CS 485
Advanced Object Oriented Design

Factories (ch 20 & 23 & 11 & 24)

Spring 2019

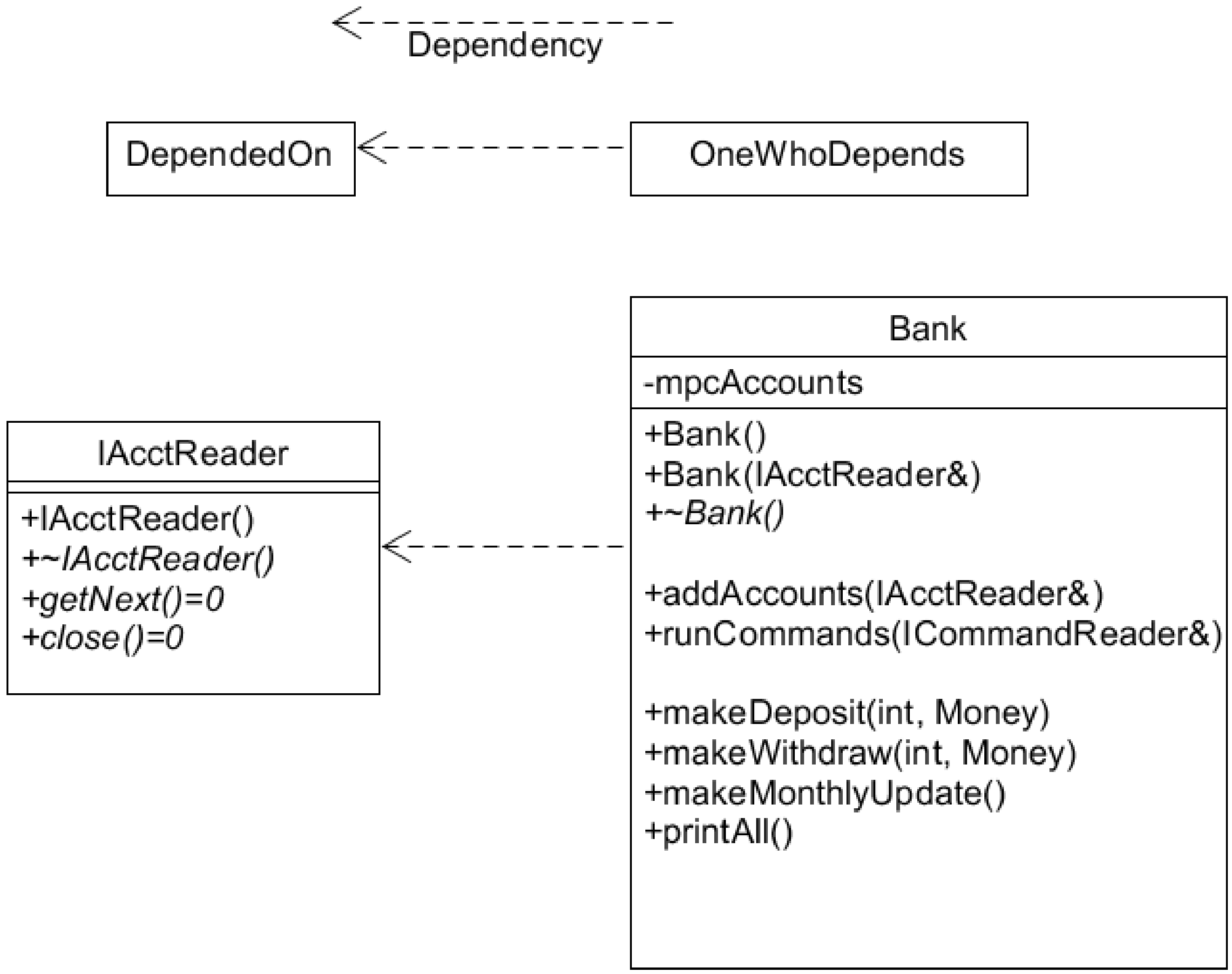
http://www.netobjectives.com/PatternRepository/index.php?title=PatternsByAlphabet

http://www.netobjectives.com/files/books/dpe/design-patterns-matrix.pdf

Review - Patterns

- Creational
  - Factories
- Behavioral
  - Command
  - **Strategy**
  - **Template Method**
- Structural
  - **Facade**

UML Update


Dependency
DependedOn
OneWhoDepends
IAcctReader
+IAcctReader()
+~IAcctReader()
+getNext()=0
+close()=0
Bank
-mpcAccounts
+Bank()
+Bank(IAcctReader&)
+~Bank()
+addAccounts(IAcctReader&)
+runCommands(ICommandReader&)
+makeDeposit(int, Money)
+makeWithdraw(int, Money)
+makeMonthlyUpdate()
+printAll()

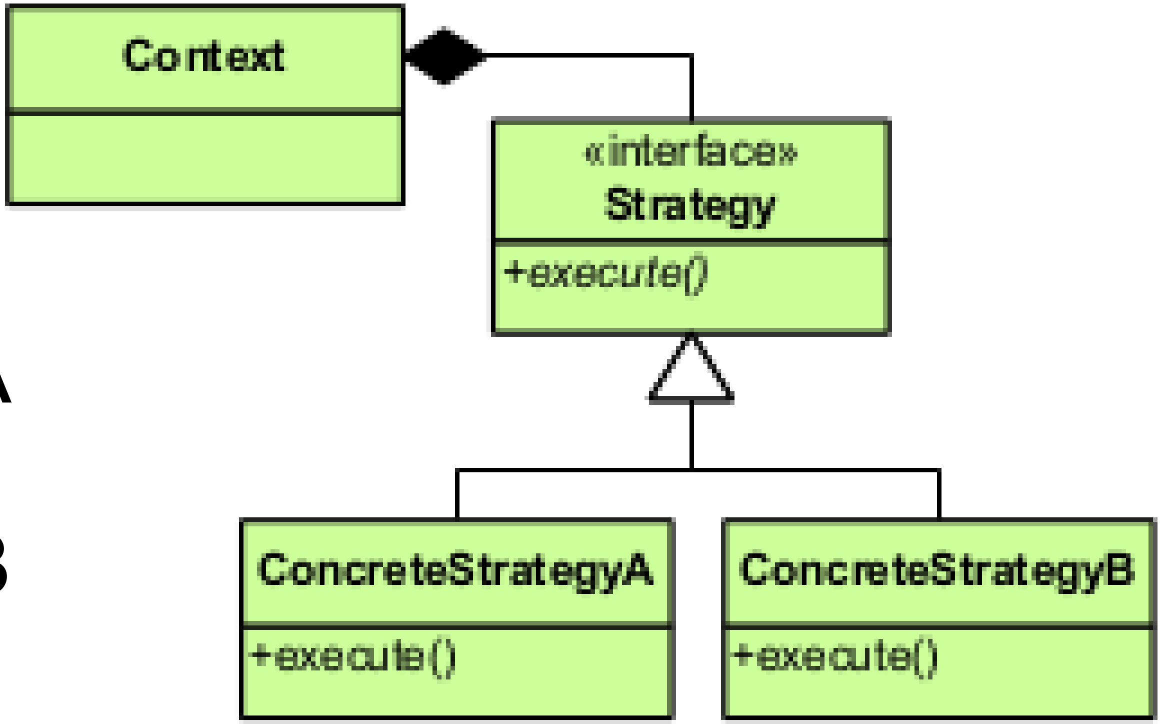
Factories

- Chapter 20 - Overview
- Chapter 23 - Factory Method
- Chapter 11 - Abstract Factories
- Chapter 24 - Summary

- "Objects that make other objects" - Shalloway
- Decouple the creation of objects from the client
  - hide creation details
  - hide concrete classes
  - *allow subclasses to decide how and which concrete classes to instantiate*

Motivation

- Strategy Pattern
- Context knows nothing about ConcreteStrategyA or ConcreteStrategyB
- Who creates the concrete Strategy?
- Could be a Factory!



Guidelines

- Define objects and how they work together
- Write factories that instantiate the correct objects for the right situtation...
- An object should either
  - **make/manage** other objects

    OR
  - **use** other objects

Shalloway, p 351

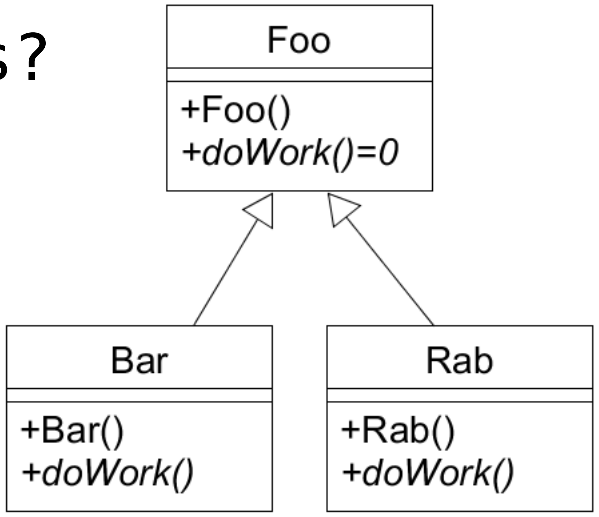
Factory Method

- Single method that creates objects
  - may take a parameter to determine which class to instantiate

- Where does the method live?
  - public static method in a Factory class
  - public static method in the *parent* class
  - private method in a Creator class

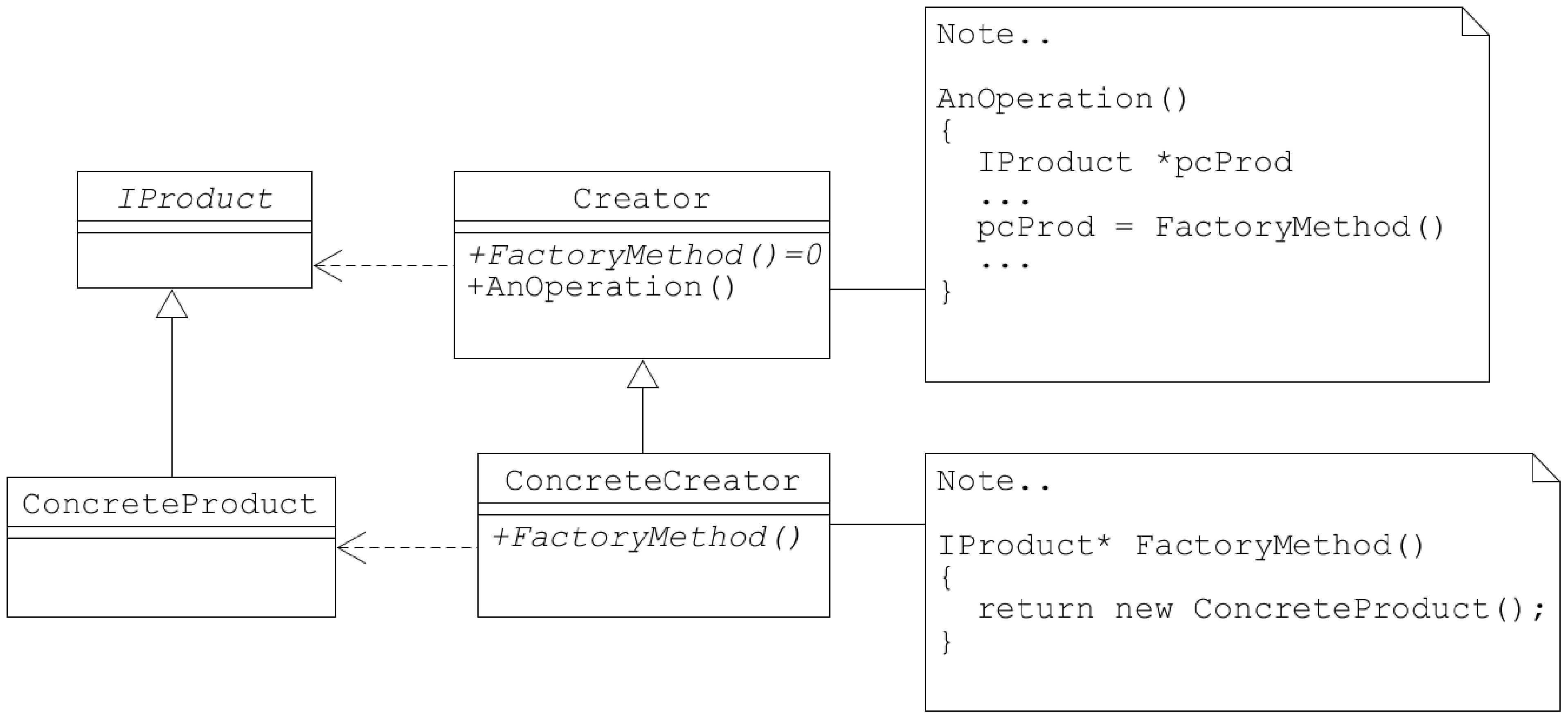
Options you might see in real life

```
class FooFactory // Problems? Benefits?
{
  public:
    static Foo* makeFoo(char fooType);
};
```



```
class Foo // Problems? Benefits? SOLID?
{
  public:
    static Foo* makeFoo(char fooType);
  ...
};
```

Basic Factory Method Pattern



```
AnOperation()
{
   IProduct *pcProd
   ...
   pcProd = FactoryMethod()
   ...
}
```

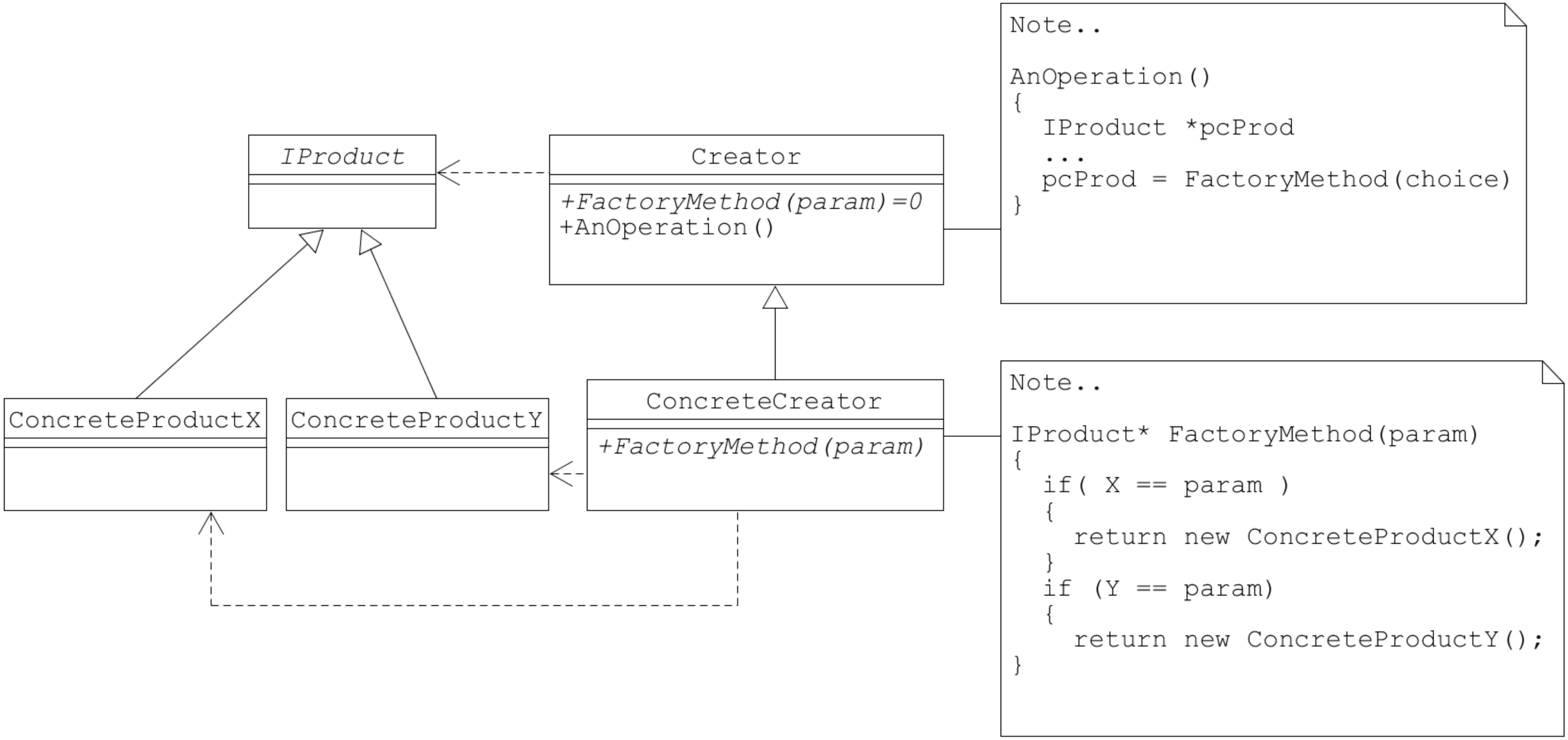
```
IProduct* FactoryMethod()
{
   return new ConcreteProduct();
}
```

`// Problems? Benefits? Advantages? SOLID?`

Note: AnOperation() could be a Template Method.

Shalloway, p 389

Parameterized Factory Method

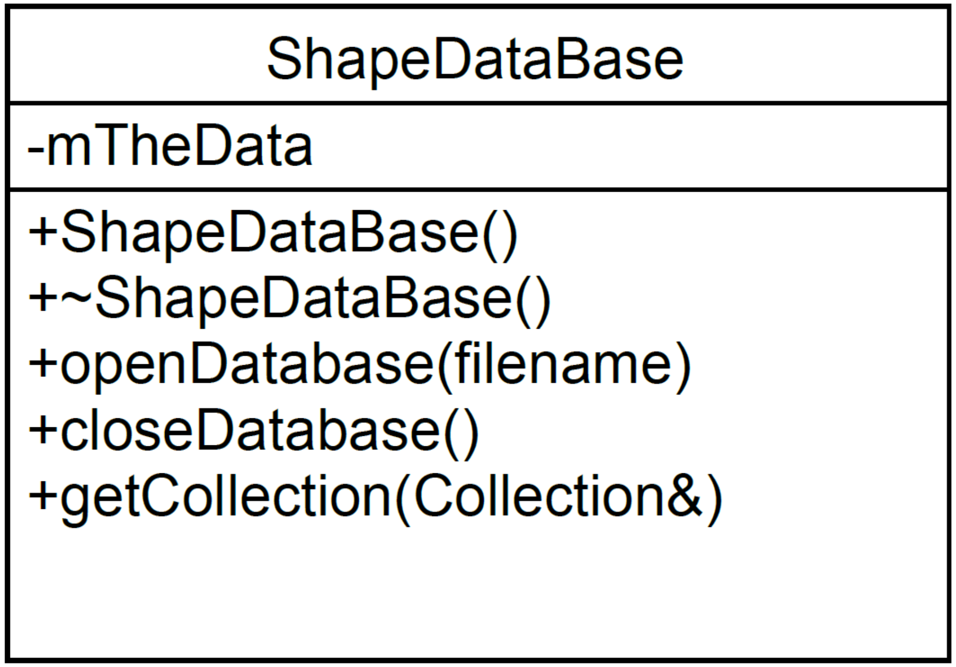


```
Note..

AnOperation()
{
  IProduct *pcProd
  ...
  pcProd = FactoryMethod(choice)
}
```

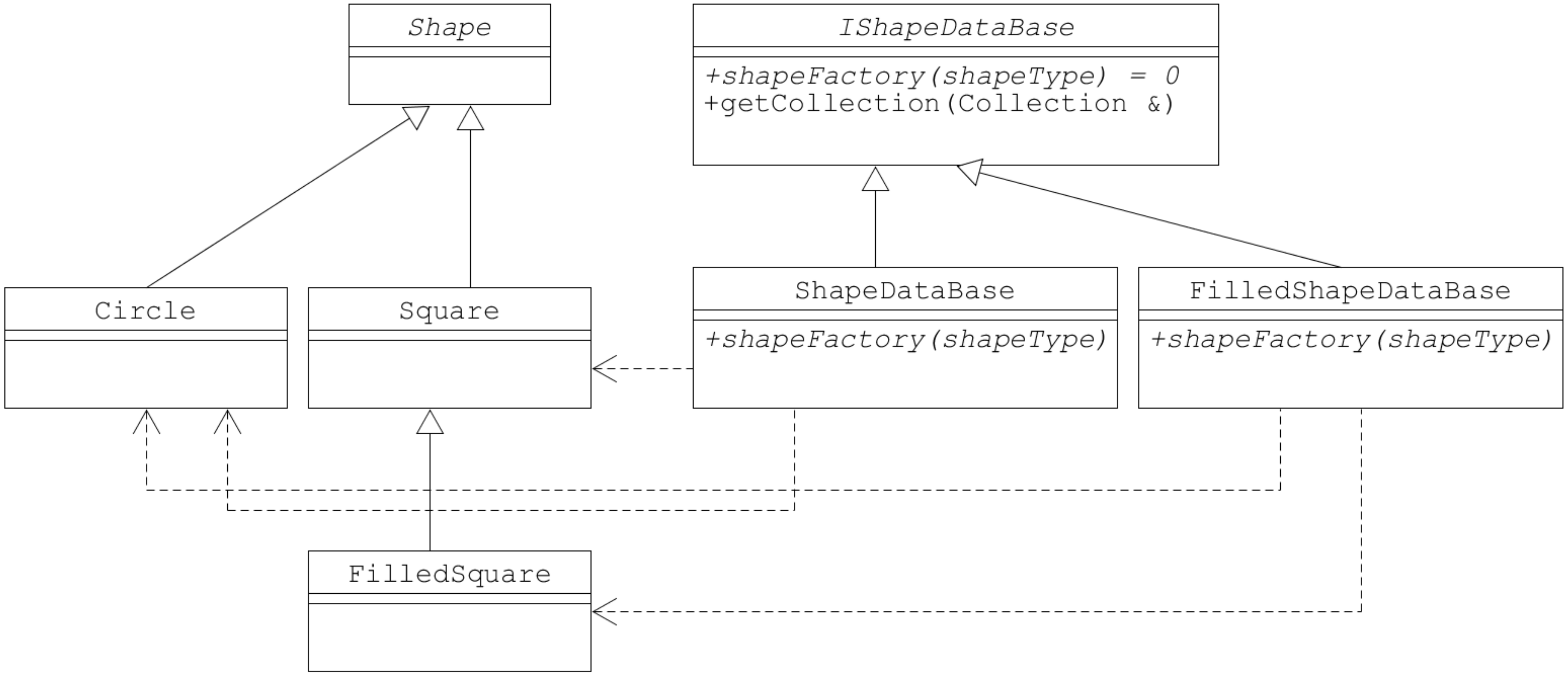
```
Note..

IProduct* FactoryMethod(param)
{
  if( X == param )
  {
    return new ConcreteProductX();
  }
  if (Y == param)
  {
    return new ConcreteProductY();
}
```

Example

- Add default constructors
  - Shape
  - Circle
  - Square
  - Color
- Add IShapesDataBase
  - abstract parent class for ShapesDataBase
- Add Virtual Friend Idiom to Shape heirarchy
  - I explained this backwards Friday. See next slide.



Example



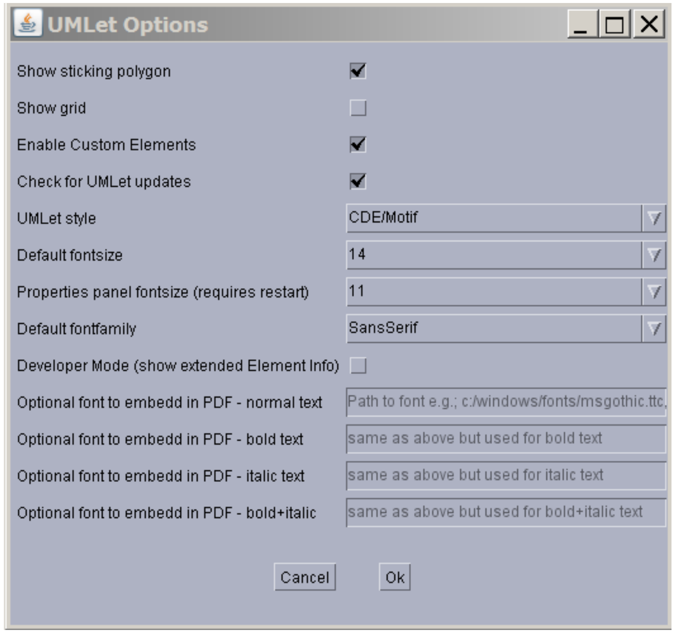
```
// IShapeDataBase contains many other methods
```

```cpp
Shape* ShapeDataBase::shapeFactory (char shapeType)
{
  Shape *pcShape = nullptr;

  switch (shapeType)
  {
  case 'S':
    pcShape = new Square ();
    break;
  case 'C':
    pcShape = new Circle ();
    break;
  }
  return pcShape;
}
```

```cpp
void ShapeDataBase::getCollection (Collection & rcCollection)
{
  char shapeType;
  Shape *pcShape;

  while (mTheData >> shapeType)
  {
    pcShape = shapeFactory (shapeType);
    if (nullptr != pcShape)
    {
      mTheData >> *pcShape;
      rcCollection.addShape (pcShape);
    }
  }
}
```

Abstract Factory Pattern

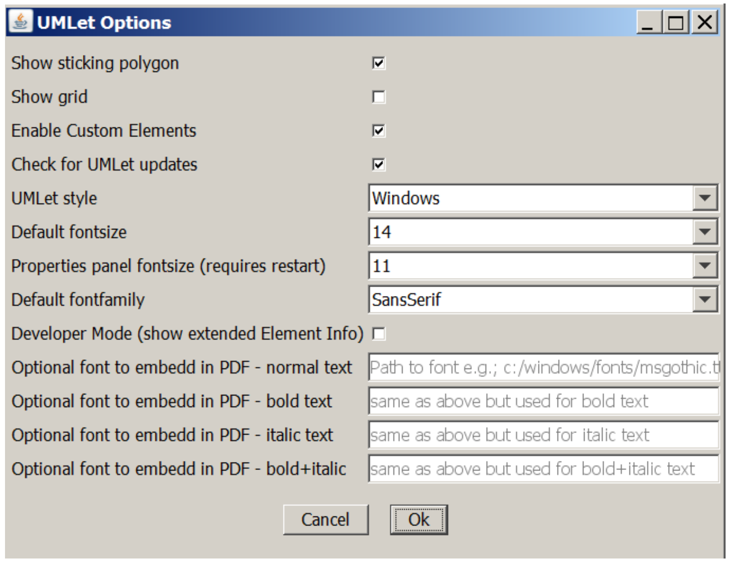
- One abstract factory class for an interface
- A set of concrete factories
  - each factories makes a family of objects



UMLet Options

Show sticking polygon	✓
Show grid	☐
Enable Custom Elements	✓
Check for UMLet updates	✓
UMLet style	CDE/Motif
Default fontsize	14
Properties panel fontsize (requires restart)	11
Default fontfamily	SansSerif
Developer Mode (show extended Element Info)	☐
Optional font to embedd in PDF - normal text	Path to font e.g.; c:/windows/fonts/msgothic.ttc,
Optional font to embedd in PDF - bold text	same as above but used for bold text
Optional font to embedd in PDF - italic text	same as above but used for italic text
Optional font to embedd in PDF - bold+italic	same as above but used for bold+italic text

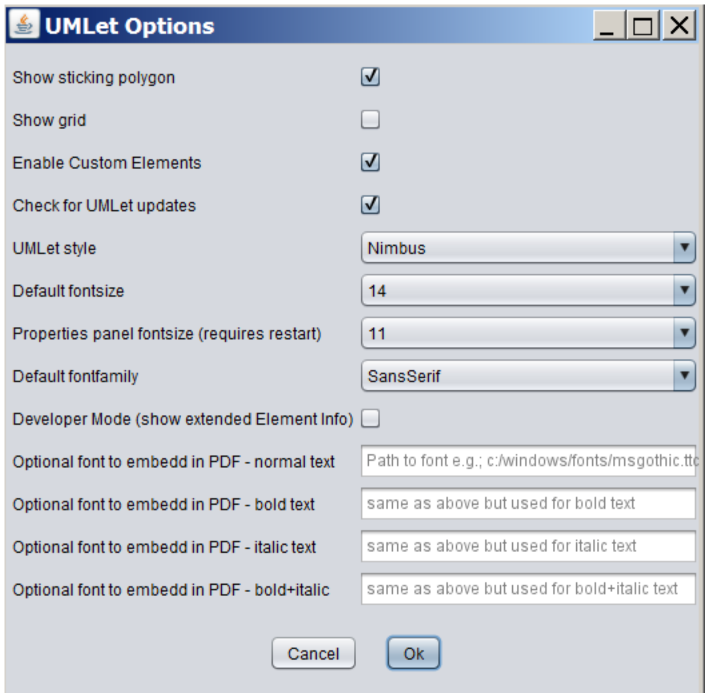
Cancel Ok



UMLet Options

Show sticking polygon	✓
Show grid	☐
Enable Custom Elements	✓
Check for UMLet updates	✓
UMLet style	Windows
Default fontsize	14
Properties panel fontsize (requires restart)	11
Default fontfamily	SansSerif
Developer Mode (show extended Element Info)	☐
Optional font to embedd in PDF - normal text	Path to font e.g.; c:/windows/fonts/msgothic.t
Optional font to embedd in PDF - bold text	same as above but used for bold text
Optional font to embedd in PDF - italic text	same as above but used for italic text
Optional font to embedd in PDF - bold+italic	same as above but used for bold+italic text

Cancel Ok

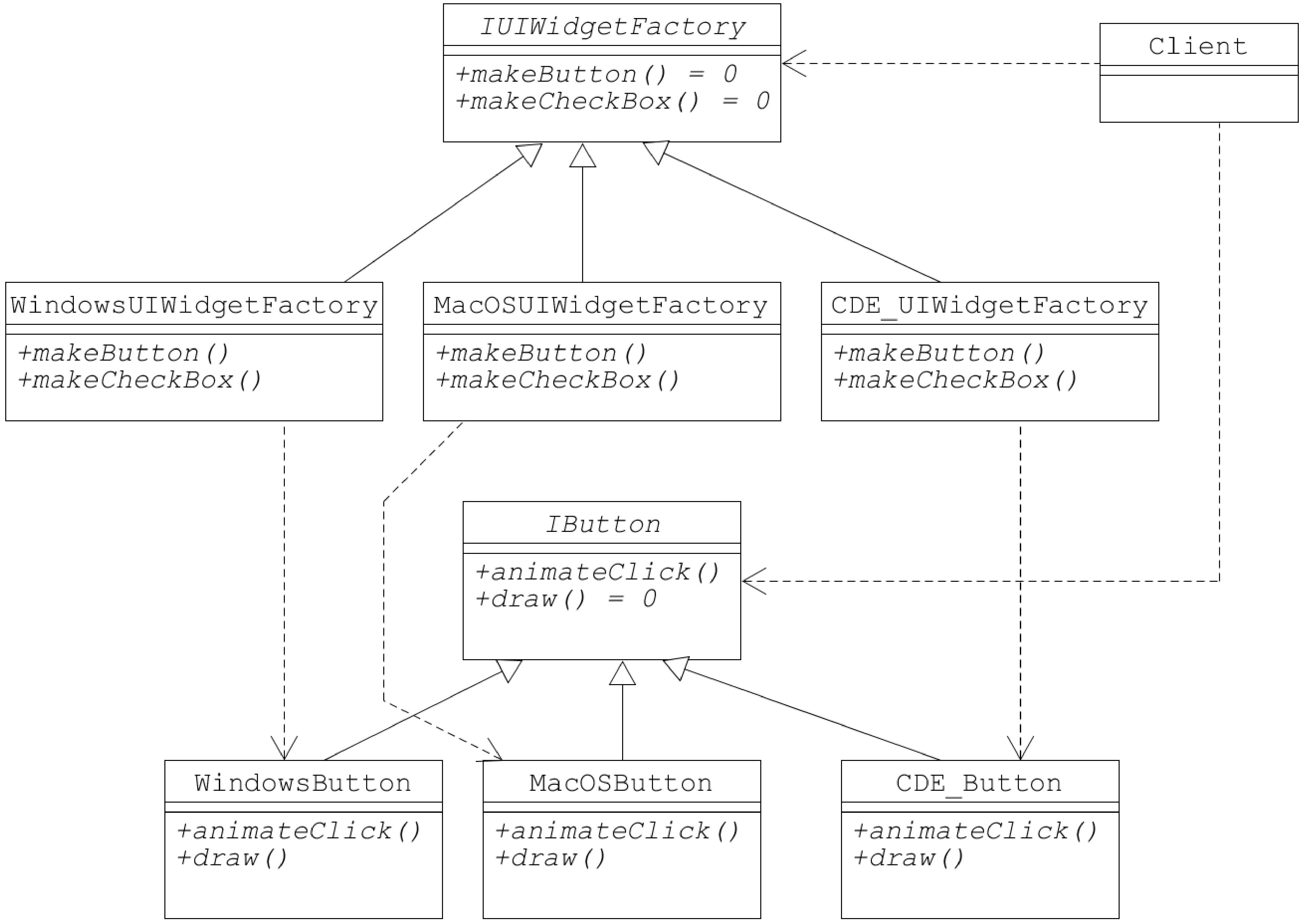


UMLet Options

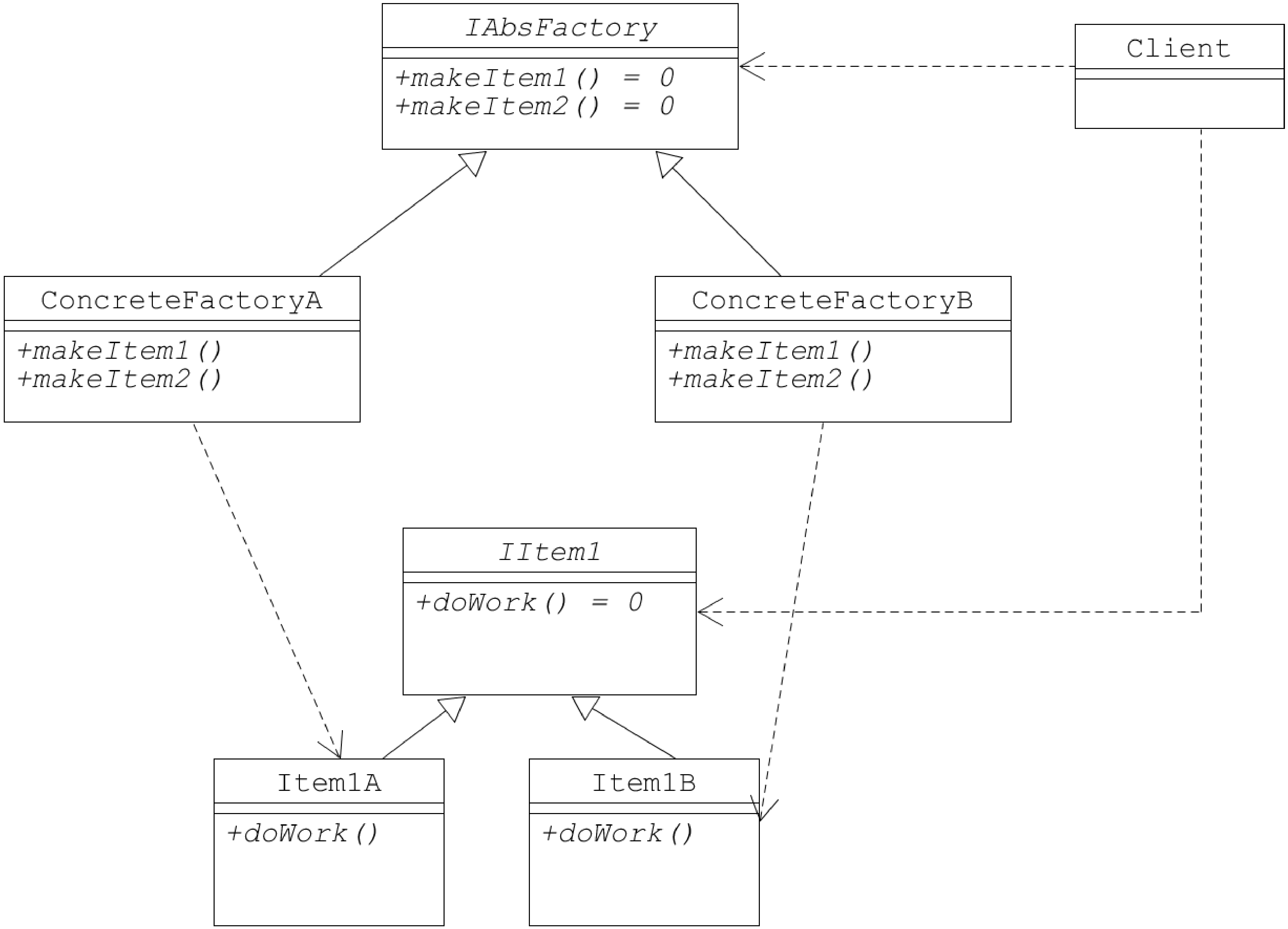
Show sticking polygon	✓
Show grid	☐
Enable Custom Elements	✓
Check for UMLet updates	✓
UMLet style	Nimbus
Default fontsize	14
Properties panel fontsize (requires restart)	11
Default fontfamily	SansSerif
Developer Mode (show extended Element Info)	☐
Optional font to embedd in PDF - normal text	Path to font e.g.; c:/windows/fonts/msgothic.tt
Optional font to embedd in PDF - bold text	same as above but used for bold text
Optional font to embedd in PDF - italic text	same as above but used for italic text
Optional font to embedd in PDF - bold+italic	same as above but used for bold+italic text

Cancel Ok

CDE/Motif Windows Nimbus/MacOS



adapted from Shalloway, p208


IAbsFactory
+makeItem1() = 0
+makeItem2() = 0
Client
ConcreteFactoryA
+makeItem1()
+makeItem2()
ConcreteFactoryB
+makeItem1()
+makeItem2()
IItem1
+doWork() = 0
Item1A
+doWork()
Item1B
+doWork()