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

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

One vs. two parameters

Basic formula is the same (can cut and paste)
**Should Reuse!**

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

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

Should reuse code to avoid duplication of effort and errors such as: (weight)/1.94

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
I NTERACTIVE E XECUTION OF THE S TEPS

- 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
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

```java
public class AMyBMICalculator {
    public double calculateMyBMI(double weight) {
        return (new ABMICalculator()).calculateBMI(weight, 1.94);
    }
}
```
**Method Invocation Syntax**

```
(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.

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

calculateMyBMI(74.98)

19.92

19.92

calculateBMI(74.98, 1.94)
public double calculateMyBMI(double weight) {
    return (new ABMICalculator() .calculateBMI(weight, 1.94);}

public double calculateBMI(double weight, double height) {
    return weight/(height*height);}
**MATHEMATICAL INTUITION BEHIND FUNCTION INVOCATION**

\[ \tan(x) = \frac{\sin(x)}{\cos(x)} \]

\[ \tan(90) \]

\[ \downarrow \quad \uparrow 1 \]

\[ \sin(90) \quad \cos(90) \]

\[ \downarrow \quad \uparrow 0 \]
AverageBMI Calculator

Weight 1  Weight 2
public class AMyAverageBMICalculator {
    public double calculateMyAverageBMI(double weight1, double weight2) {
    }
}
public class AMyAverageBMICalculator {
    public double calculateMyAverageBMI(double weight1, double weight2) {
        <edit here>
    }
}
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

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

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

```java
class AMyAverageBMICalculator {
    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

```java
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;
    }
}
```
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**

```java
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 ABMIColorCalculator()).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**

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

- **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

Weight in pounds

Height in inches
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
public class APoundInchBMICalculator {

    public double calculateBMI(
        double weightInLbs, double heightInInches)
    {
    }

    }
}
public class APoundInchBMICalculator {

    public double calculateBMI(
            double weightInLbs, double heightInInches)
    {
    }
}

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
2\textsuperscript{ND} 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 `calcuateBMI()` of ABMICalculator with these values
- Return the value returned by this call
3\textsuperscript{RD} 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 ABMICalculator with these values
- Return the value returned by this call

**Programming Language Algorithm**

```java
public class APoundInchBMICalculator {

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

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

calculateBMI(165, 70)
  75
toKgs (165)

23.72

toMetres (70)
  1.778

23.72
toCentiMetres (70)
  177.8

(new ABMICalculator). calculateBMI(75, 1.77)
EXTERNAL VS. INTERNAL METHOD INVOCATION SYNTAX

```
(new ABMICalculator()).calculateBMI(75, 1.77));
```

- **Target Object**: `ABMICalculator`
- **Method Name**: `calculateBMI`
- **Actual Parameters**: `75, 1.77`
**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
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`

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

(new APoundInchBMICalculator()).toMetres(70)

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

}
“Well-Known” vs. “Obscure” Number

Well-Known Number (Natural constant)

weightInLbs/2.2

Obscure Number

(new ABMICalculator). 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
    - \( \pi \)
- 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
public class APoundInchBMICalculator {

    public double calculateBMI(
            double weightInLbs, double heightInInches) {
        return (new ABMIColorator()).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;
    }
}