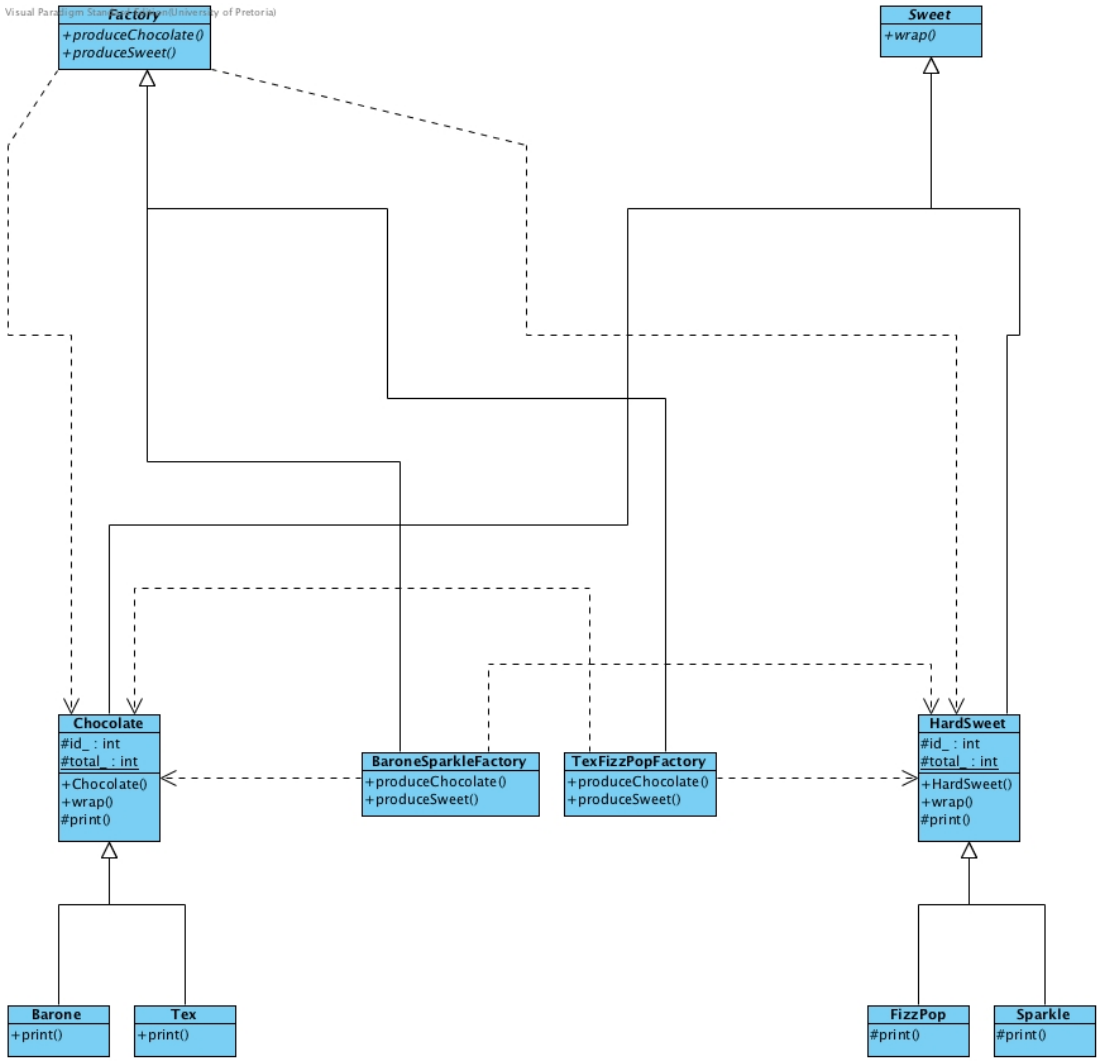



Visual Paradigm Standard Edition(University of Pretoria)
Factory
+createCurvedInstance() : Shape *
+createStraightInstance() : Shape *
BoxShapeFactory
+createCurvedInstance() : Shape *
+createStraightInstance() : Shape *
ElongatedShapeFactory
+createCurvedInstance() : Shape *
+createStraightInstance() : Shape *
Shape
#id_ : int
#total_ : int
+Shape()
+perimeter() : double
+area() : double
+draw() : void
RoundedShape
+draw() : void
AngularShape
+draw() : void
Circle
+draw() : void
Ellipse
+draw() : void
Rectangle
+draw() : void
Square
+draw() : void

The Factory Method



- Assume we want to produce *hard sweets* such as `FizzPops` and `Sparkles`.
- Our Barone factory will begin producing either Barones or Sparkles and our Tex factory both Tex and Fizz Pops.


Factory
+produceChocolate()
+produceSweet()
Sweet
+wrap()
Chocolate
#id_ : int
#total_ : int
+Chocolate()
+wrap()
#print()
BaroneSparkleFactory
+produceChocolate()
+produceSweet()
TexFizzPopFactory
+produceChocolate()
+produceSweet()
HardSweet
#id_ : int
#total_ : int
+HardSweet()
+wrap()
#print()
Barone
+print()
Tex
+print()
FizzPop
#print()
Sparkle
#print()

You have been appointed as the official designer for Katherine Jenkins. You have been commissioned to create her evening wear, sports wear and leisure wear. After you have been revived from your fainting spell, you begin working on your business model. Being a good Computer Scientist you immediately think of design patterns and in particular the Factory Method.

Unfortunately your Factory Method solution is short-lived. You have also been contracted to select her accessories - such as watches, necklaces, rings etc. as well as her shoes. You decide to make your selections from catalogues. Your task is to put outfits together for the evening, when playing sport and when relaxing. How would you apply the Abstract Factory to this problem?

Your friend from Woolies selects designs to be sold under their new luxury branding. You are contracted to make your designs available for the majority of the sizes.