



COMP 110 FUNCTIONS

Instructor: Prasun Dewan

PREREQUISITE

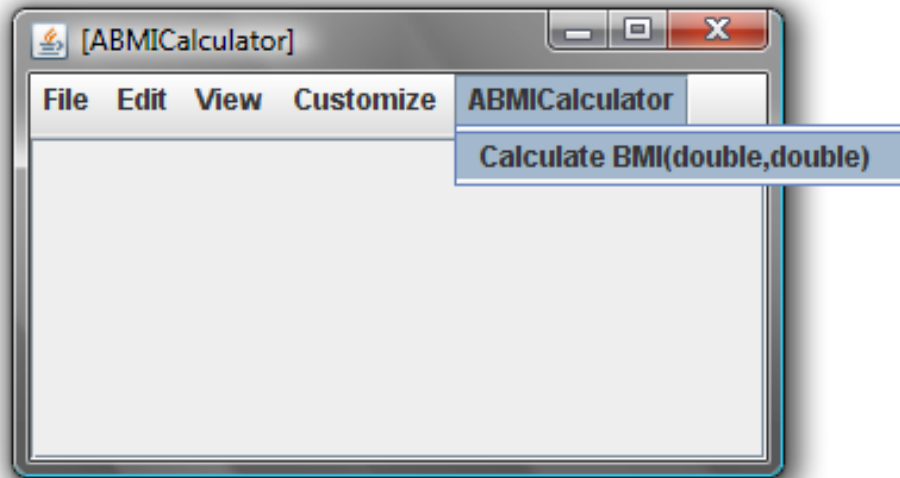
- Objects



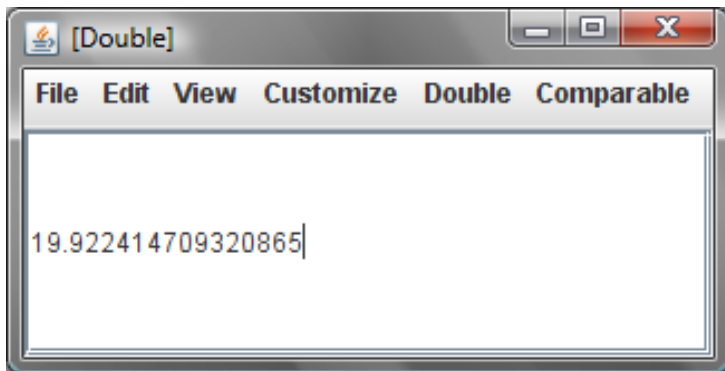
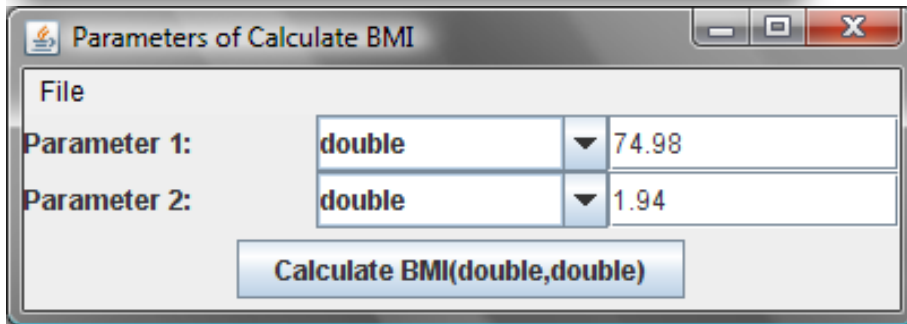
OUTLINE

- Programmatic instantiation of objects
- Functions calling other functions
- Algorithm and stepwise refinement
- Code Reuse
- Programming Style
- Variables, Named Constants, Literals
- Comments and Identifier Names

GENERAL PURPOSE BMI CALCULATOR

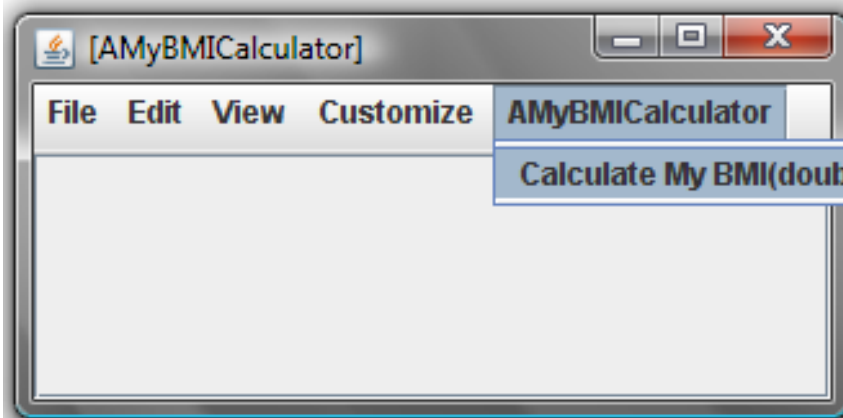


Does not assume height or weight

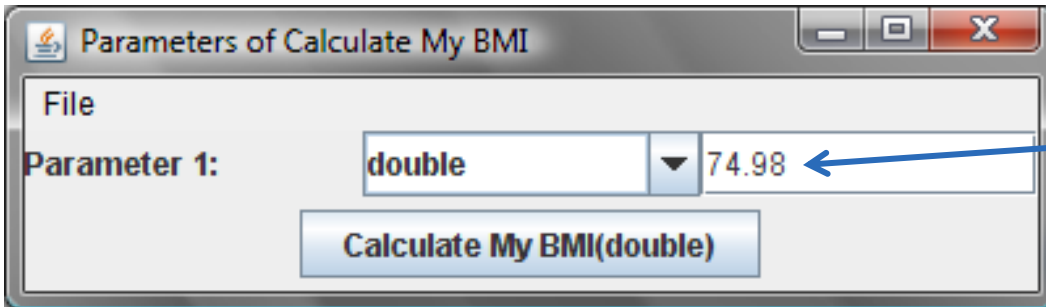


Specialized could know my height

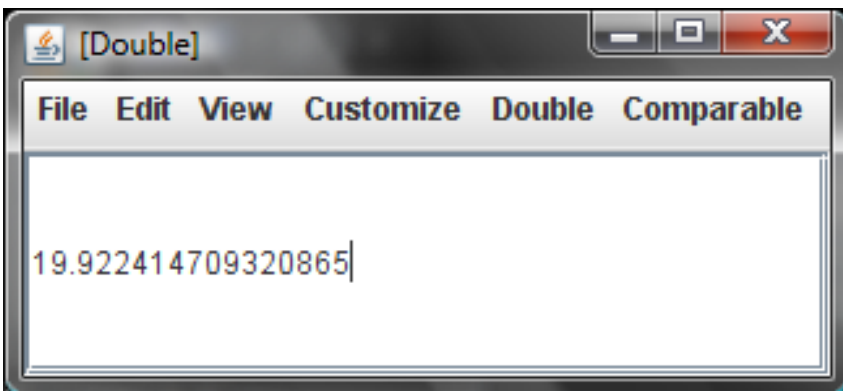
BMI CALCULATOR SPECIALIZED FOR AN INDIVIDUAL'S HEIGHT



Need to enter only weight



weight



Height is hard-coded into the program

A SOLUTION

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
  
    }  
}
```

A SOLUTION (EDIT)

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
  
    }  
}
```

A SOLUTION

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        return weight/ (1.94 * 1.94);  
    }  
}
```

Relationship with
ABMICalculator?

CUSTOMIZED VS. GENERAL PURPOSE

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        return weight/ (1.94 * 1.94);  
    }  
}
```

One vs. two
parameters

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

Basic formula is the
same (can cut and
paste)

SHOULD REUSE!

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        return weight/ (1.94 * 1.94);  
    }  
}
```

Should reuse code to avoid duplication of effort and errors such as: $(\text{weight})/1.94$

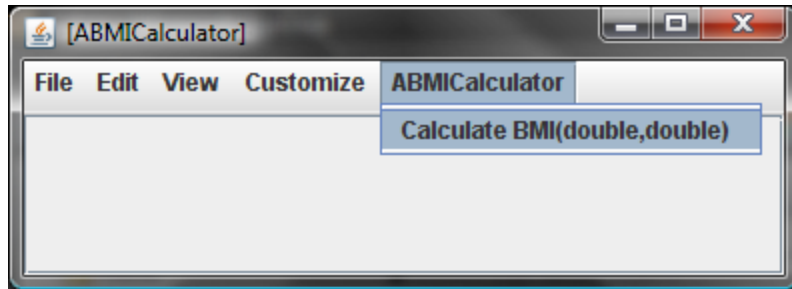
```
public class ABMICalculator {  
    public double calculateBMI (double weight,  
                                double height) {  
        return weight/ (height * height);  
    }  
}
```

Particularly important for complex code

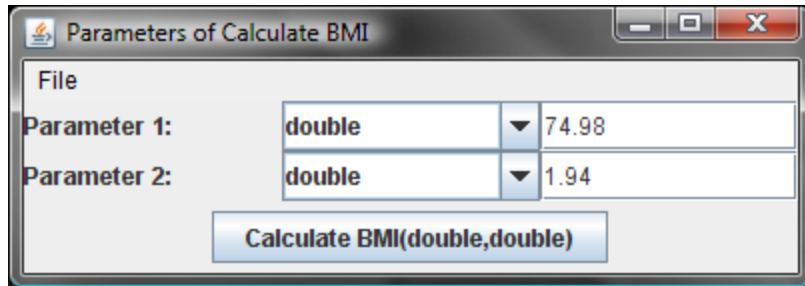
HOW TO REUSE ABMICALCULATOR

- Create an instance of ABMICalculator
- Invoke the method calculateBMI() on this instance passing it my weight and my height as actual parameters
- The value returned by the method is my BMI

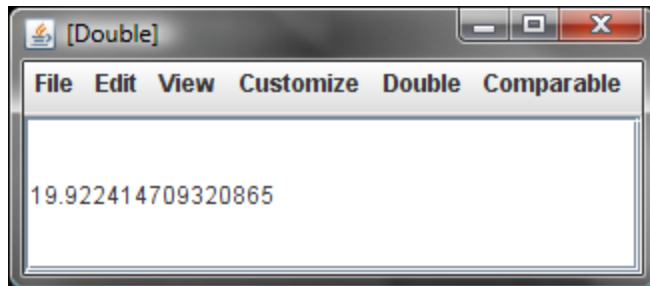
INTERACTIVE EXECUTION OF THE STEPS



- Create an instance of ABMCalculator



- Invoke the method `calculateBMI()` on this instance passing it my weight and my height as actual parameters



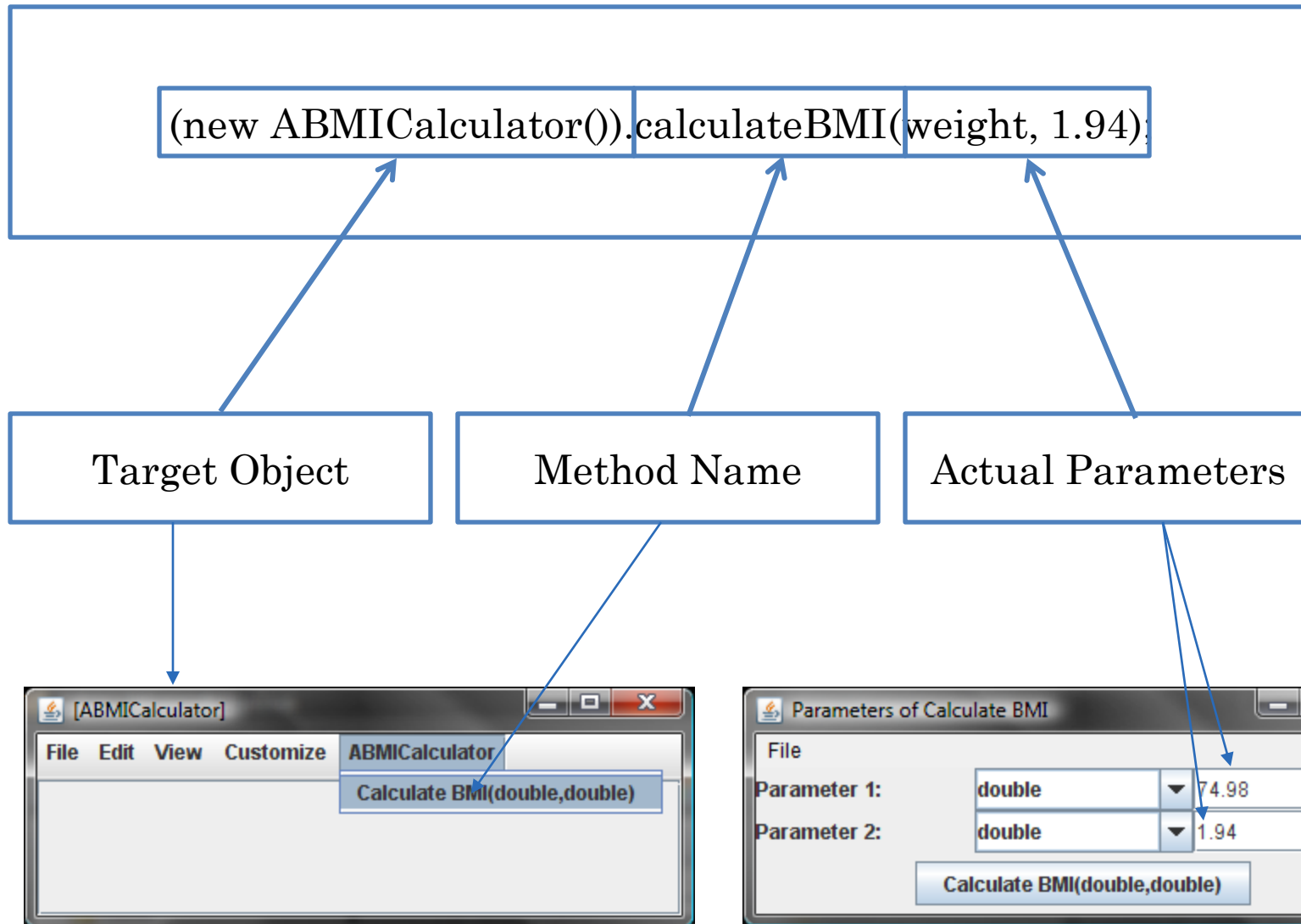
- The value returned by the method is my BMI

PROGRAMMING THE STEPS

- Create an instance of ABMICalculator
- Invoke the method calculateBMI() on this instance passing it my weight and my height as actual parameters
- The value returned by the method is my BMI

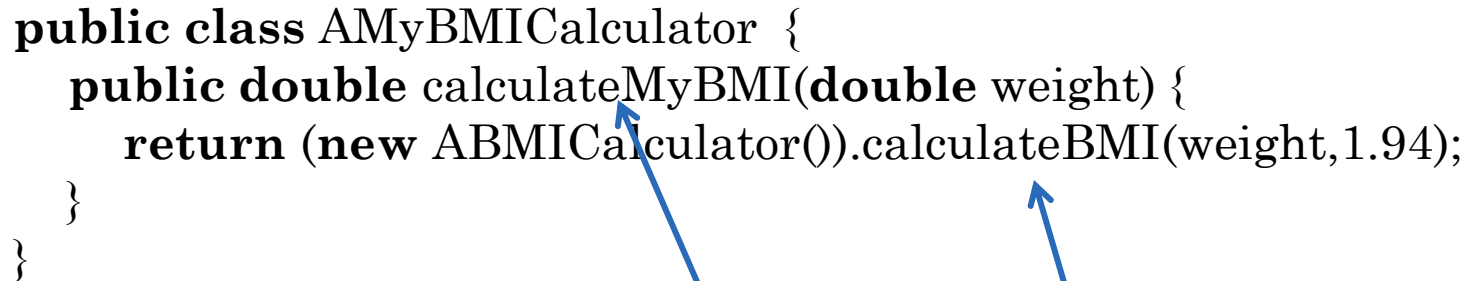
```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        return (new ABMICalculator()).calculateBMI(weight, 1.94);  
    }  
}
```

METHOD INVOCATION SYNTAX



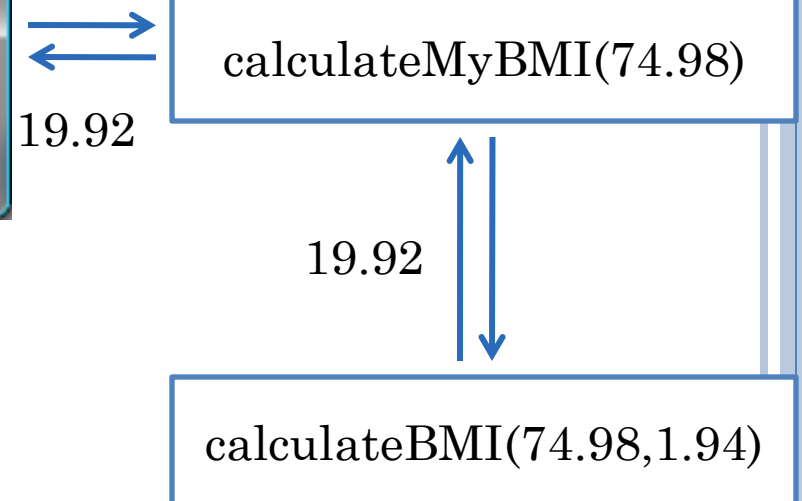
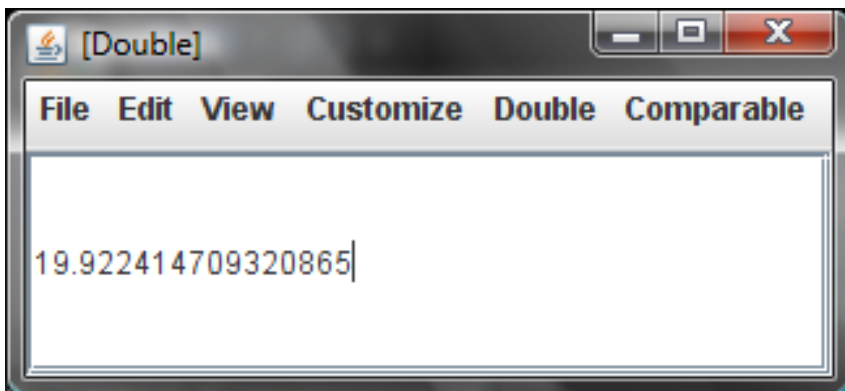
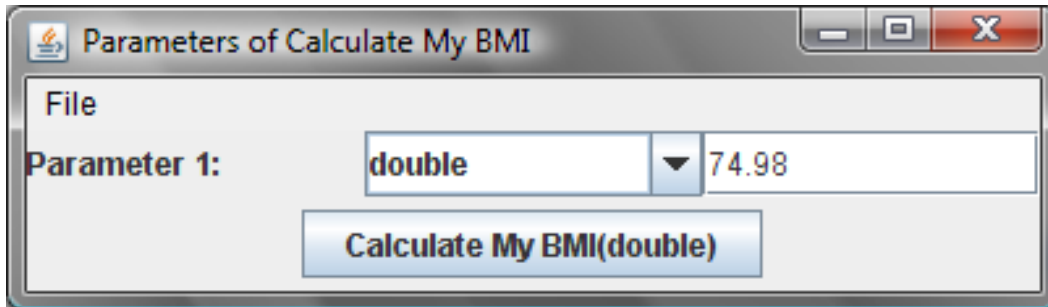
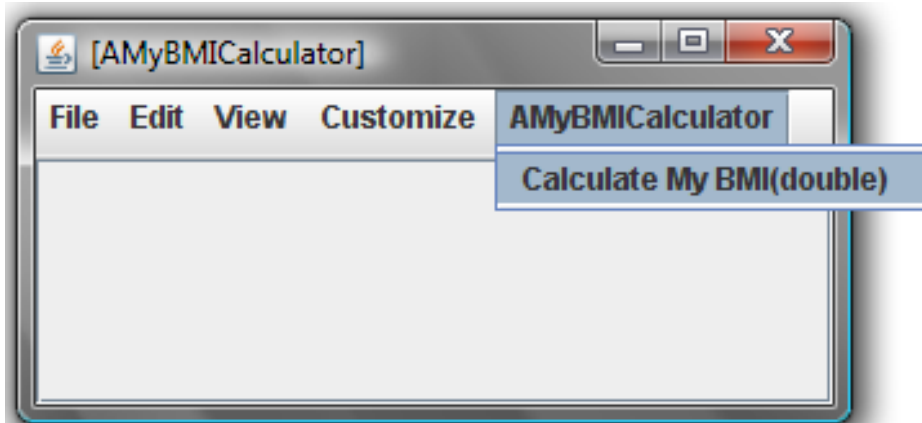
FUNCTION COMPOSITION

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        return (new ABMICalculator()).calculateBMI(weight,1.94);  
    }  
}
```

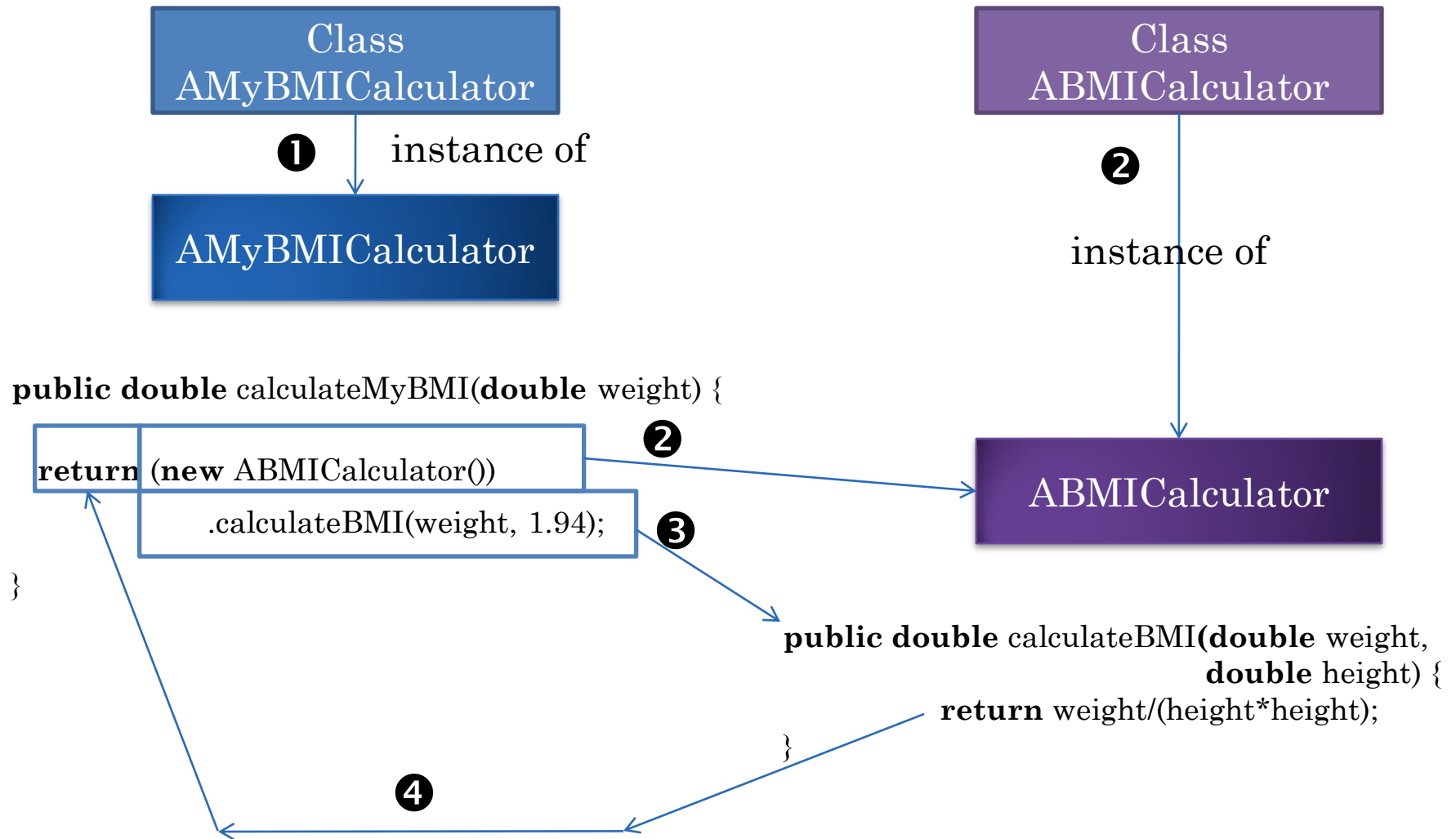


- The body of the *calling function* calls (invokes) other functions to do its job
- Passes the “buck” to the *callee or called functions*
- calculateMyBMI() calls calculateBMI()
- Supports reuse

CALL GRAPHS

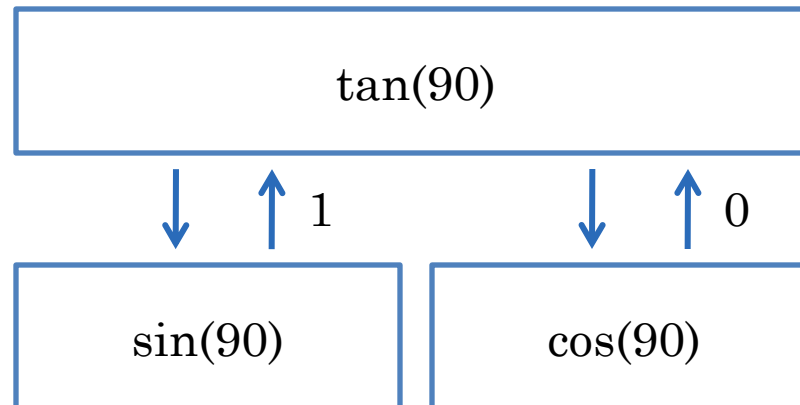


GRAPHICAL ILLUSTRATION OF THE CALCULATEMYBMI CALL

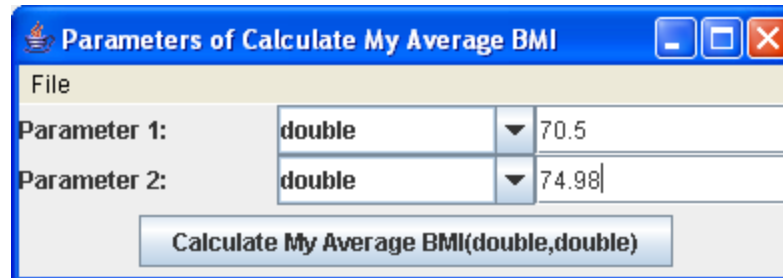
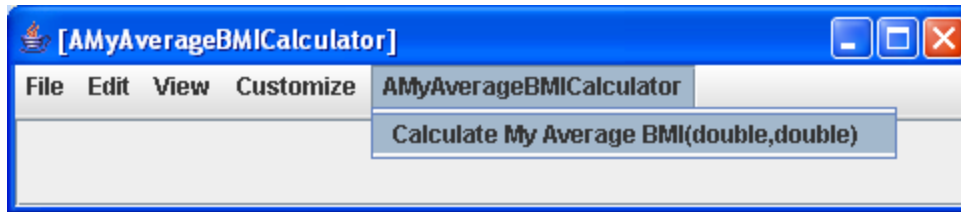


MATHEMATICAL INTUITION BEHIND FUNCTION INVOCATION

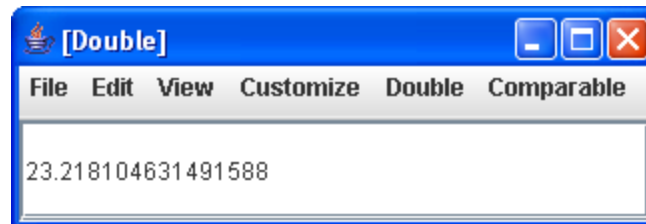
$$\tan(x) = \sin(x) / \cos(x)$$



AVERAGEBMICALCULATOR



← Weight 1
← Weight 2



A SOLUTION

```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                         double weight2) {  
  
    }  
}
```

A SOLUTION (EDIT)

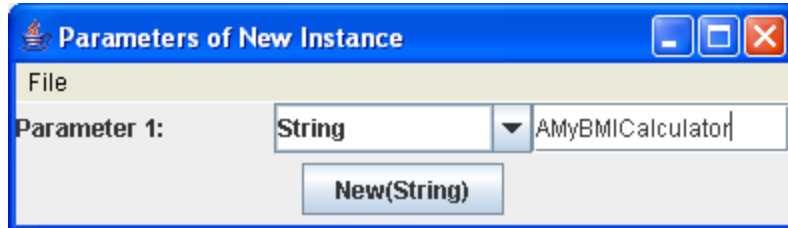
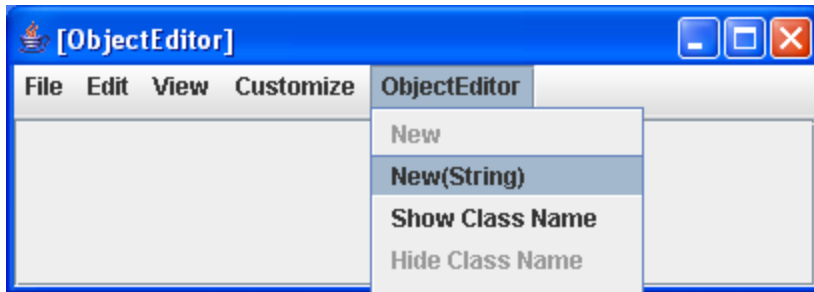
```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                         double weight2) {  
        <edit here>  
    }  
}
```

A SOLUTION

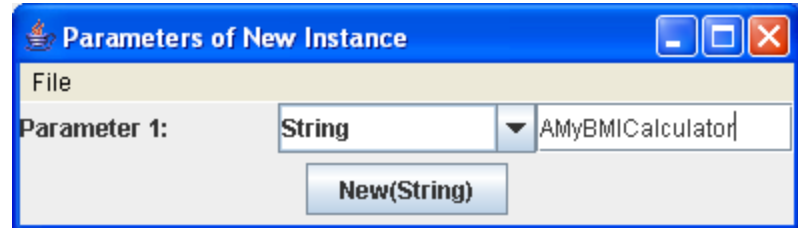
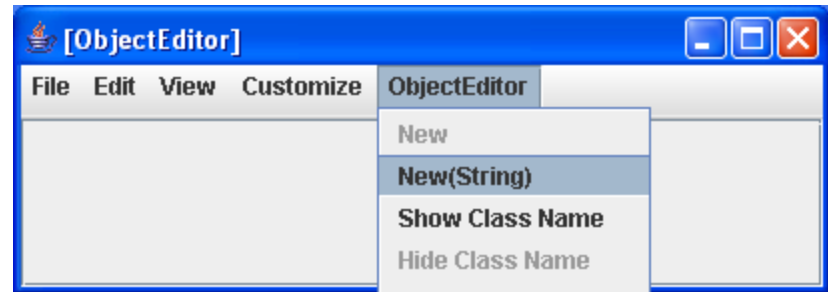
```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                         double weight2) {  
        return ((new ABMICalculator()).calculateBMI(weight1, 1.94) +  
                (new ABMICalculator()).calculateBMI(weight2, 1.94))/2;  
    }  
}
```

Creating a new instance of AMyAverageBMICalculator each time calculateBMI is to be called!

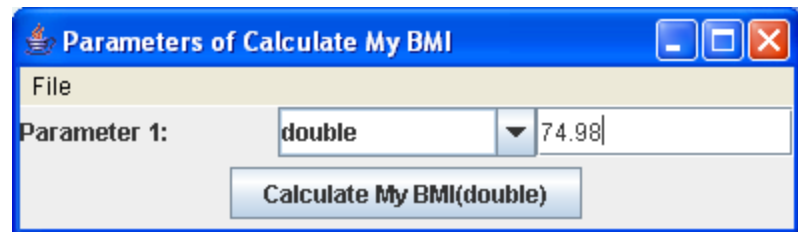
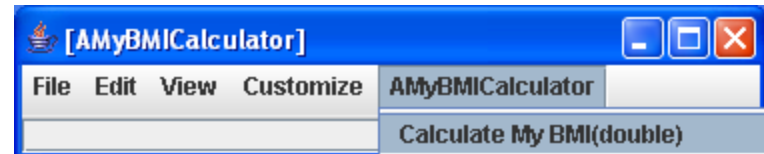
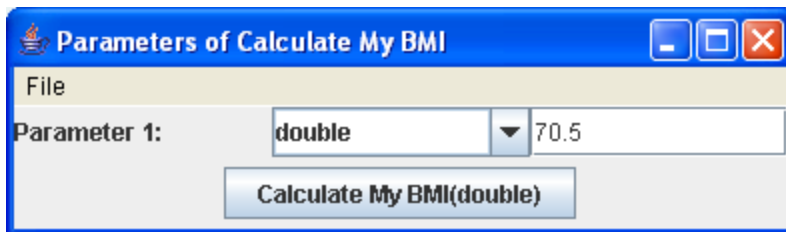
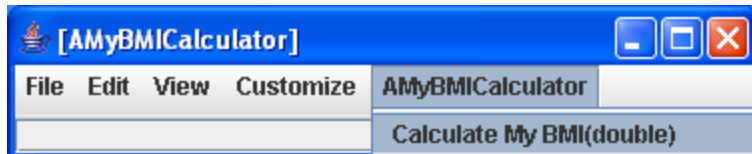
INTERACTIVE EQUIVALENT



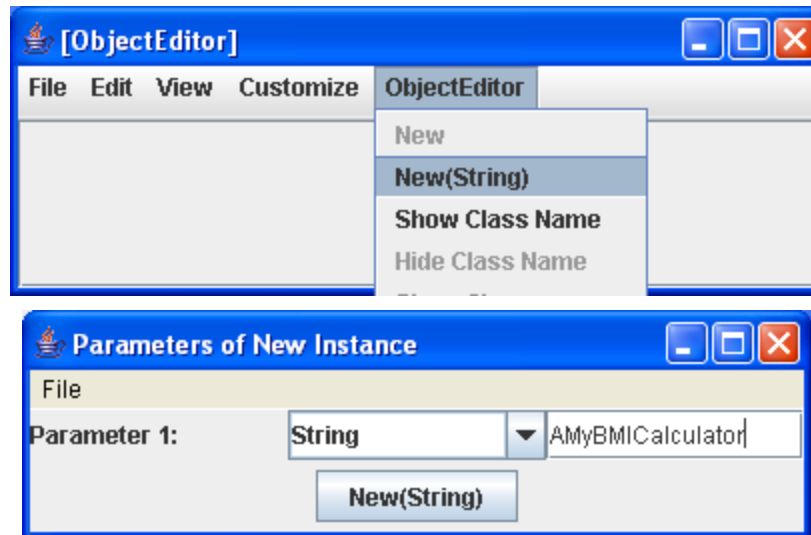
Instance 1



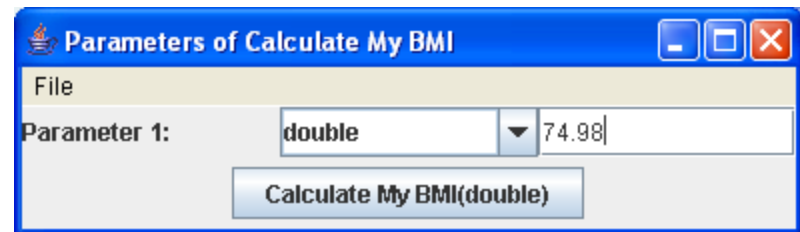
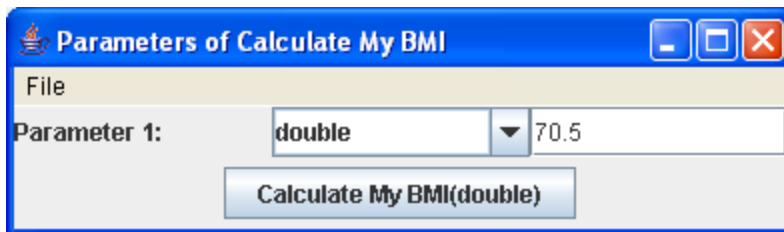
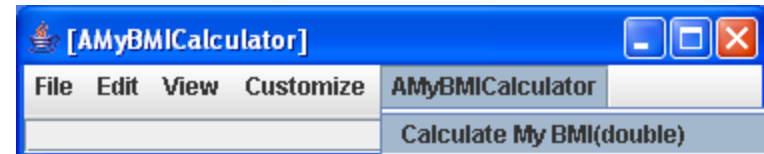
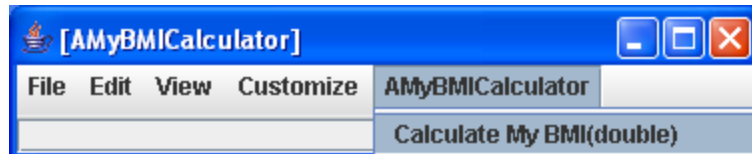
Instance 2



A BETTER INTERACTIVE APPROACH



Instance 1



ObjectEditor window identifies the appropriate instance. Need way to name objects in a program.

NAMING MEMORY LOCATIONS

- Can name values in a program by using variables
- Each program value stored in a memory location
- Variable declarations name memory locations
- Have already seen variable declarations!

FORMAL PARAMETERS AS VARIABLES

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        return weight/ (1.94 * 1.94);  
    }  
}
```

- Formal parameters are special kinds of variables.
- `weight` is name of memory location that stores the actual parameter passed by caller

INTERNAL METHOD VARIABLES

- Like formal parameters are declared with type and name
- Name in subsequent code refers to value stored in memory location
- Declared in a method body rather than header
- Can be explicitly given initial values
- Which can be changed later
- Make program more efficient as an extra object is not instantiated

```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                         double weight2) {  
        ABMICALculator aBMICALculator = new ABMICALculator();  
        return (aBMICALculator.calculateBMI(weight1, 1.94) +  
                aBMICALculator.calculateBMI(weight2, 1.94))/2;  
    }  
}
```

MORE USE OF VARIABLES

- bmi1 and bmi2 name memory locations that store the two intermediate results
- Not really needed to make programs efficient

```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                         double weight2) {  
        ABMICALculator aBMICALculator = new ABMICALculator();  
        double bmi1 = aBMICALculator.calculateBMI(weight1, 1.94);  
        double bmi2 = aBMICALculator.calculateBMI(weight2, 1.94);  
        return (bmi1 + bmi2)/2;  
    }  
}
```

WHICH IS BETTER?

```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                       double weight2) {  
        AMyBMICalculator aMyBMICalculator = new AMyBMICalculator();  
        return (aMyBMICalculator.calculateMyBMI(weight1) +  
                aMyBMICalculator.calculateMyBMI(weight2)) / 2;  
    }  
}
```

```
public class AMyAverageBMICalculator {  
    public double calculateMyAverageBMI(double weight1,  
                                       double weight2) {  
        AMyBMICalculator aMyBMICalculator = new AMyBMICalculator();  
        double bmi1 = aMyBMICalculator.calculateMyBMI(weight1);  
        double bmi2 = aMyBMICalculator.calculateMyBMI(weight2);  
        return (bmi1 + bmi2) / 2;  
    }  
}
```

- First solution is more concise
- Second solution separates various steps, giving names to each intermediate calculated value
- Hard to argue between them
 - Second solution makes it easier to single-step through code

PROGRAMMING STYLE

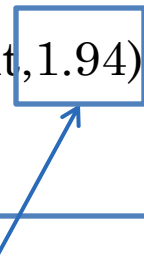
- More than one solution to a problem
- Some solutions arguably “better” than others
 - E.g. one solution allows reuse other does not.
- Programming style determines which solution is chosen
- Style as important as correctness
- Good style often promotes correctness

STYLE RULES

- Elements of Style
 - Support code reuse
 - Other style rules?

IMPROVING THE STYLE

```
public class AMyBMICalculatorWithReuse {  
    public double calculateMyBMI(double weight) {  
        return (new ABMICalculator()).calculateBMI(weight, 1.94);  
    }  
}
```

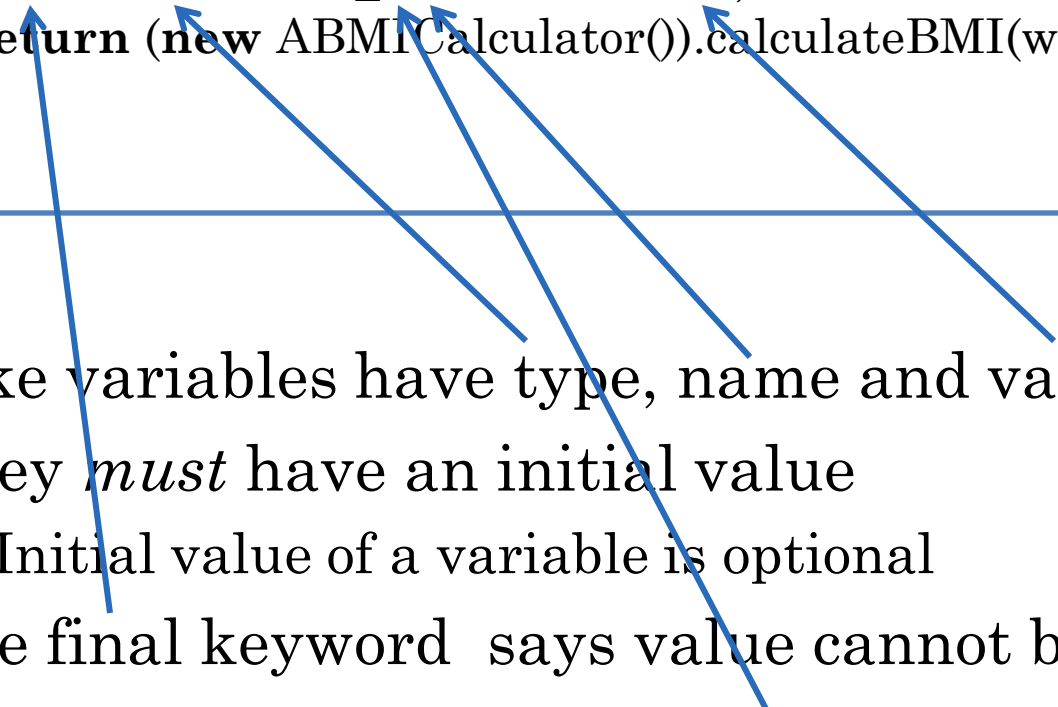


A Magic Number

A mysterious (at least
to some) number in
code

NAMED CONSTANTS

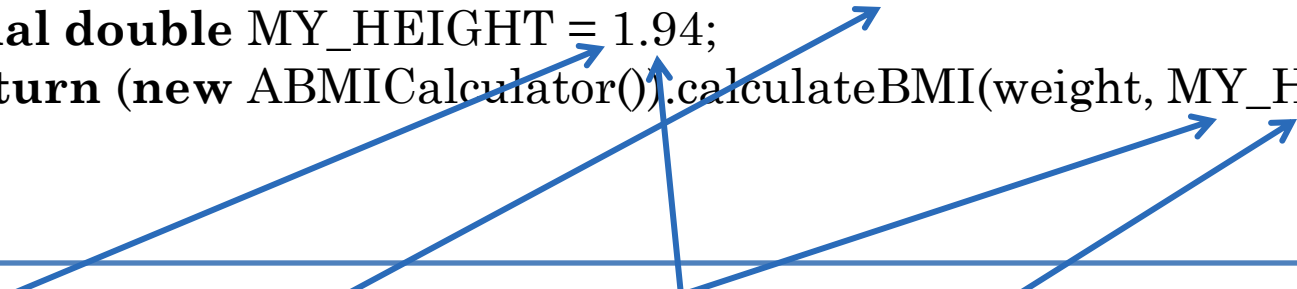
```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        final double MY_HEIGHT = 1.94;  
        return (new ABMCalculator()).calculateBMI(weight, MY_HEIGHT);  
    }  
}
```



- Like variables have type, name and value
- They *must* have an initial value
 - Initial value of a variable is optional
- The final keyword says value cannot be changed later
- The name is all caps by convention

NAMED CONSTANTS, LITERALS, CONSTANTS & VARIABLES

```
public class AMyBMICalculator {  
    public double calculateMyBMI(double weight) {  
        final double MY_HEIGHT = 1.94;  
        return (new ABMICalculator()).calculateBMI(weight, MY_HEIGHT);  
    }  
}
```



Literal

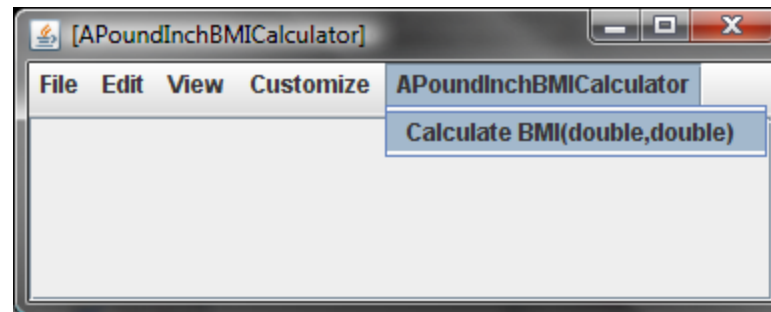
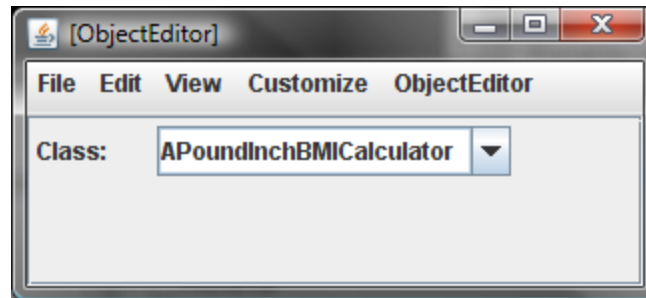
Variable

Constant

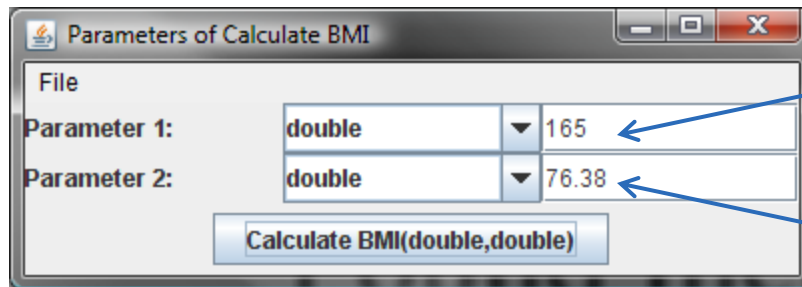
Named
Constant

- Literal
 - A value directly specified in the program
- Constant
 - A fixed value
 - Can be literal or named constant
- Variable
 - A potentially variable value

POUND INCH BMI CALCULATOR



POUND INCH BMI CALCULATOR



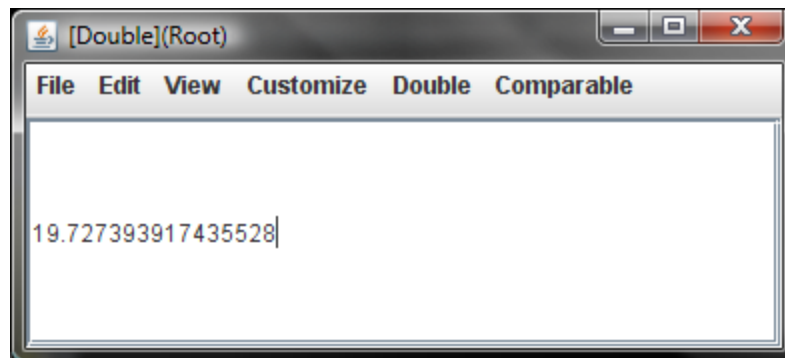
A dialog box titled "Parameters of Calculate BMI" with a standard Windows window frame. It contains two parameter entries. "Parameter 1:" is followed by a dropdown menu showing "double" and a text field containing "165". "Parameter 2:" is followed by a dropdown menu showing "double" and a text field containing "76.38". Below these is a button labeled "Calculate BMI(double,double)".

Parameter	Type	Value
Parameter 1:	double	165
Parameter 2:	double	76.38

Calculate BMI(double,double)

Weight in pounds

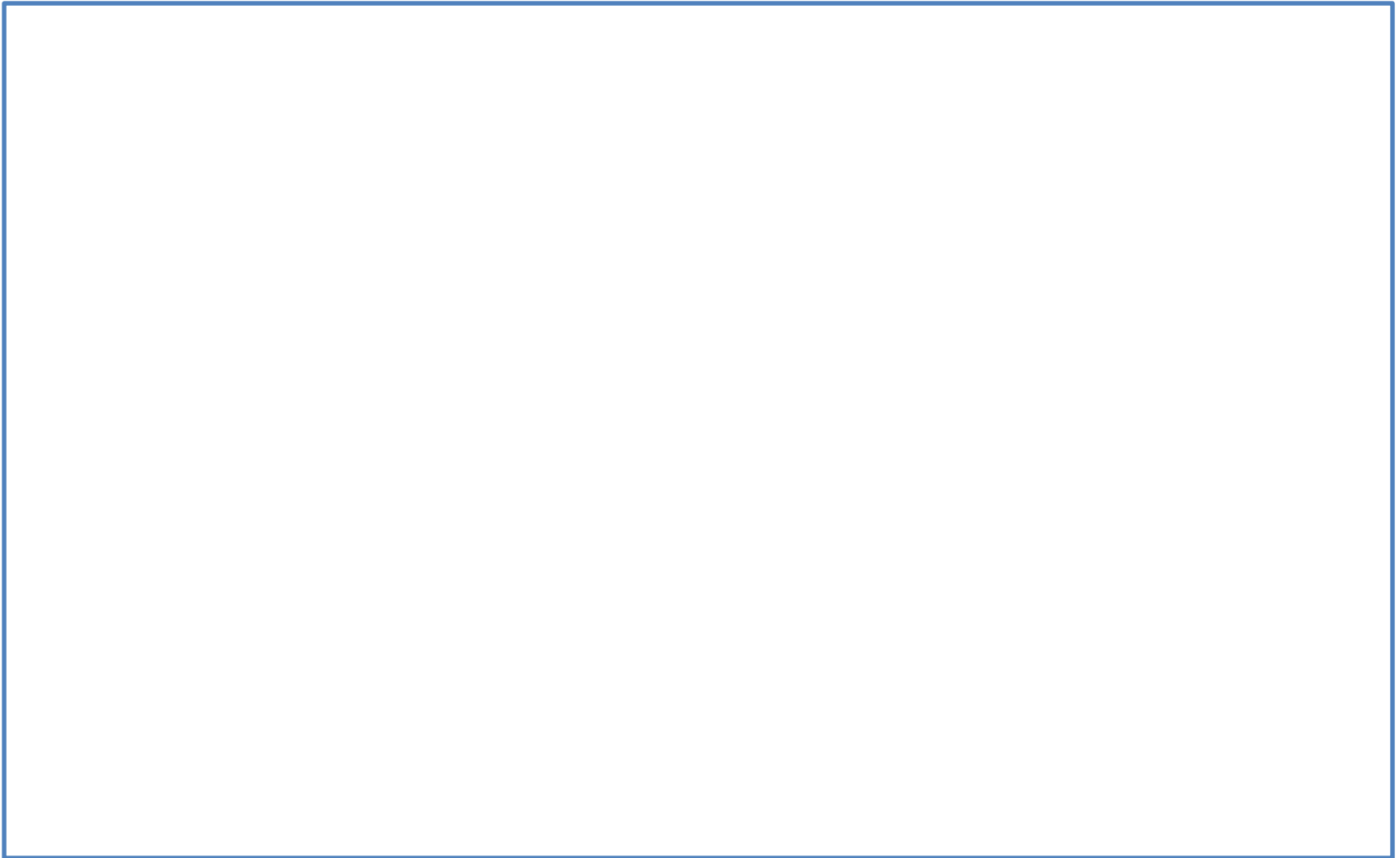
Height in inches



A window titled "[Double](Root)" with a menu bar containing "File", "Edit", "View", "Customize", "Double", and "Comparable". The main area is a text field containing the number "19.727393917435528" followed by a cursor.

19.727393917435528|

STEPS FOR REUSING ABMICALCULATOR



STEPS FOR REUSING ABMICALCULATOR



STEPS FOR REUSING ABMICALCULATOR

- Calculate weight in Kgs from weight in Pounds
- Calculate height in Metres from height in inches
- Call calculateBMI() of ABMICalculator with these values
- Return the value returned by this call

A SOLUTION

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches)  
    {  
  
    }  
}
```


A SOLUTION (EDIT)

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches)  
    {  
  
    }  
}
```

A SOLUTION (EDIT)

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches)  
    {  
        return (new ABMICalculator()).calculateBMI(  
            weightInLbs/2.2, heightInInches*2.54/100);  
    }  
}
```

ALGORITHM

- Description of solution to a problem.
- Can be in any “language”
 - graphical
 - natural or programming language
 - natural + programming language (pseudo code)
- Can describe solution to various levels of detail

REAL-WORLD ALGORITHM

- Enter Class
- Distribute handouts
- Set up laptop projection.
- Revise topics learnt in the last class.
- Teach today's topics.
- Leave Class

ALGORITHM FOR REUSING ABMICALCULATOR

- Calculate weight in Kgs from weight in Pounds
- Calculate height in Metres from height in inches
- Call calculateBMI() of ABMICalculator with these values
- Return the value returned by this call

2ND LEVEL ALGORITHM

- Calculate weight in kgs from weight in Pounds
 - Divide weight in Pounds by 2.2
- Calculate height in Meters from height in inches
 - Calculate height in centimeters from height in inches and divide it by 100 to get height in meters
- Call `calculateBMI()` of `ABMICalculator` with these values
- Return the value returned by this call

3RD LEVEL ALGORITHM

- Calculate weight in kgs from weight in Pounds
 - Divide weight in Pounds by 2.2
- Calculate height in Metres from height in inches
 - Calculate height in centimetres from height in inches and divide it by 100 to get height in metres
 - Multiply height in Inches by 2.54 to get height in centimetres
- Call `calculateBMI()` of `ABMICalculator` with these values
- Return the value returned by this call

STEPWISE REFINEMENT

Natural Language Algorithm

- Calculate weight in Kgs from weight in Pounds
- Calculate height in Meters from height in inches
- Call calculateBMI() of ABMCalculator with these values
- Return the value returned by this call



Programming Language Algorithm

```
public class APoundInchBMICalculator () {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
  
        return (new ABMCalculator()).calculateBMI(  
            weightInLbs/2.2, heightInInches*2.54/100);  
    }  
}
```


STYLE PROBLEMS

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches)  
    {  
        return (new ABMICalculator()).calculateBMI(  
            weightInLbs/2.2, heightInInches*2.54/100);  
    }  
}
```

- Unlike algorithm, code is single-level
- By defining functions for each algorithm level we can create multi-level code
- Multi-level code would be more reusable as there are more parts that can be used independently

MULTI-LEVEL CODE

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
        return (new ABMCalculator()).calculateBMI(  
            toKgs(weightInLbs), toMetres(heightInInches));  
    }  
  
    public double toMetres(double heightInInches) {  
        ???  
    }  
  
    public double toKgs(double weightInLbs) {  
        ???  
    }  
}
```

- Can be reused in contexts other than BMI (designing for reuse).
- Design for Reuse vs. Reusing available code

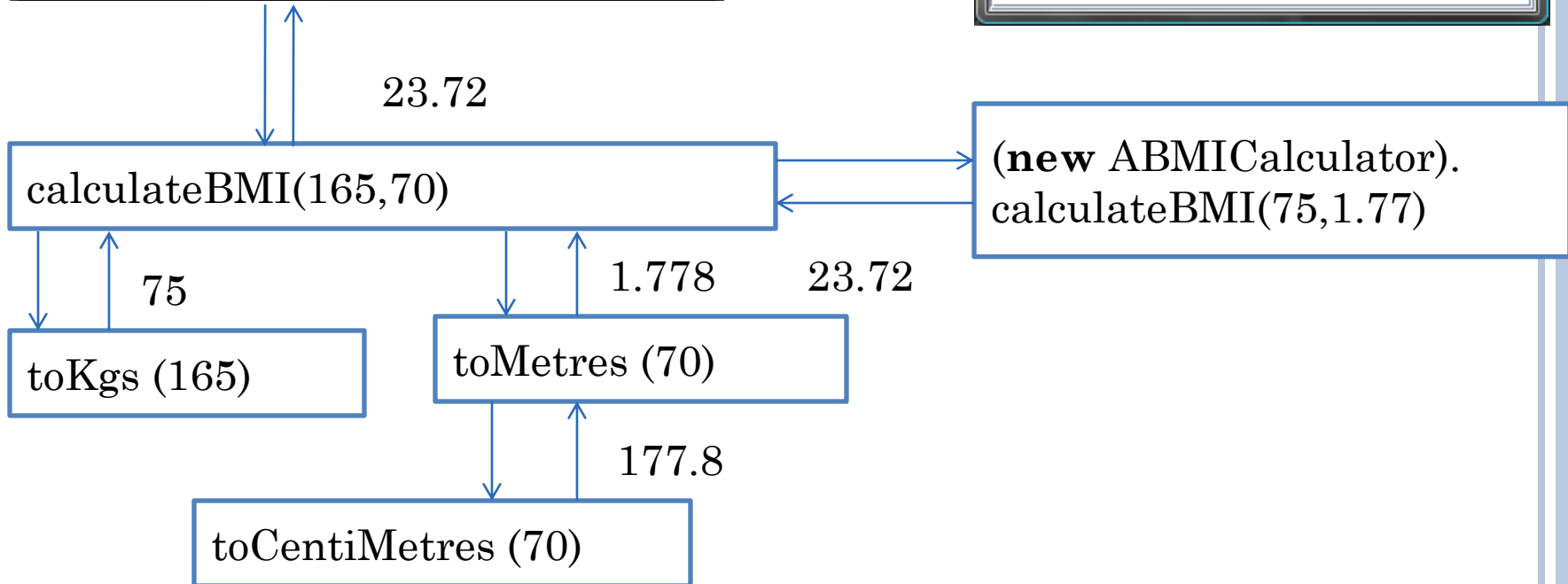
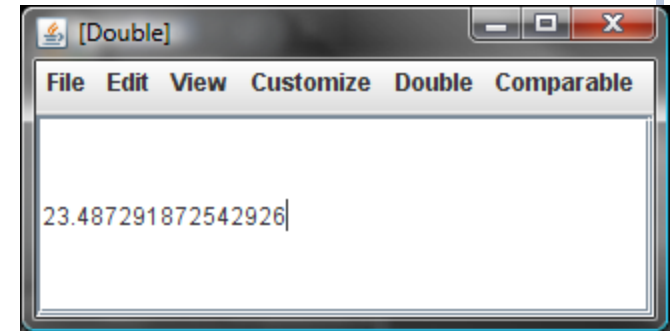
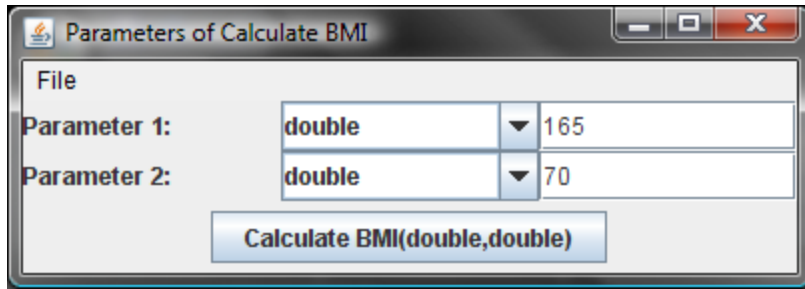
MULTI-LEVEL CODE (EDIT)

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
        return (new ABMICalculator()).calculateBMI(  
            toKgs(weightInLbs), toMetres(heightInInches));  
    }  
  
    public double toMetres(double heightInInches) {  
        ???  
    }  
  
    public double toKgs(double weightInLbs) {  
        ???  
    }  
}
```

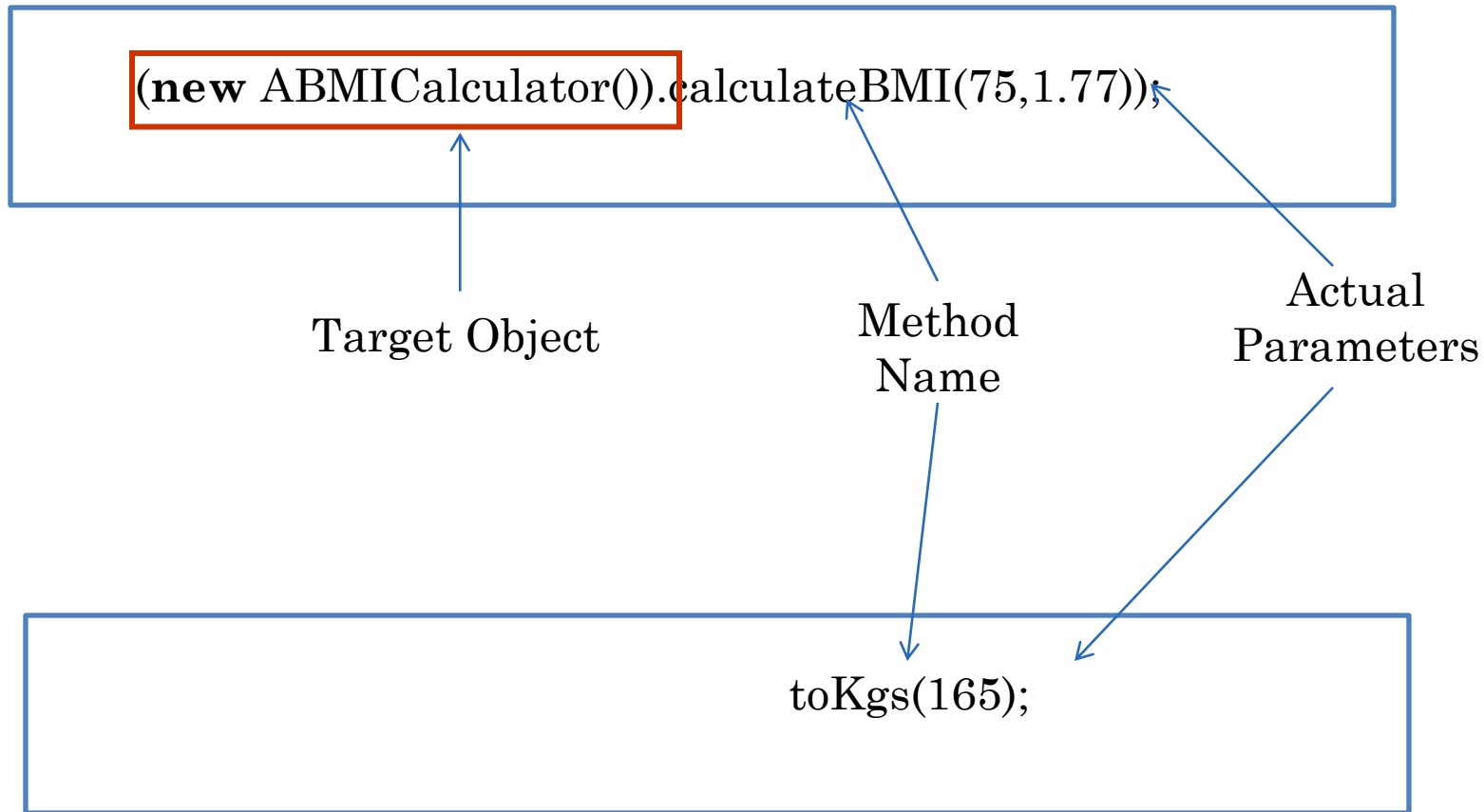
MULTI-LEVEL CODE

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
        return (new ABMICalculator()).calculateBMI(  
            toKgs(weightInLbs), toMetres(heightInInches));  
    }  
  
    public double toMetres(double heightInInches) {  
        return toCentiMetres(heightInInches)/100;  
    }  
    public double toCentiMetres(double heightInInches) {  
        return heightInInches*2.54;  
    }  
    public double toKgs(double weightInLbs) {  
        return weightInLbs/2.2;  
    }  
}
```

MULTI-LEVEL CALL GRAPH



EXTERNAL VS. INTERNAL METHOD INVOCATION SYNTAX



EXTERNAL VS. INTERNAL METHOD INVOCATION SYNTAX

- External Method Call
 - Caller (calling method) and callee (called method) belong to different objects
 - calculateBMI() of APoundInchBMICalculator instance calls calculateBMI() of ABMICalculator instance
- Internal Method Call
 - Caller and callee methods belong to same object
 - calculateBMI() of APoundInchBMICalculator instance calls toKgs() of APoundInchBMICalculator instance
- Target object optional in internal method call
- Target object needed because multiple objects may have the same method
- When target object omitted caller's object is target object

REUSABILITY

```
public class APoundInchBMICalculator {  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
        return (new ABMICalculator()).calculateBMI(  
            toKgs(weightInLbs), toMetres(heightInInches));  
    }  
    public double toMetres(double heightInInches) {  
        return toCentiMetres(heightInInches)/100;  
    }  
    public double toCentiMetres(double heightInInches) {  
        return heightInInches*2.54;  
    }  
    public double toKgs(double weightInLbs) {  
        return weightInLbs/2.2;  
    }  
}
```

Can be reused in other
classes

`(new APoundInchBMICalculator()).toCentiMetres(70)`

`(new APoundInchBMICalculator()).toMetres(70)`

`(new APoundInchBMICalculator()).toKgs(165)`

LACK OF REUSABILITY

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches)  
    {  
        return (new ABMICalculator()).calculateBMI(  
            weightInLbs/2.2, heightInInches*2.54/100);  
    }  
}
```

- A single method implements all three conversions
- Cannot reuse each conversion independent of BMI calculation

MAGIC NUMBERS REVISITED

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
        return (new ABMICalculator()).calculateBMI(  
            toKgs(weightInLbs), toMetres(heightInInches));  
    }  
  
    public double toMetres(double heightInInches) {  
        return toCentiMetres(heightInInches)/100;  
    }  
    public double toCentiMetres(double heightInInches) {  
        return heightInInches*2.54;  
    }  
    public double toKgs(double weightInLbs) {  
        return weightInLbs/2.2;  
    }  
}
```

magic numbers?

“WELL-KNOWN” VS. “OBSCURE” NUMBER

Well-Known Number
(Natural constant)



`weightInLbs/2.2`

Obscure Number



`(new ABMCalculator). calculateBMI(74, 1.77);`

WHAT IS A MAGIC NUMBER?

- Obscure number is a magic number
- Well-known number is not
 - A number defined by law of nature
 - e.g number of centimeters in an inch
 - Π
- What is well-known depends on the audience
 - e.g. number of centimeters in an inch
- Numbers defined by law of nature may not be considered magic numbers
- All other numbers should be considered magic numbers

REMOVING ALL POTENTIALLY MAGIC NUMBERS

```
public class APoundInchBMICalculator {  
  
    public double calculateBMI(  
        double weightInLbs, double heightInInches) {  
        return (new ABMICalculator()).calculateBMI(  
            toKgs(weightInLbs), toMetres(heightInInches));  
    }  
  
    public double toMetres(double heightInInches) {  
        final double CMS_IN_METRES = 100;  
        return toCentiMetres(heightInInches)/ CMS_IN_METRES;  
    }  
  
    public double toCentiMetres(double heightInInches) {  
        final double CMS_IN_INCH = 2.54;  
        return heightInInches* CMS_IN_INCH;  
    }  
  
    public double toKgs(double weightInLbs) {  
        final double LBS_IN_KG = 2.2;  
        return weightInLbs/LBS_IN_KG;  
    }  
}
```