Comp 401: Interfaces

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

Looping

```
public class Factorials {
  public static long loopingFactorial(int n) {
    long product = 1;
    while (n > 0) {
      product *= n;
      n -= 1;
    }
    return product;
  }
  …
}
```

Recursion

Base condition

```
public class Factorials {
  public static long recursingFactorial(int n) {
    if (n <= 1)
      return 1;
    return n*recursingFactorial(n-1);
  }
}
```

Recursive step

Functional programming

Looping Spreadsheet

```
public class ALoopingFactorialSpreadsheet {
  int number;
  long factorial;
  public int getNumber() {
    return number;
  }
  public void setNumber(int newVal) {
    number = newVal ;
    factorial = Factorials.loopingFactorial(number);
  }
  public long getFactorial() {
    return factorial;
  }
}
```

```
ALoopingFactorialSpreadsheet factorial1 =
    new ALoopingFactorialSpreadsheet ();
ALoopingFactorialSpreadsheet factorial2 =
    new ALoopingFactorialSpreadsheet ();
factorial1.setNumber(2);
factorial2.setNumber(7);
```

Recursing Spreadsheet

```
public class ARecursingFactorialSpreadsheet {
  int number;
  long factorial;
  public int getNumber() {
    return number;
  }
  public void setNumber(int newVal) {
    number = newVal ;
    factorial = Factorials.recursingFactorial(number);
  }
  public long getFactorial() {
    return factorial;
  }
}
```

Uses looping rather than recursive factorial

User does not know or care

The public methods (visible components) of the two classes remain the same

```
ARecursingFactorialSpreadsheet factorial1 =
    new ARecursingFactorialSpreadsheet ();
ARecursingFactorialSpreadsheet factorial2 =
    new ARecursingFactorialSpreadsheet ();
factorial1.setNumber(2);
factorial2.setNumber(7);
```

Method calling code remains the same

Interfaces: Capturing Common Visible Components

```
public interface FactorialSpreadsheet {
  public int getNumber();
  public void setNumber(int newVal) ;
  public long getFactorial();
}
```

Like a class, an interface must be in its own file, with a .java suffix

Looping Interface Implementation

```
public class ALoopingFactorialSpreadsheet implements FactorialSpreadsheet {
  int number;
  long factorial;
  public int getNumber() {
    return number;
  }
  public void setNumber(int newVal) {
    number = newVal ;
    factorial = Factorials.loopingFactorial(number);
  }
  public long getFactorial() {
    return factorial;
  }
}
```

Interface as type

```
FactorialSpreadsheet factorial1 =
    new ALoopingFactorialSpreadsheet ();
FactorialSpreadsheet factorial2 =
    new ALoopingFactorialSpreadsheet ();
factorial1.setNumber(2);
factorial2.setNumber(2);
```

Creating a Class instance

Another Implementation of same Interface

```
public class ARecursingFactorialSpreadsheet implements FactorialSpreadsheet {
  int number;
  long factorial;
  public int getNumber() {
    return number;
  }
  public void setNumber(int newVal) {
    number = newVal ;
    factorial = Factorials.recursingFactorial(number);
  }
  public long getFactorial() {
    return factorial;
  }
}
```

```
FactorialSpreadsheet factorial1 =
    new ALoopingFactorialSpreadsheet ();
FactorialSpreadsheet factorial2 =
    new ARecursiveFactorialSpreadsheet ();
factorial1.setNumber(2);
Factorial2.setNumber(7);
factorial1 = factorial2;
```

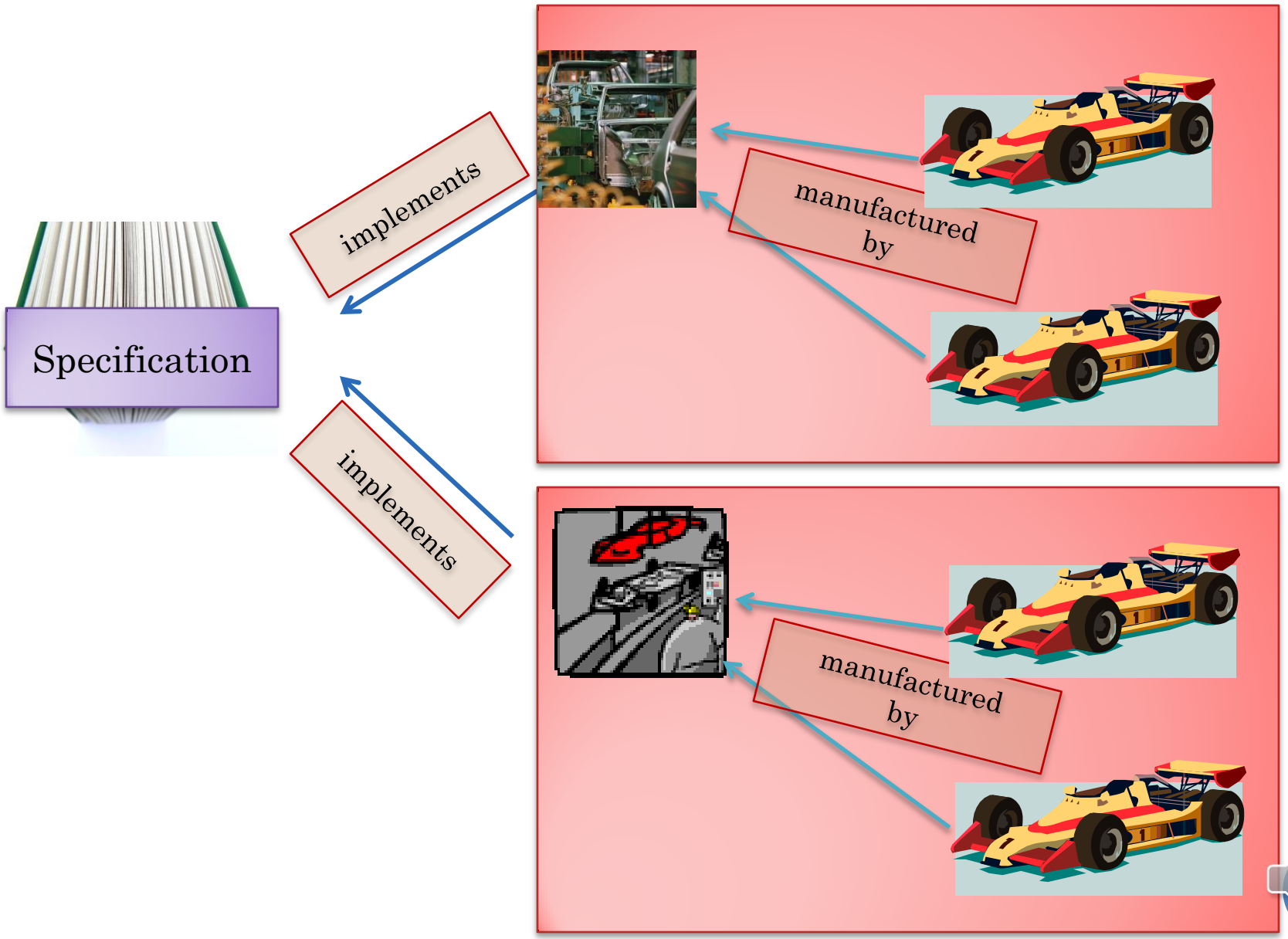
can be assigned an instance of any class that implements the interface

Makes it easier to switch implementations

Interface Implementation

```
public class ABMISpreadsheet implements BMISpreadsheet {
  double height;
  double weight;
  public ABMISpreadsheet(double theInitialHeight, double theInitialWeight) {
    setHeight(theInitialHeight);
    setWeight(theInitialWeight);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
  }
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight) {
    height = newHeight;
  }
  public double getBMI() {
    return weight/(height*height);
  }
}
```

Real-World Analogy for Interfaces



BMISpreadsheet Interface

```
public interface BMISpreadsheet {
  double getWeight();
  void setWeight(double newWeight);
  double getHeight();
  void setHeight(double newHeight);
  double getBMI();
}
```

Matching Headers

Contract

```
public class AnotherBMISpreadsheet implements BMISpreadsheet {
    double height, weight, bmi;
    public double getHeight() {
        return height;
    }
    public void setHeight(double newHeight) {
        height = newHeight;
        bmi = weight/(height*height);
    }
    …
```

Parameter names never matter to Java

BMISpreadsheet.java

```
public interface BMISpreadsheet {

    public double getWeight();
    public void setWeight(double newVal);

    public double getHeight();
    public void setHeight(double newVal);

    public double getBMI();

}
```

Alternative Implementation?

```
public class ABMISpreadsheet implements BMISpreadsheet {
  double height;
  double weight;
  public ABMISpreadsheet(double theInitialHeight, double theInitialWeight) {
    setHeight(theInitialHeight);
    setWeight(theInitialWeight);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
  }
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight) {
    height = newHeight;
  }
  public double getBMI() {
    return weight/(height*height);
  }
}
```

Another BMI Spreadsheet

```
public class AnotherBMISpreadsheet implements BMISpreadsheet{
  double height, weight, bmi;
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight)
    height = newHeight;
    bmi = weight/(height*height);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
    bmi = weight/(height*height);
  }
  public double getBMI() {
     return bmi;
  }
}
```

Independent value setters compute and store the new dependent properties returned by the property getters

Instead of dependent property getters computing the values

BMISpreadsheet Interface (Review)

```
public interface BMISpreadsheet {
  double getWeight();
  void setWeight(double newWeight);
  double getHeight();
  void setHeight(double newHeight);
  double getBMI();
}
```

BMISpreadsheet (Review)

```
public class ABMISpreadsheet implements BMISpreadsheet {
  double height;
  double weight;
  public ABMISpreadsheet(double theInitialHeight, double theInitialWeight) {
    setHeight(theInitialHeight);
    setWeight(theInitialWeight);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
  }
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight) {
    height = newHeight;
  }
  public double getBMI() {
    return weight/(height*height);
  }
}
```

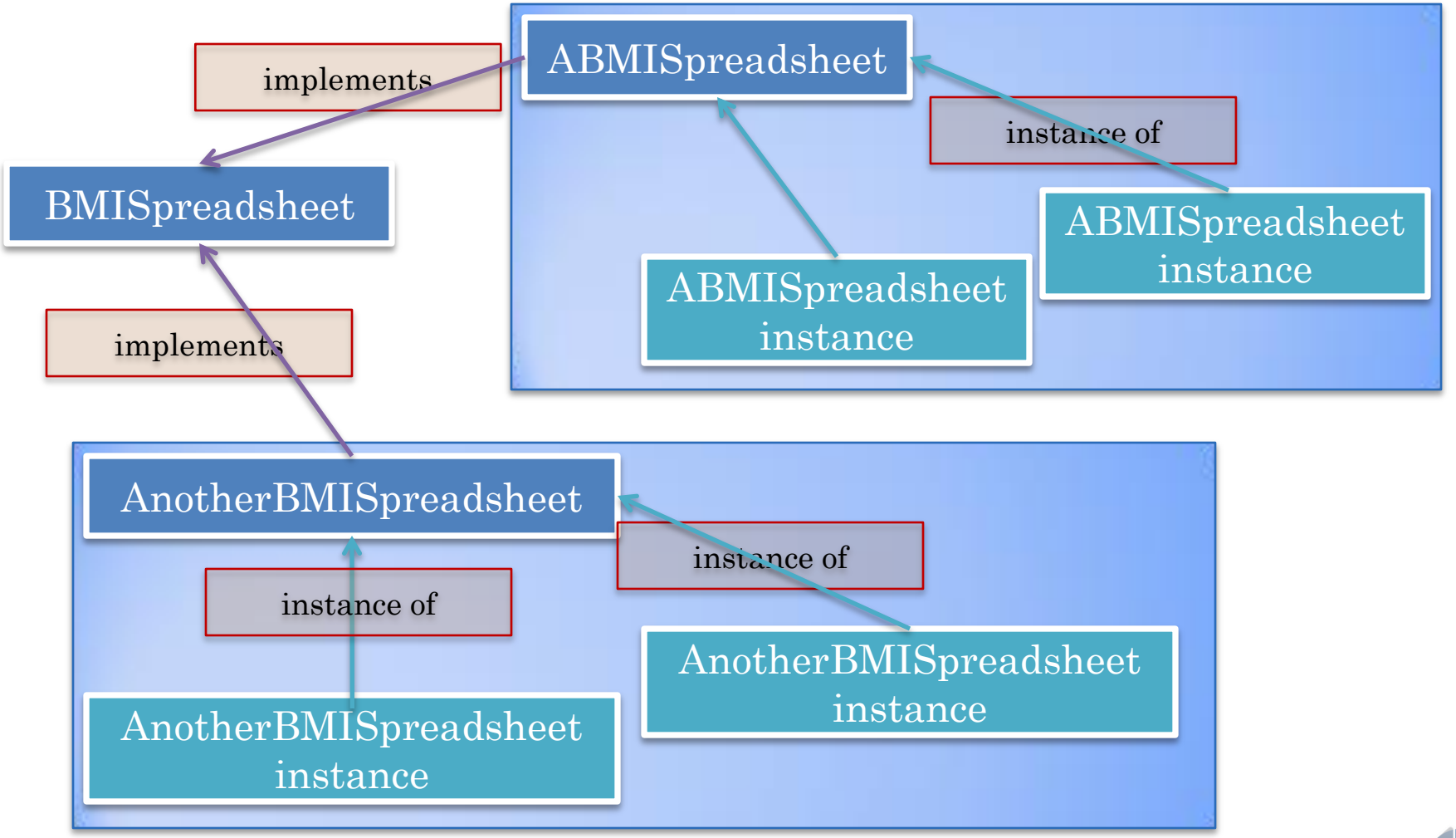
Another BMI Spreadsheet (Review)

```
public class AnotherBMISpreadsheet implements BMISpreadsheet{
  double height, weight, bmi;
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight)
    height = newHeight;
    bmi = weight/(height*height);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
    bmi = weight/(height*height);
  }
  public double getBMI() {
     return bmi;
  }
}
```

Independent value setters compute and store the new dependent properties returned by the property getters

Instead of dependent property getters computing the values

Interface


implements
ABMISpreadsheet
instance of
BMISpreadsheet
ABMISpreadsheet instance
ABMISpreadsheet instance
implements
AnotherBMISpreadsheet
instance of
instance of
AnotherBMISpreadsheet instance
AnotherBMISpreadsheet instance

Java Instanceof Boolean Operator

(**new** ABMISpreadsheet(1.77, 75)) **instanceof** ABMISpreadsheet	**true**
(**new** ABMISpreadsheet(1.77, 75)) **instanceof** BMISpreadsheet	**true**
(**new** ABMISpreadsheet(1.77, 75)) **instanceof** AnotherBMISpreadsheet	**false**

- true and false are values of Java type boolean
- Instanceof is Java keyword
- If class or interface of object o is T then o instanceof T returns true

Using Interfaces to Type

```
public class BMISpreadsheetUser {
    public static void main(String[] args) {
        BMISpreadsheet bmiSpreadsheet = new
ABMISpreadsheet(1.77, 75);
        System.out.println(bmiSpreadsheet.getBMI());
        bmiSpreadsheet = new AnotherBMISpreadsheet();
        bmiSpreadsheet.setHeight(1.77);
        bmiSpreadsheet.setWeight(75);
        System.out.println(bmiSpreadsheet.getBMI());
    }
}
```

Interface as type

Same variable assigned instances of two different classes

Type Checking

```
public class BMISpreadsheetUser {
    public static void main(String[] args) {
        BMISpreadsheet bmiSpreadsheet = new
ABMISpreadsheet(1.77, 75);
        System.out.println(bmiSpreadsheet.obtainBMI());
        bmiSpreadsheet = new AnotherBMISpreadsheet();
        bmiSpreadsheet.setHeight(1.77);
        bmiSpreadsheet.setWeight(75);
        System.out.println(bmiSpreadsheet.getBMI());
    }
}
```

Not defined in interface (or class)

Interface Methods Considered in Type Checking

```
public class BMISpreadsheetUser {
    public static void main(String[] args) {
        BMISpreadsheet bmiSpreadsheet = new
ABMISpreadsheet(1.77, 75);
        System.out.println(bmiSpreadsheet.getBMI());
        bmiSpreadsheet = new AnotherBMISpreadsheet();
        bmiSpre
        bmiSpre
        System.o
    }
}
```

BMISpreadsheet.java

```
public interface BMISpreadsheet {

    le getWeight();
    setWeight(double newVal);

    le getHeight();
    public void setHeight(double newVal);

    public double getBMI();

}
```

Not defined in interface so illegal, even though defined in class

Class-based Typing and Overloaded Methods

```
public void print (ABMISpreadsheet aBMISpreadsheet) {
  System.out.println ("Height:" + aBMISpreadsheet.getHeight());
  System.out.println ("Weight:" + aBMISpreadsheet.getWeight());
  System.out.println("BMI:" + aBMISpreadsheet.getBMI());
}
```

```
print (new AnotherBMISpreadsheet());
print (new ABMISpreadsheet());
```

```
public void print (AnotherBMISpreadsheet aBMISpreadsheet) {
  System.out.println ("Height:" + aBMISpreadsheet.getHeight());
  System.out.println ("Weight:" + aBMISpreadsheet.getWeight());
  System.out.println("BMI:" + aBMISpreadsheet.getBMI());
}
```

```
print (new AnotherBMISpreadsheet());
print (new ABMISpreadsheet());
```

Interface-Based Typing and Polymorphic Methods

```
public void print (BMISpreadsheet aBMISpreadsheet) {
  System.out.println ("Height:" + aBMISpreadsheet.getHeight());
  System.out.println ("Weight:" + aBMISpreadsheet.getWeight());
  System.out.println("BMI:" + aBMISpreadsheet.getBMI());
}
```

```
print (new AnotherBMISpreadsheet());
print (new ABMISpreadsheet());
```

Polymorphic method: A method having at least one parameter that can be assigned objects of different classes

Yet Another Spreadsheet Class

```
public class ABMISpreadsheetAndCalculator implements ??? {
    double height, weight, bmi;
    public double getHeight() {
        return height;
    }
    public void setHeight(double newHeight) {
        height = newHeight;
        bmi = calculateBMI(height, weight);
    }
    public double getWeight() {
        return weight;
    }
    public void setWeight(double newWeight) {
        weight = newWeight;
        bmi = calculateBMI(height, weight);
    }
    public double getBMI() {
        return bmi;
    }
    public double calculateBMI(double height, double weight) {
        return weight/(height*height);
}
```

Implementing Multiple Interfaces

```
public class ABMISpreadsheetAndCalculator implements BMISpreadsheet, BMICalculator{
    double height, weight, bmi;
    public double getHeight() {
        return height;
    }
    public void setHeight(double newHeight) {
        height = newHeight;
        bmi = calculateBMI(height, weight);
    }
    public double getWeight() {
        return weight;
    }
    public void setWeight(double newWeight) {
        weight = newWeight;
        bmi = calculateBMI(height, weight);
    }
    public double getBMI() {
        return bmi;
    }
    public double calculateBMI(double height, double weight) {
        return weight/(height*height);
}
```

BMICalculator Interface

```
public interface BMICalculator {
   public double calculateBMI(double height, double weight);
}
```

ABMICalculator With Interface

```
public class ABMICalculator implements BMICalculator {
  public double calculateBMI(double height, double weight) {
    return weight/(height*height);
  }
}
```

Java Instanceof Boolean Operator

`new ABMISpreadsheetAndCalculator() instanceof ABMISpreadsheetAndCalculator`	**true**
`new ABMISpreadsheetAndCalculator() instanceof BMISpreadsheet`	**true**
`new ABMISpreadsheetAndCalculator() instanceof BMICalculator`	**false**
`new ABMISpreadsheetAndCalculator() instanceof ABMICalculator`	**false**
`new ABMISpreadsheetAndCalculator() instanceof ABMISpreadsheet`	**false**

If class or interface of object o is T then o instanceof T returns true

Typing

```
 public static void main (String[] args) {
    BMICalculator bmiCalculator = new ABMISpreadsheetAndCalculator();
    BMISpreadsheet bmiSpreadsheet = new ABMISpreadsheetAndCalculator();
    double bmi = bmiCalculator.calculateBMI(1.77, 75);
    bmi = bmiSpreadsheet.getBMI();
//     bmi = bmiCalculator.getBMI();
//     bmi = bmiSpreadsheet.calculateBMI(1.77, 75);

    }
```

```
public static void main (String[] args) {
  BMICalculator[] bmiCalculators = {new ABMISpreadsheetAndCalculator(),
                                    new ABMICalculatorWithInterface()};
  }
```

Car Analogy

- A car is characterized by
  - Its make
  - License plate
  - Registration
- Licensing authority groups car by the registration or license plate

Cannot Instantiate Specification

- Cannot order a car from a specification
  - Must order from factory
  - A car defined by Corvette specification ordered from factory implementing the specification

- Cannot instantiate interface
  - Must instantiate class
  - BMISpreadsheet instance created by instantiating class implementing interface

Interface as a Syntactic Specification

```
public class ABMISpreadsheet implements BMISpreadsheet{
    double height;
    public double getHeight() {
        return height;
    }
    public void setHeight(double newHeight) {
        height = newHeight;
    }
    double weight;
    public double getWeight() {
        return weight;
    }
    public void setWeight(double newWeight) {
        weight = newWeight;
    }
    public double getBMI() {
        return weight/(height*height);
    }
}
```

Interface as a Syntactic Specification

```
public class ABMISpreadsheet implements BMISpreadsheet{
    double height;
    public double getHeight() {
        return height;
    }
    public void setHeight(double newHeight) {
        height = newHeight;
    }
    double weight;
    public double getWeight() {
        return weight;
    }
    public void setWeight(double newWeight) {
        weight = newWeight;
    }
    public double getBMI() {
        return 13450;
    }
}
```

Syntactic Contract

Bombay Market Index

Bottom Up: BMISpreadsheet

```
public class ABMISpreadsheet implements BMISpreadsheet {
  double height;
  double weight;
  public ABMISpreadsheet(double theInitialHeight, double theInitialWeight) {
    setHeight(theInitialHeight);
    setWeight(theInitialWeight);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
  }
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight) {
    height = newHeight;
  }
  public double getBMI() {
    return weight/(height*height);
  }
}
```

Bottom Up; Another

```
public class AnotherBMISpreadsheet implements BMISpreadsheet{
  double height, weight, bmi;
  public double getHeight() {
    return height;
  }
  public void setHeight(double newHeight)
    height = newHeight;
    bmi = weight/(height*height);
  }
  public double getWeight() {
    return weight;
  }
  public void setWeight(double newWeight) {
    weight = newWeight;
    bmi = weight/(height*height);
  }
  public double getBMI() {
     return bmi;
  }
}
```

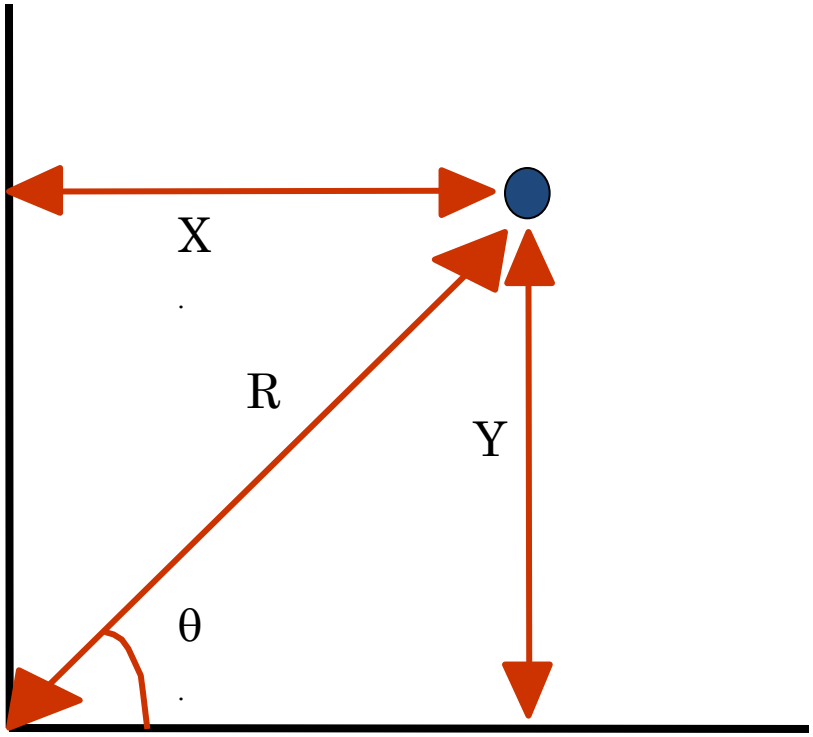
Independent value setters compute and store the new dependent properties returned by the property getters

Instead of dependent property getters computing the values

Bottom Up: BMISpreadsheet Interface

```
public interface BMISpreadsheet {
  double getWeight();
  void setWeight(double newWeight);
  double getHeight();
  void setHeight(double newHeight);
  double getBMI();
}
```

Top Down: Mathematical Point



Point Interface

```
public interface Point {
   public int getX();
   public int getY();
   public double getAngle();
   public double getRadius();
}
```

Read-only properties defining immutable object

Class: ACartesianPoint

```
public class ACartesianPoint implements Point {
  int x, y;
  public ACartesianPoint(int theX, int theY) {
    x = theX;
    y = theY;
  }
  public ACartesianPoint(double theRadius, double theAngle) {
     x =  (int) (theRadius*Math.cos(theAngle));
     y = (int) (theRadius*Math.sin(theAngle));
  }
  public int getX() { return x; }
  public int getY() { return y; }
  public double getAngle() { return Math.atan2(y, x); }
  public double getRadius() { return Math.sqrt(x*x + y*y); }
}
```

Class: APolarPoint

```
public class APolarPoint implements Point {
  double radius, angle;
  public APolarPoint(double theRadius, double theAngle) {
    radius = theRadius;
    angle = theAngle;
  }
  public APolarPoint(int theX, int theY) {
    radius = Math.sqrt(theX*theX + theY*theY);
    angle = Math.atan((double) theY/theX);
  }
  public int getX() { return (int) (radius*Math.cos(angle)); }
  public int getY() { return (int) (radius*Math.sin(angle)); }
  public double getAngle() { return angle; }
  public double getRadius() { return radius; }
}
```

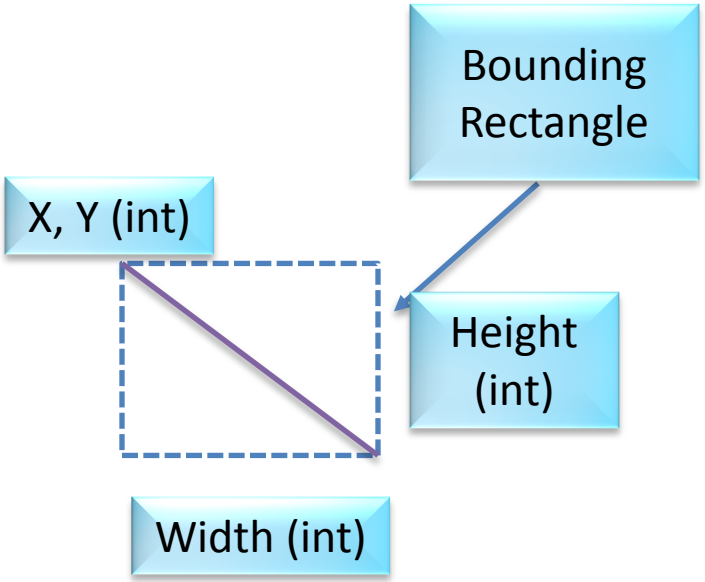
Representing Geometric Objects

- Geometric example to show multiple useful implementations of an interface
- Most geometric objects have multiple representations

What is a Representation

- Logical representation
  - Defined by its interface
  - Specifies properties
- Physical representation
  - Defined by its instance variables
- ACartesianPoint and APolarPoint have the same logical reperesentation but different physical representation

Line Logical Representation (Properties)



XY-Based Line/Oval/Rectangle Interface

```
public interface Line {
        public int getX();
        public void setX(int newX);
        public int getY();
        public void setY(int newY);
        public int getWidth();
        public void setWidth(int newVal);
        public int getHeight();
        public void setHeight(int newHeight);
}
```

Line Implementation

```
public class ALine implements Line {
  int x, y, width, height;
  public ALine (int initX, int initY, int initWidth, int initHeight) {
    x = initX;
    y = initY;
    width = initWidth;
   height = initHeight;
  }
  public int getX() {return x;}
  public void setX(int newX) {x = newX;}
  public int getY() { return y; }
  public void setY(int newY) {y = newY;}
  public int getWidth() {return width;}
  public void setWidth(int newVal) {width = newVal;}
  public int getHeight() {return height;}
  public void setHeight(int newHeight) {height = newHeight;}
}
```

Interfaces

- Define contracts between our users and implementers
- Optional – they may not be used
- Good style to use them
  - Each instance method must be in some interface (static methods cannot be in interfaces)
  - Each object variable must be type by an interface
- Can be derived bottom up or top down

Relating Interface and Class Names

Class Name:

<Qualifier><Interface>
- ABMISpreadsheet
- ASpaceEfficientBMISpreadsheet
- SpaceEfficientBMISpreadsheet

<Interface><Qualifier> Impl
- BMISpreadsheetImpl
- BMISpreadsheetSpaceEfficientImpl

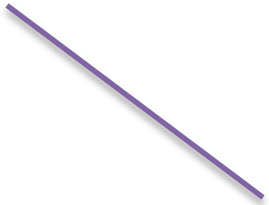
Interface Name:

<ClassName>Interface
- ABMISpreadsheetInterface

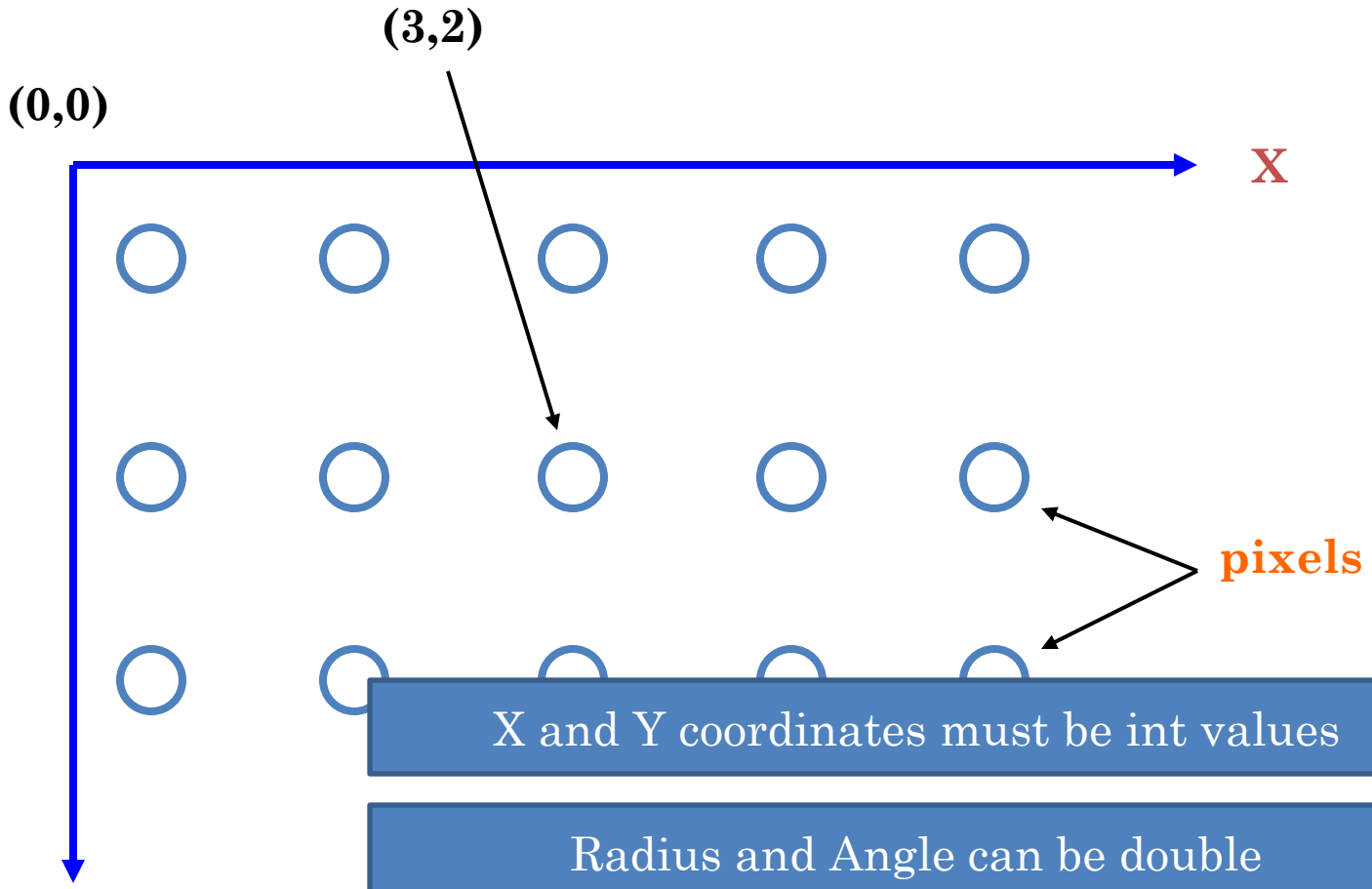
Assumes only one implementation of interface will be created

Extra

Line



Java Coordinate System



- X and Y coordinates must be int values
- Radius and Angle can be double
- Angle is a decimal value between 0 and 2π
- Each window has its own coordinate system