Abstract Factory Pattern

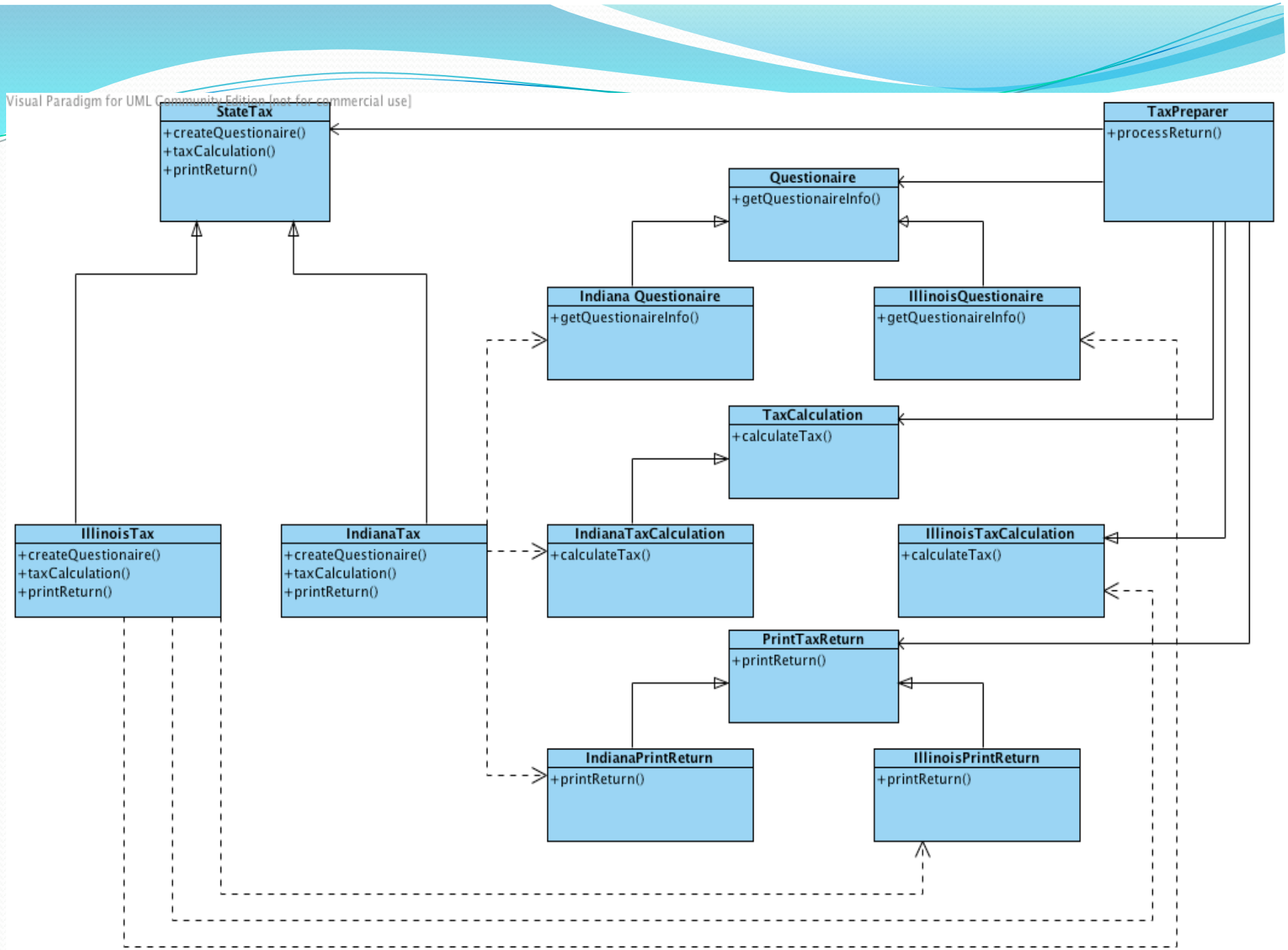
Tom Hoch

The Abstract Factory

- Provides an interface for creating families of related or dependent objects without specifying their concrete classes.
- Is one level of abstraction higher than the factory pattern
- Is sometimes called a kit

Reasons for

- When the systems needs to be independent of how its products are created and represented
- When the system needs to be configured with one of multiple families of products
- When a family of products need to be used together and this constraint needs to be enforced
- When you need to provide a library of products, expose their interfaces not the implementation.


Visual Paradigm for UML Community Edition [not for commercial use]
StateTax
+createQuestionaire()
+taxCalculation()
+printReturn()
TaxPreparer
+processReturn()
Questionaire
+getQuestionaireInfo()
Indiana Questionaire
+getQuestionaireInfo()
IllinoisQuestionaire
+getQuestionaireInfo()
TaxCalculation
+calculateTax()
IllinoisTax
+createQuestionaire()
+taxCalculation()
+printReturn()
IndianaTax
+createQuestionaire()
+taxCalculation()
+printReturn()
IndianaTaxCalculation
+calculateTax()
IllinoisTaxCalculation
+calculateTax()
PrintTaxReturn
+printReturn()
IndianaPrintReturn
+printReturn()
IllinoisPrintReturn
+printReturn()

Illinois Questionaire	
Name	
SS#	
Illinois Income	

Taxes due	
You owe Illinois	$100

Illinois Tax Return	
Name	J. Blago
Address	Pontiac, IL
Amount Due	$100

Java Code - Client

```
package state_Tax_Return;

public class TaxPreparer {
private Questionaire questionaire;
private TaxCalculation taxCalc;
private PrintTaxReturn printReturn;

public TaxPreparer(StateTax stateTax) {
questionaire = stateTax.createQuestionaire();
taxCalc = stateTax.taxCalculation();
printReturn = stateTax.printReturn();
}

public void processReturn() {
questionaire.getQuestionaireInfo();
taxCalc.calculateTax();
printReturn.printReturn();
}
```

```
public static void main(String[]
args) {
String state = "IL";
TaxPreparer taxPreparer = null;
StateTax stateTax = null;

if (state == "IN"){
stateTax = new IndianaTax();
} else if (state == "IL") {
stateTax = new IllinoisTax();
}
taxPreparer = new
TaxPreparer(stateTax);
taxPreparer.processReturn();

}

}
```

Java Code – Abstract Factory

```
package state_Tax_Return;

public abstract class StateTax {
public abstract Questionaire createQuestionaire();
public abstract TaxCalculation taxCalculation();
public abstract PrintTaxReturn printReturn();
}
```

Java Code – Concrete Factories Illinois

```
package state_Tax_Return;

public class IllinoisTax extends StateTax {
@Override
public Questionaire createQuestionaire() {
return new IllinoisQuestionaire();
}
@Override
public TaxCalculation taxCalculation() {
return new IllinoisTaxCalculation();
}
@Override
public PrintTaxReturn printReturn() {
return new IllinoisPrintReturn();
}

}
```

Java Code – Concrete Factories Indiana

```
package state_Tax_Return;

public class IndianaTax extends StateTax{
@Override
public Questionaire createQuestionaire() {
return new IndianaQuestionaire();
}
@Override
public TaxCalculation taxCalculation() {
return new IndianaTaxCalculation();
}
@Override
public PrintTaxReturn printReturn() {
return new IndianaPrintReturn();
}
}
```

Java Code - Abstract Products

```
package state_Tax_Return;
public abstract class Questionaire {
abstract void getQuestionaireInfo();
}
package state_Tax_Return;
public abstract class TaxCalculation {
abstract void calculateTax();
}
package state_Tax_Return;
public abstract class PrintTaxReturn {
abstract void printReturn();
}
```

Java Code – Concrete Products Illinois

```
package state_Tax_Return;
public class IllinoisQuestionaire extends Questionaire {
@Override
void getQuestionaireInfo() {
System.out.println("How much did you make in Illinois?");}
}
public class IllinoisTaxCalculation extends TaxCalculation {
@Override
void calculateTax() {
System.out.println("Tax Due is $100");}
}
public class IllinoisPrintReturn extends PrintTaxReturn {
@Override
void printReturn() {
System.out.println("Mail $100 to Illinois");}
}
```

Java Code – Concrete Products Indiana

```
package state_Tax_Return;
public class IndianaQuestionaire extends Questionaire {
@Override
void getQuestionaireInfo() {
System.out.println("How much did you make in Indiana?");}
}
public class IndianaTaxCalculation extends TaxCalculation {
@Override
void calculateTax() {
System.out.println("Tax Due is $200");}
}
public class IndianaPrintReturn extends PrintTaxReturn {
@Override
void printReturn() {
System.out.println("Mail $200 to Indiana");}
}
```

This will help the Designer by

- Isolating concrete classes
- Allowing the designer to change the product family easily
- Promoting strong look and feel