Interface & Polymorphism

class DataSet

```
/**
* Computes information about a set of
* data values.
*/
public class DataSet  {

  /**
  * Constructs an empty data set.
  */
  public DataSet() {
    sum = 0;
    count = 0;
    maximum = 0;
  }

  /**
  * Adds a data value to the data set.
  * @param x a data value
  */
  public void add(double x) {
    sum = sum + x;
    if (count == 0 || maximum < x)
      maximum = x;
    count++;
  }

  /**
  * Gets the average of the added data.
  * @return the average or 0 if no data
  *   has been added
  */
  public double getAverage() {
    if (count == 0) return 0;
    else return sum / count;
  }

  /**
  * Gets the largest of the added data.
  * @return the maximum or 0 if no data
  *    has been added
  */
  public double getMaximum() {
    return maximum;
  }

  private double sum;
  private double maximum;
  private int count;
}
```

DataSet has BankAccount

```
public class DataSet { // Modified for BankAccount objects
...
   public void add(BankAccount x) {
      sum = sum + x.getBalance();
      if (count == 0 || maximum.getBalance() < x.getBalance())
         maximum = x;
      count ++;
   }
   public BankAccount getMaximum() {
      return maximum;
   }
   private double sum;
   private BankAccount maximum;
   private int count;
}
```

DataSet has Coin

```
public class DataSet { // Modified for Coin objects
. . .
  public void add(Coin x) {
    sum = sum + x.getValue();
    if (count == 0 || maximum.getValue() < x.getValue())
      maximum = x;
    count++;
  }

  public Coin getMaximum() {
    return maximum;
  }

  private double sum;
  private Coin maximum;
  private int count;
}
```

การวัดแตกต่างกัน

สร้าง getMeasure() ที่เป็น common

```
sum = sum + x.getMeasure();
if (count == 0 || maximum.getMeasure() < x.getMeasure())
  maximum = x;
count++;
```

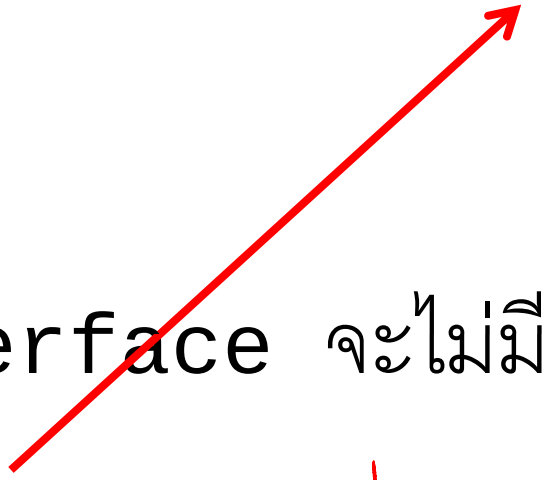
- แล้ว x จะเป็นประเภทไหน?
- ประเภทไหนก็ได้ที่มีเมท็อด getMeasure

interface เป็น type ที่ประกาศ เซตของเมท็อด เปรียบเหมือน contract

ทางเลือก: Interface Measurable

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

เมท็อดใน `interface` จะไม่มีการ `implement`



abstract method

Interface เป็น type

- เมท็อดทั้งหมดเป็น *abstract*
- เมท็อดทั้งหมดเป็น *public* โดยอัตโนมัติ
- ไม่มี *field* มีแต่ constant เช่น
  - *int NORTH = 1;*
  - *int EAST = 2;*
  - *ไม่ต้องเติม public static final เพราะจะเป็นโดยอัตโนมัติอยู่แล้ว*

DataSet ใหม่

```
public class DataSet { // Modified for Measurable objects
. . .
   public void add(Measurable x) {
      sum = sum + x.getMeasure();
      if (count == 0 || maximum.getMeasure() < x.getMeasure())
         maximum = x;
      count++;
   }

   public Measurable getMaximum() {
      return maximum;
   }

   private double sum;
   private Measurable maximum;
   private int count;
}
```

การใช้ interface

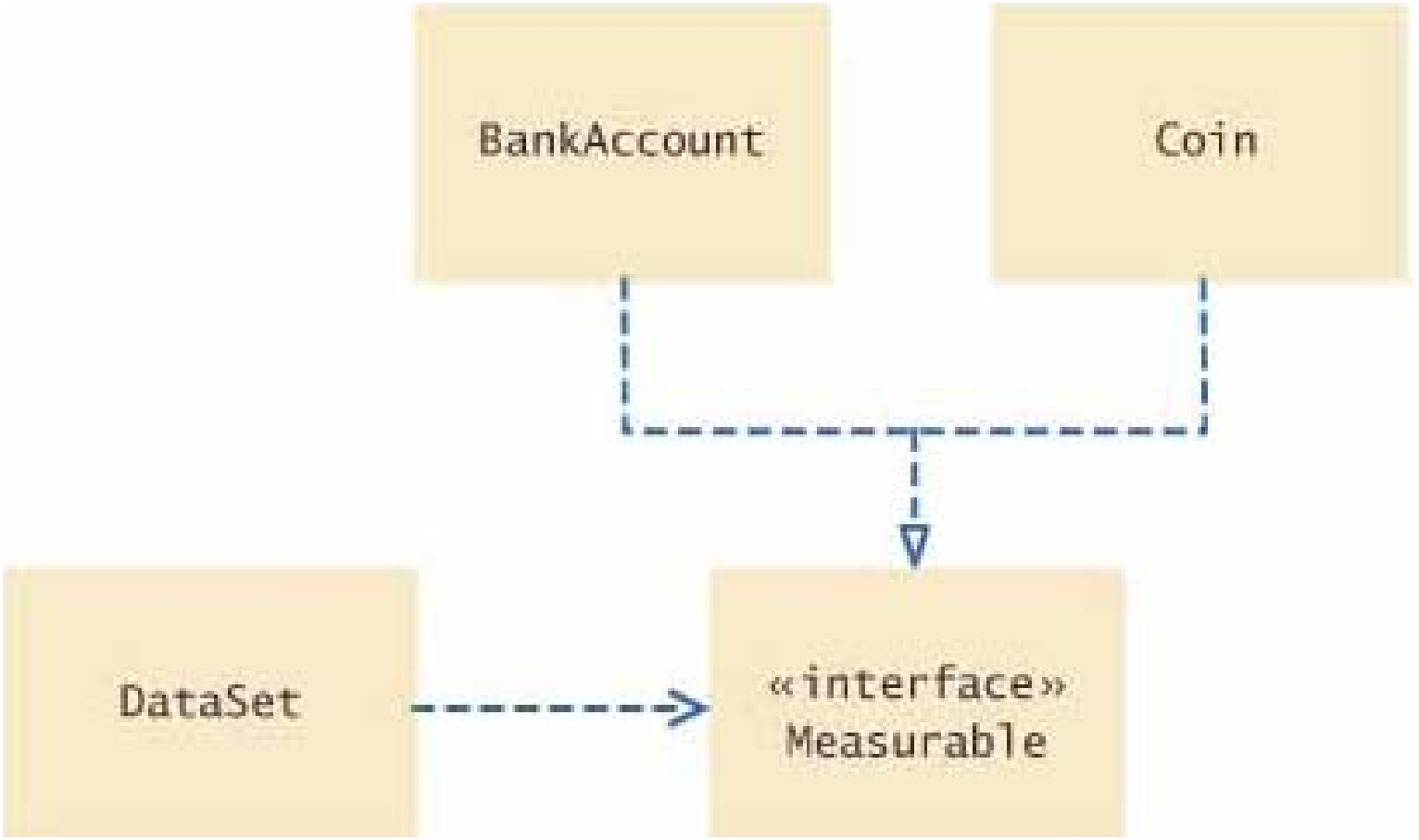
- สร้างคลาสใหม่ ที่ implements interface
- implements ได้หลาย interface แต่ extends ได้ 1 คลาส
- ต้อง implement (เขียนให้สมบูรณ์) ทุกเมท็อดที่กำหนดใน interface

```
class ClassName implements Measurable {
   public double getMeasure() {
      // implementation
   }

   // Additional methods and fields
}
```

BankAccount & Coin

```
public class BankAccount implements Measurable {
  public double getMeasure() {
    return balance;
  }
  . . .
}

public class Coin implements Measurable {
  public double getMeasure() {
    return value;
  }
  . . .
}
```

Relation



Class type ←→ Interface type

```
BankAccount account = new BankAccount(10000);
Measurable x = account // OK

Coin salung = new Coin(0.25, “salung”);
x = salung;   // OK
```

- You can convert from a class type to an interface type, provided the class implements the interface.

```
x = new Point(2.0, 2.0); // Error
x = new Measurable() // Error
```

Polymorphism

```
BankAccount account = new BankAccount(10000);
Measurable x = account // OK

Coin salung = new Coin(0.25, “salung”);
x = salung;   // OK
```

- เวลาเรียก `x.getMeasure()` จะเลือก `getMeasure()` ของ **`BankAccount`** หรือ **`Coin`**

- ขึ้นอยู่กับตอน run-time ว่า x อ้างอิงไปที่อ็อบเจ็คอะไร
- เรียกว่าเป็น polymorphism

Polymorphism

- เป็น late binding

- การ overload จะเป็น early binding เพราะสามารถรู้ได้ว่าจะเรียกเมท็อดไหนตอน compile-time

What is an Inner Class?

- Class declared within another class
- Accessing the members of the inner class:
  - Need to instatiate an object instance of an inner class first
  - Example:

```
innerObj.innerMember = 5;
//innerObj is an instance of the inner class
//innerMember is a member of the inner class
```

การเข้าถึงสมาชิกของ Outer class ภายใน Inner class

- Methods of the inner class can directly access members of the outer class
  - Example:

```
1. class Out {
2.   int OutData;
3.   class In {
4.     void inMeth() {
5.       OutData = 10;
6.     }
7.   }
8. }
```

Inner Class

```
class OuterClass {
  int data = 5;
  class InnerClass {
    int data2 = 10;
    void method() {
      System.out.println(data);
      System.out.println(data2);
    }
  }

//continued...
```

Inner Class

```
  public static void main(String args[]) {
    OuterClass oc = new OuterClass();
    InnerClass ic = oc.new InnerClass();
    System.out.println(oc.data);
    System.out.println(ic.data2);
    ic.method();
  }
}
```

ใช้เมื่อไม่ต้องการให้คลาสนี้มองเห็นจากที่อื่น แต่ยังมีการใช้ในคลาสนี้

Anonymous Inner Class

inner class ที่ไม่มีชื่อ

```
Measurable x = new Measurable() {
  public double getMeasure() {
    // codes in getMeasure()
  }
};
```

หรือ

```
dataSet.add(new Measurable() {
  public double getMeasure() {
    return . . .
  }});
```

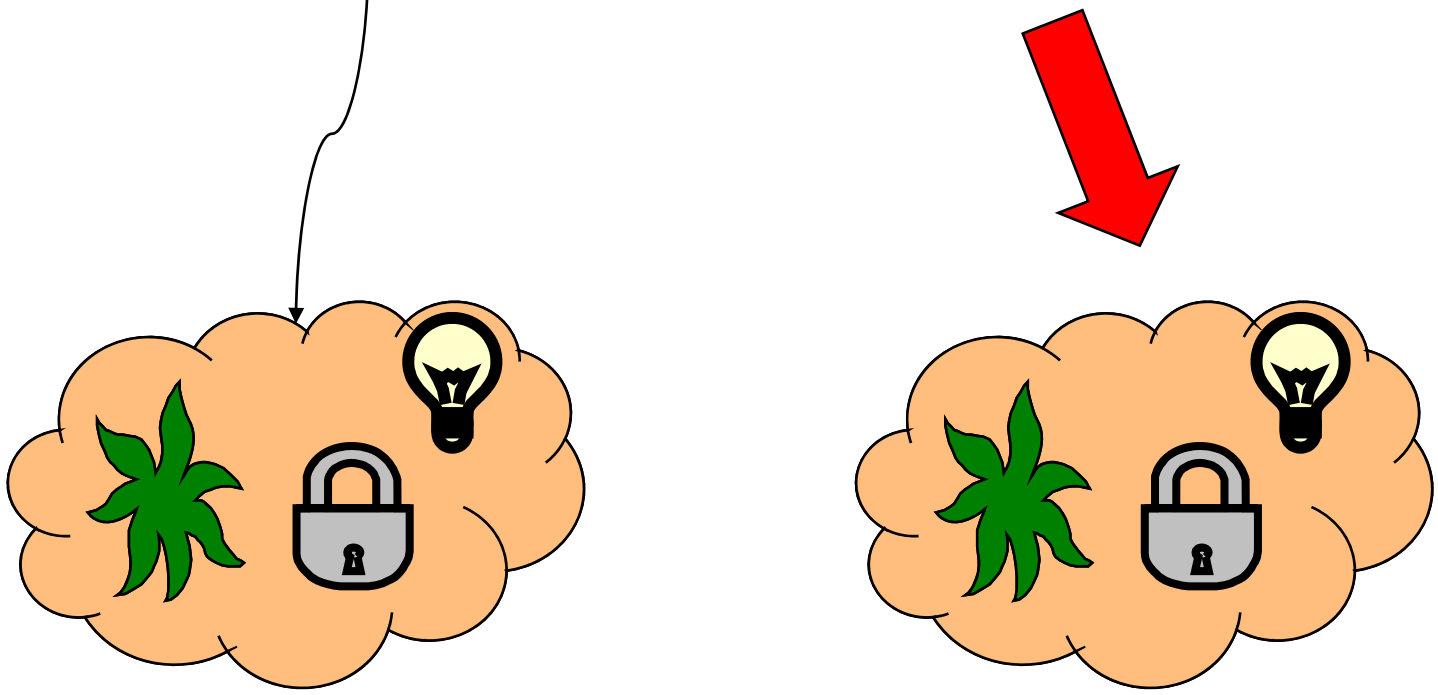
ใช้เยอะในตอนเขียน event listener ของ GUI

***เดี๋ยวได้เรียน

The clone() method

- To create an object from an existing object

```
aCloneableObject.clone()
```



Clone example

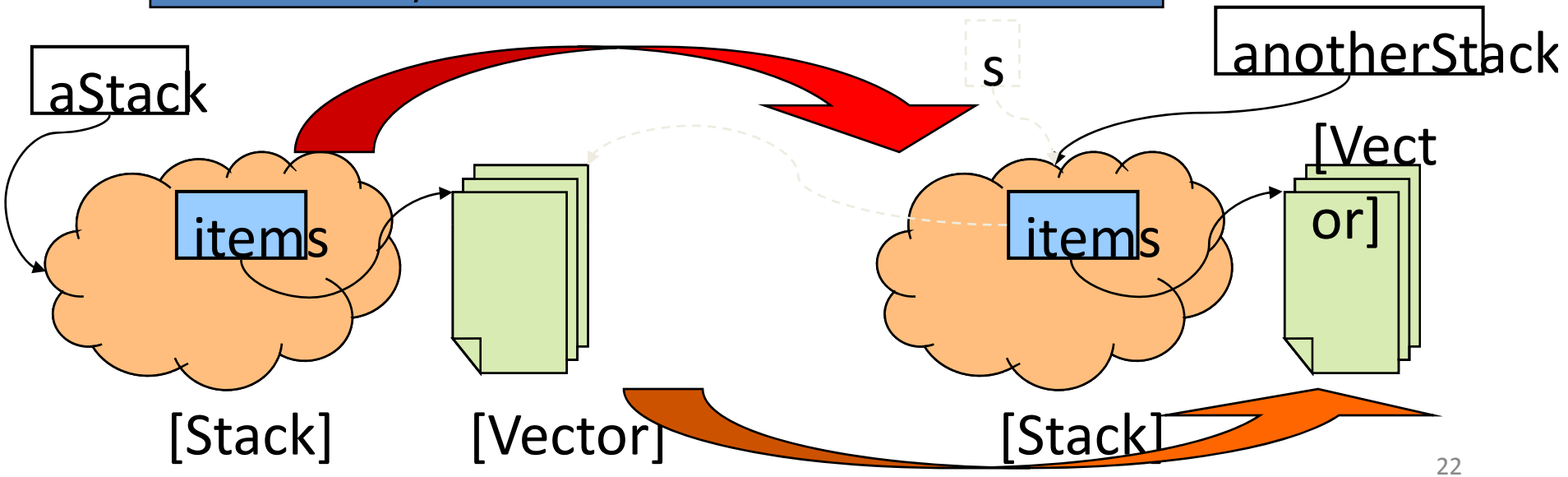
```
public class Stack implements Cloneable {
  private Vector items;

  // code for Stack's methods and constructor not shown

  protected Object clone() {
    try {
      Stack s = (Stack)super.clone();  // clone the stack
      s.items = (Vector)items.clone(); // clone the vector
      return s;                        // return the clone
    } catch (CloneNotSupportedException e) {
      // this shouldn't happen because Stack is Cloneable
      throw new InternalError();
    }
  }
}
```

To have **`clone()`**, one must implement **`Cloneable`**, otherwise **`CloneNotSupportedException`** will be thrown.

Clone example

```
Stack aStack = new Stack();
Stack anotherStack = aStack.clone();
```

```
Stack s = (Stack)super.clone();
s.items = (Vector)items.clone();
return s;
```



ปัญหาของ clone()

- tricky มาก ที่จะ implement ได้อย่างถูกต้อง
- ต้อง implements Cloneable แต่ Cloneable ไม่ได้กำหนดให้มีเมท็อด clone

- copy constructor เป็นทางเลือก
  - public SomeType(SomeType anObj)