

**SE 2AA4, CS 2ME3 (Introduction to Software
Development)**

Winter 2017

25 Specification Via UML (Ch. 5 and others)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 8, 2017



Specification Via UML

- Administrative details
- Best specification technique?
- Interfaces in UML
 - ▶ Measurable interface
 - ▶ Measurer interface
- Generic classes in UML
- Use cases with UML
- Sequence diagrams in UML

Administrative Details

- Who has skimmed the homework problem?
- Some delay likely in grading A2
- A3 deadlines
 - ▶ Part 1 - Specification: due 11:59 pm [Mar 11](#)
 - ▶ Part 2 - Code: due 11:59 pm Mar 20
- A4
 - ▶ Your own design and specification
 - ▶ Due April 3 at 11:59 pm
- Sources of slides
 - ▶ DataSet example based on Cay Horstmann, Big Java, John Wiley & Sons, 2002
 - ▶ Other UML examples from Stevens, Using UML, 2006
 - ▶ Ghezzi et al 2003

Best?

- What is the best software development tool?
- What is the most important software design principle?
- What is the best specification technique?

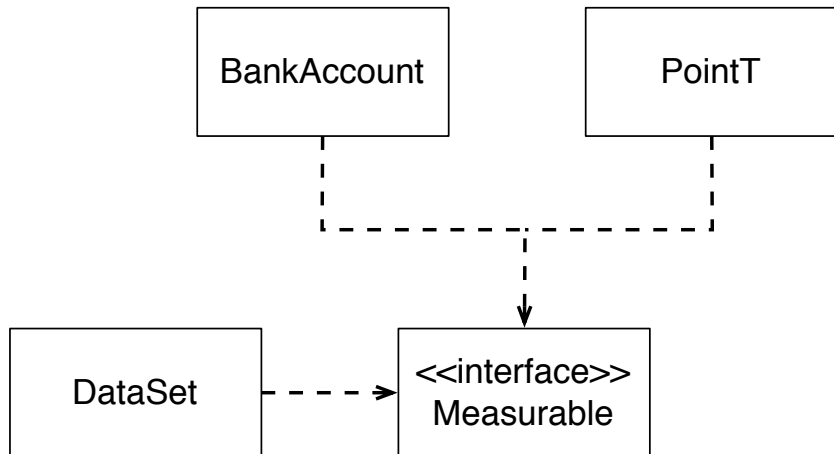
Best Continued

- What is the best programming language?
- What is the best engineering/scientific discipline?
- What is the best movie? video game?
- What is the best genre of music?
- What is the best food?

Deciding the Best Strategy For a Given Problem

- What is the approach at your company?
- Likely maintenance, so many decisions have likely been made.
- What tools/techniques/programming language etc. do you know?
- What can you afford in terms of cost/time?
- What tool is appropriate for the task at hand?
- What are the requirements?
 - ▶ Verifiability?
 - ▶ Maintainability?
 - ▶ Reusability?
 - ▶ etc.
- etc.

UML Diagram of Measurable Interface



- Realization arrow should have an outline triangle
- UML diagram can also show interface method names
- Realization arrow is like weak generalization (inheritance)

DataSet Without Interface I

```
public class DataSet
{
    private double sum;
    private double maximum;
    private int count;

    public DataSet()
    {
        sum = 0;
        count = 0;
        maximum = 0;
    }
    public void add(double x)
    {
```


DataSet Without Interface II

```
    sum = sum + x;  
    if (count == 0 || maximum < x)  
        maximum = x;  
    count++;  
}  
public double getAverage()  
{  
    if (count == 0) return 0;  
    else return sum/count;  
}  
public double getMaximum()  
{ return maximum;  
}  
}
```

PointT I

```
import static java.lang.Math.*;
public class PointT {
    private double xc;
    private double yc;
    public PointT(double x, double y) {
        xc = x;
        yc = y;}
    // ..
    public double distToOrigin() {
        return sqrt(pow(xc,2.0) +
            pow(yc,2.0));
    }
}
```

DataSet for Points I

```
public class DataSetPoint
{
    private double sum;
    private PointT maximum;
    private int count;
    public DataSetPoint()
    {
        sum = 0;
        count = 0;
        maximum = null;
    }
    public void add(PointT x)
    {
        sum = sum + x.distToOrigin();
    }
}
```

DataSet for Points II

```
        if (count == 0 ||
            maximum.distToOrigin() <
            x.distToOrigin()) maximum = x;
        count++;
    }
    public double getAverage()
    { if (count == 0) return 0;
      else return sum/count;
    }
    public PointT getMaximum()
    {
        return maximum;
    }
}
```

Bank Account Class I

```
public class BankAccount
{
    private double balance;

    public BankAccount()
    { balance = 0;}
    public void deposit(double amount)
    { balance = balance + amount;}
    public void withdraw(double amount)
    { balance = balance - amount;}
    public double getBalance()
    { return balance;}
}
```

DataSet for Bank Accounts I

```
public class DataSetBankAccount
{
    private double sum;
    private BankAccount maximum;
    private int count;

    public DataSetBankAccount()
    {
        sum = 0;
        count = 0;
        maximum = null;
    }
    public void add(BankAccount x)
    {
```

DataSet for Bank Accounts II

```
        sum = sum + x.getBalance();  
        if (count == 0 ||  
            maximum.getBalance() <  
            x.getBalance()) maximum = x;  
        count++;  
    }  
    public double getAverage()  
    { if (count == 0) return 0;  
      else return sum/count;  
    }  
    public BankAccount getMaximum()  
    { return maximum;  
    }  
}
```

Measurable Interface I

```
public interface Measurable  
{  
    double getMeasure();  
}
```


Data Set with Measurable Interface I

```
public class DataSetInterface
{
    private double sum;
    private Measurable maximum;
    private int count;

    public DataSetInterface()
    {
        sum = 0;
        count = 0;
        maximum = null;
    }
    public void add(Measurable x)
    {
```

Data Set with Measurable Interface II

```
    sum = sum + x.getMeasure();  
    if (count == 0 ||  
        maximum.getMeasure() <  
        x.getMeasure()) maximum = x;  
    count++;  
}  
public double getAverage()  
{ if (count == 0) return 0;  
  else return sum/count;  
}  
public Measurable getMaximum()  
{ return maximum;  
}  
}
```

PointT with Measurable Interface I

```
import static java.lang.Math.*;
public class PointTInterface implements
    Measurable
{
    private double xc;
    private double yc;
    public PointTInterface(double x, double
        y) {
        xc = x;
        yc = y;
    }
    //..
    public double distToOrigin() {
```

PointT with Measurable Interface II

```
        return sqrt(pow(xc,2.0) +  
                    pow(yc,2.0));  
    }  
    public double getMeasure(){  
        return distToOrigin();  
    }  
}
```

Bank Account with Measurable Interface I

```
public class BankAccountInterface
    implements Measurable
{
    private double balance;
    public BankAccountInterface()
    {
        balance = 0;
    }
    //..
    public double getBalance()
    {
        return balance;
    }
    public double getMeasure()
    {
        return balance;
    }
}
```

Using DataSet I

```
public class DataSetTest
{
    public static void main(String[] args)
    {
        DataSetInterface bankData = new
            DataSetInterface();
        bankData.add(new
            BankAccountInterface());
        BankAccountInterface b = new
            BankAccountInterface();
        b.deposit(134.56);
        bankData.add(b);
        System.out.println("Average balance =
            " + bankData.getAverage());
        Measurable max =
            bankData.getMaximum();
    }
}
```

Using DataSet II

```
System.out.println(" Highest balance =  
    " + max.getMeasure());  
DataSetInterface pointData = new  
    DataSetInterface();  
pointData.add(new  
    PointTInterface(1.0, 1.0));  
pointData.add(new  
    PointTInterface(2.0, 2.0));  
pointData.add(new  
    PointTInterface(3.0, 3.0));  
System.out.println(" Average distance  
    to origin = " +  
    pointData.getAverage());  
max = pointData.getMaximum();
```

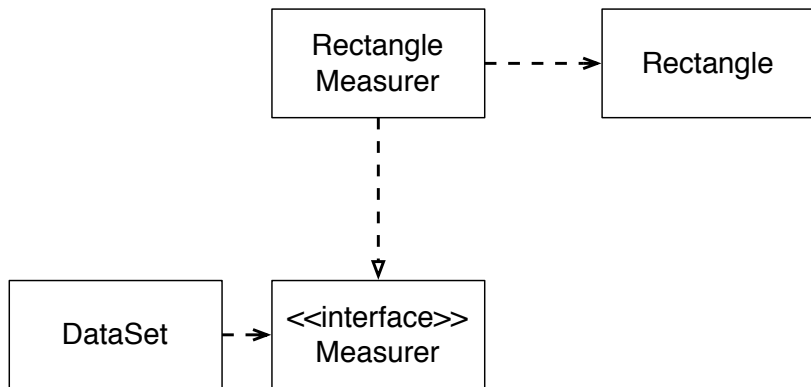
Using DataSet III

```
        System.out.println(" Greatest distance  
            to origin = " + max.getMeasure());  
    }  
}
```


Interface Strategy

- There are limitations to the Measurable interface
 - ▶ You can only add a Measurable interface to classes that you control
 - ▶ You can measure an object in only one way
- Move responsibility for measuring outside of objects themselves
- Have another object carry out the comparison
- Introduce a Measurer interface

UML Diagram of Measurer Interface



- Rectangle is part of Class java.awt
- You cannot change it

Measurer Interface I

```
public interface Measurer  
{  
    double measure(Object anObject);  
}
```

Data Set with New Strategy I

```
public class DataSetStrategy
{
    private double sum;
    private Object maximum;
    private int count;
    private Measurer measurer;
    public DataSetStrategy(Measurer
        aMeasurer)
    {
        sum = 0;
        count = 0;
        maximum = null;
        measurer = aMeasurer;
    }
    public void add(Object x)
    {
```

Data Set with New Strategy II

```
    sum = sum + measurer.measure(x);  
    if (count == 0 ||  
        measurer.measure(maximum) <  
        measurer.measure(x)) maximum = x;  
    count++;  
}  
public double getAverage()  
{ if (count == 0) return 0;  
  else return sum/count;  
}  
public Object getMaximum()  
{ return maximum;  
}  
}
```

Rectangle Measurer I

```
import java.awt.Rectangle;  
class RectangleMeasurer implements Measurer  
{  
    public double measure(Object anObject)  
    {  
        Rectangle aRectangle = (Rectangle)  
            anObject;  
        double area = aRectangle.getWidth() *  
            aRectangle.getHeight();  
        return area;  
    }  
}
```

Using Rectangle Measurer I

```
import java.awt.Rectangle;  
public class DataSetStrategyTest  
{  
    public static void main(String[] args)  
    {  
        class RectangleMeasurer implements  
            Measurer  
        {  
            public double measure(Object  
                anObject)  
            {  
                Rectangle aRectangle =  
                    (Rectangle) anObject;
```

Using Rectangle Measurer II

```
        double area =  
            aRectangle.getWidth() *  
            aRectangle.getHeight();  
        return area;  
    }  
}  
Measurer m = new RectangleMeasurer();  
DataSetStrategy data = new  
    DataSetStrategy(m);  
data.add(new Rectangle(5, 10, 20,  
    30));  
data.add(new Rectangle(10, 20, 30,  
    40));  
System.out.println(" Average area = "  
    + data.getAverage());
```


Using Rectangle Measurer III

```
        Rectangle max = (Rectangle)
            data.getMaximum();
        System.out.println("Maximum area = "
            + m.measure(max));
    }
}
```

Comparable Versus Comparator

- Comparable similar UML diagram to Measurable
- Comparator similar UML diagram to Measurer

```
public interface Comparable<T>
{
    int compareTo(T obj);
}
```

```
public interface Comparator<T>
{
    public int compare(T obj1 , T obj2);
}
```

Interface Versus Abstract

- What is the difference between an interface and an abstract class?

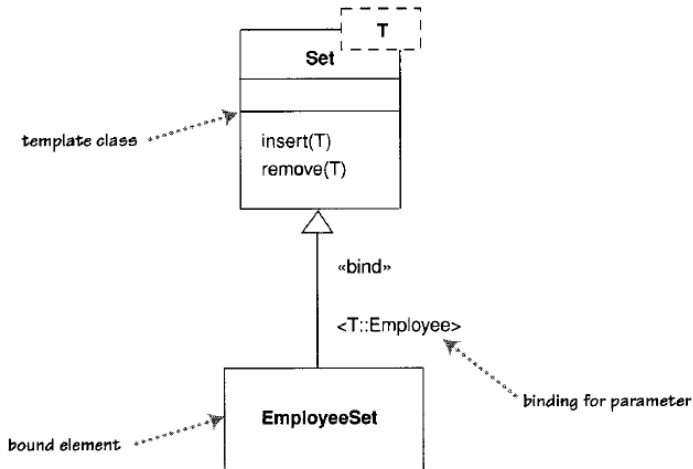
- Interface

- ▶ Methods are implicitly abstract and public
- ▶ Methods can have default implementation (JDK 8)
- ▶ Cannot have constructors
- ▶ Variables are final
- ▶ Can only extend interfaces
- ▶ Classes can extend multiple instances
- ▶ Appropriate for unrelated classes

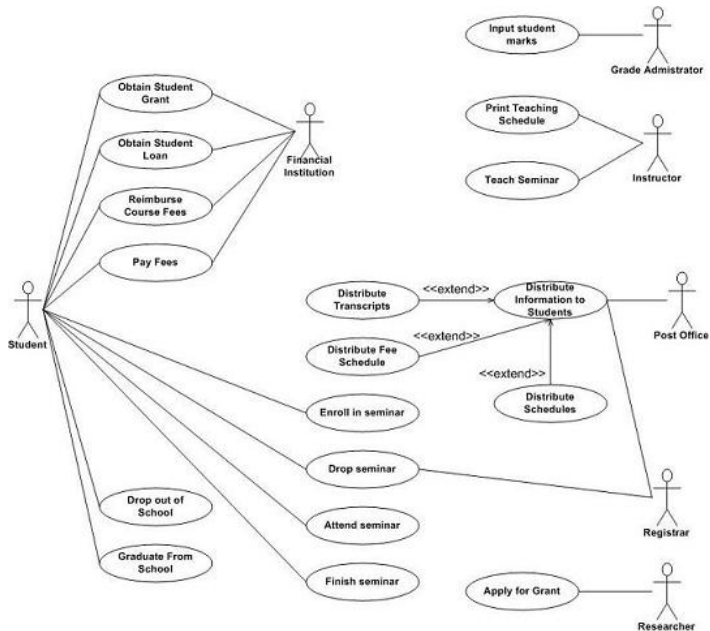
- Abstract class

- ▶ At least one method is declared as abstract
- ▶ Some methods can implement a default behaviour
- ▶ Cannot instantiate them, but can have constructors
- ▶ Variables are not necessarily final
- ▶ Can extend other class
- ▶ Can implement multiple interfaces
- ▶ Classes can extend only one abstract class
- ▶ Sharing code between closely related classes

UML Diagram for Generic Classes



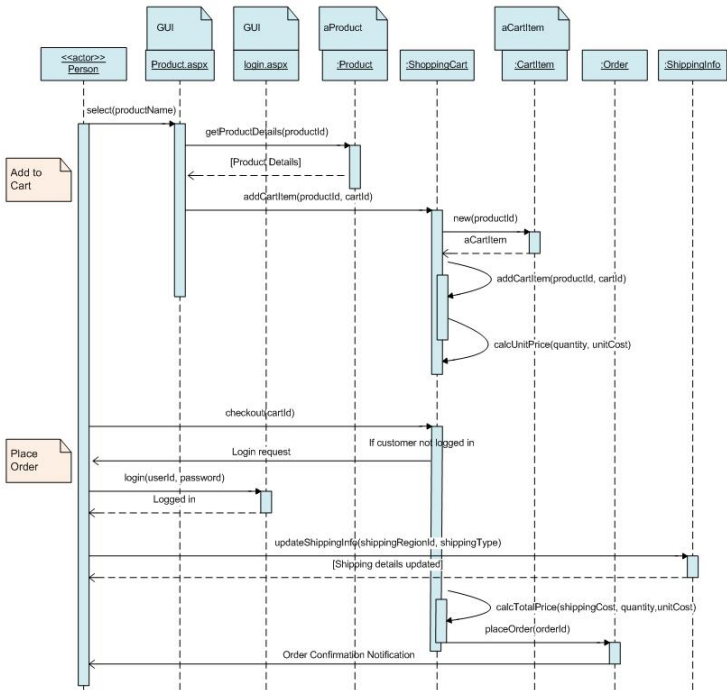
UML Class Diagram Template



UML 2 Use Case Diagrams: An Agile Introduction

Use Cases

- Often used for capturing requirements
- From user's (actor's) viewpoint
 - ▶ Person
 - ▶ Other system
 - ▶ Hardware
 - ▶ etc. (anything external)
- Each circle is a use case
- Lines represent possible interactions
- An actor represents a role, individuals can take on different roles



Sequence Diagram Question

- Is a sequence diagram an operational or a descriptive specification?
- If objects exchange a message, should there be an association between their classes?

Sequence Diagrams

- Represents a specific use case scenario
- How objects interact by exchanging messages
- Time progresses in the vertical direction
- The vertically oriented boxes show the object's lifeline