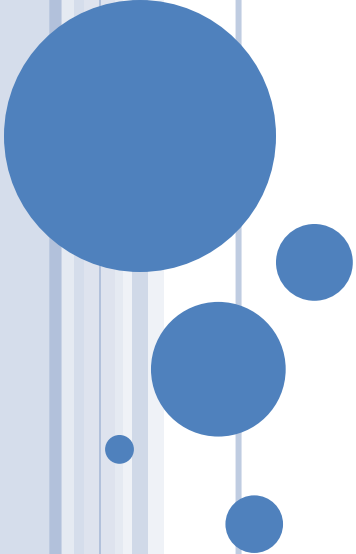




COMP 110/401

DOCUMENTATION: ASSERTIONS

Instructor: Prasan Dewan



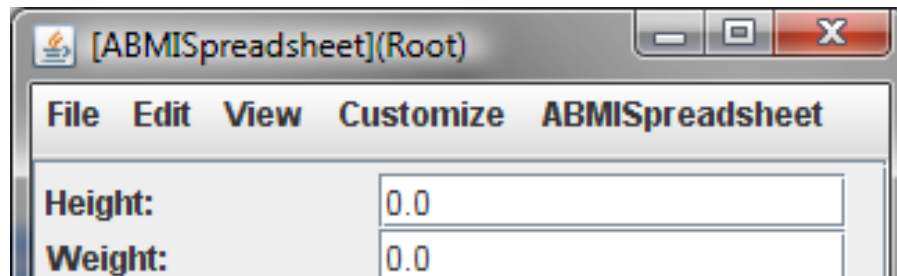
PREREQUISITE

- Documentation Annotations

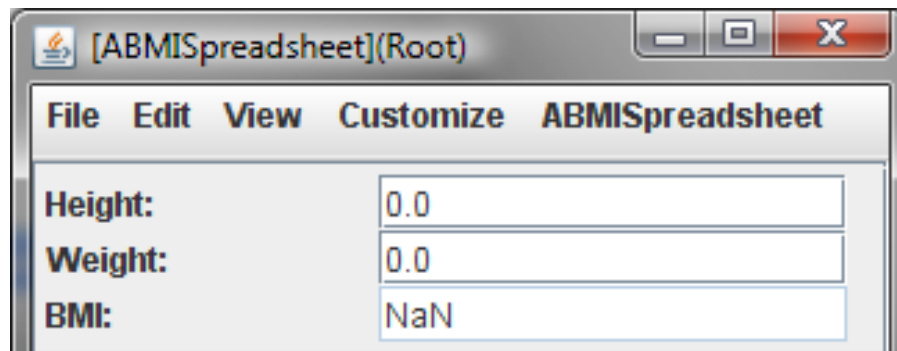


INVALID BMI

```
public double getBMI() {  
    return weight/(height*height);  
}
```



The screenshot shows a Java Swing window titled "[ABMISpreadsheet](Root)". It has a menu bar with "File", "Edit", "View", "Customize", and "ABMISpreadsheet". Below the menu bar, there are two input fields: "Height:" with the value "0.0" and "Weight:" with the value "0.0".



The screenshot shows the same Java Swing window as above, but now with an additional input field: "BMI:" with the value "NaN".

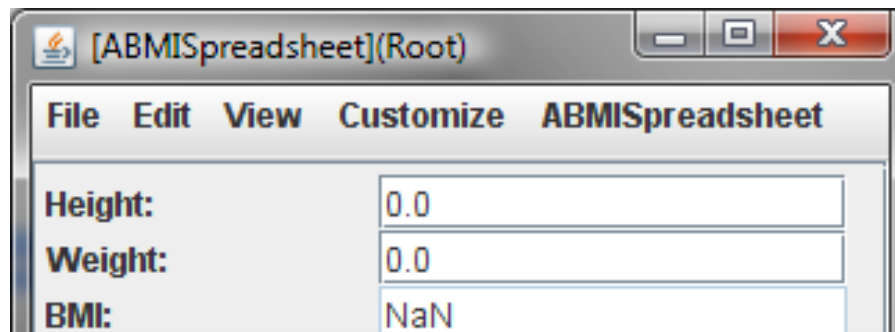
getBMI() should really not have been asked to compute with zero height and weight



COMMENTS TO DESCRIBE PRE-REQUISITE

```
/**  
 * height and weight should be >=0  
 */  
public double getBMI() {  
    return weight/(height*height);  
}
```

Height:	0.0
Weight:	0.0



The screenshot shows a window titled "[ABMISpreadsheet](Root)" with a menu bar containing "File", "Edit", "View", "Customize", and "ABMISpreadsheet". Below the menu bar, there are three input fields: "Height:" with the value "0.0", "Weight:" with the value "0.0", and "BMI:" with the value "NaN".

Height:	0.0
Weight:	0.0
BMI:	NaN

How to locate problem at execution time?



RUNTIME ERROR CHECKING

```
public double getBMI() {  
    if (weight <= 0 || height <= 0)  
        System.out.println("height and weight should be >0  
");  
    return weight / (height * height);  
}
```

Code is always executed even when
the program is correct

Value must be still returned

Conditional compilation of an "if" that
does not have to return a value ?



ASSERTIONS

```
public double getBMI() {  
    assert weight > 0 && height > 0: "height and weight  
should be >0";  
    return weight / (height * height);  
}
```

assert <Condition>: object →
announces assertion error
(object.toString() if !<Condition>)

Assertion error is like exception, no
return value needed

If assertion checking on

By default checking is off

Can enable/disable assertions for
specific classes and packages

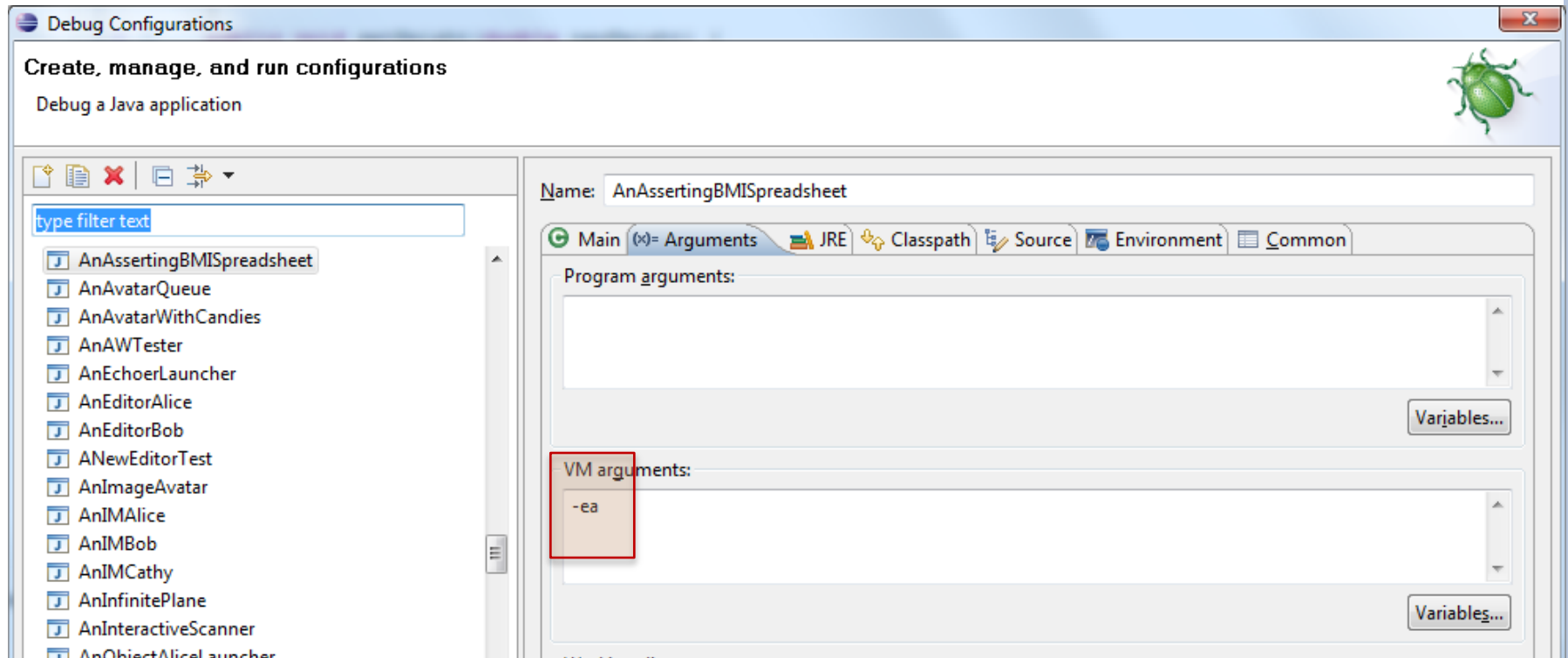
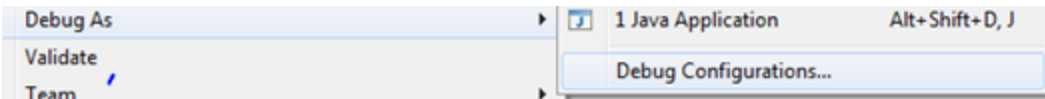
```
java -ea assignment9.MainClass -da bus.uigen
```

Enable assertions for MainClass

Disable assertions for bus.uigen package



ENABLING ASSERTIONS IN ECLIPSE



Enable all assertions



ASSERTIONS

- State some expected property of the program before/after some statement
 - Before `getBMI()` is called, height and weight should be greater than 0
- A la some expected property of an enrollee
 - Before 401 you must know loops, arrays, methods



COMPILE TIME VS. RUNTIME PROPERTIES

- Some “assertions” are language-supported
 - Compile time
 - `String s = nextElement()`
 - `@Override`
 - Runtime
 - `((String) nextElement())`
 - `@util.annotations.ObserverRegisterer(util.annotations.ObserverTypes.VECTOR_LISTENER)`
`addVectorListener(VectorListener)`
- We will consider runtime properties.
- Casting is application-independent.



APPLICATION-INDEPENDENT VS. DEPENDENT

- Language can provide us with fixed number of application-independent assertions.
- Cannot handle
 - First character of String is a letter.
 - Letter concept not burnt into language.
 - Class Character defines it
 - Innumerable assertions about letters possible
 - Second elements of string is letter.
 - Third element of string is letter.
- Need mechanism to express arbitrary assertions.
- Originally Java had no assertions.
- In 1.4, assertions were added



JAVA ASSERTIONS

- **assert** <Boolean Expression>
- **assert** <Boolean Expression>: <Value>
- Statement can be inserted anywhere to state that some condition should be true
- If condition is false, Java throws `AssertionError`, and (by default):
 - depending on which **assert** used, prints either:
 - generic message saying assertion failed, or
 - <Value>.toString()
 - prints stack trace
 - terminates program
- No value needs be returned by the method in which the assertion fails



INDIVIDUAL STATEMENT VS. BLOCK OF CODE

- Assert statement
 - States some expected property of the program before/after some individual statement
- Preconditions/Postconditions of block of code (e.g. method)
 - States some expected property of the program before/after some block of code
- Invariant of block of code
 - Precondition that is also a postcondition



PRECONDITIONS, POSTCONDITIONS, INVARIANTS

```
public boolean preGetBMI () {  
    return weight > 0 && height > 0;  
}  
public double getBMI () {  
    assert preGetBMI ();  
    return weight / (height * height);  
}
```

Pre (post) condition of block of code: an assertion that is expected to be true before (after) the block is executed

A la course prerequisite (objectives)

Invariant of a piece of code: a precondition that is also a post condition

GPA > Threshold



ASSERTIONS AS DOCUMENTATION

```
public boolean preGetBMI() {  
    return weight > 0 && height > 0;  
}  
public double getBMI() {  
    assert preGetBMI();  
    return weight / (height * height);  
}
```

```
/**  
 * height and weight should be >0  
 */  
public double getBMI() {  
    return weight / (height * height);  
}
```



ASSERTION USES

- Potentially useful for
 - documentation
 - specification
 - testing
 - formal correctness
 - user-interface adaptation



PRECONDITION PUBLIC METHOD, UI ADAPTATION, AND CONVENTIONS, TESTING

```
public boolean preGetBMI() {  
    return weight > 0 && height > 0;  
}
```

```
public double getBMI() {  
    assert preGetBMI();  
    return weight / (height * height);  
}
```

Convention could also be used by Grading (Testing) Program

A user-interface class (e.g. ObjectEditor or manual View class) can hide or disable a widget displaying some component of a model if the precondition of the method for reading the component is false

Public method allows other classes to discover preconditions and not violate them

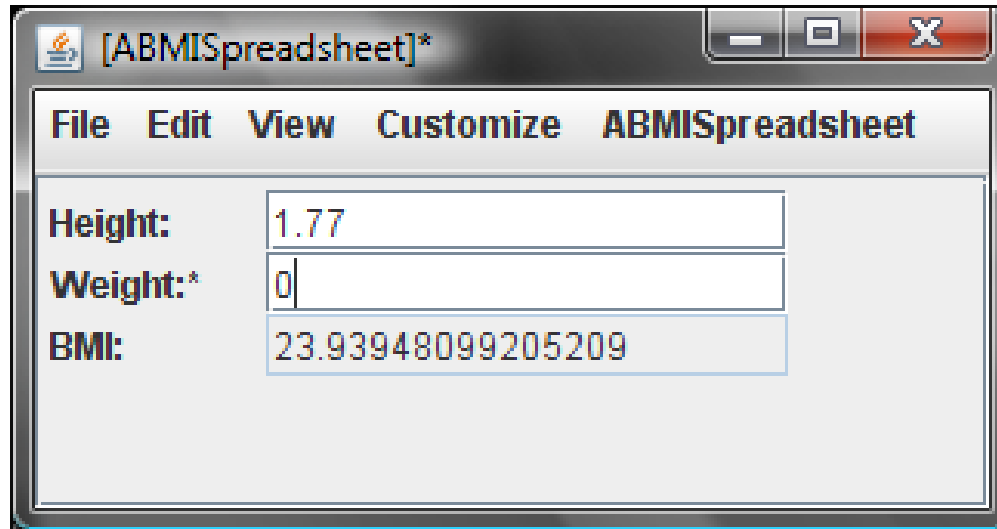
A user-interface class (e.g. OE or controller class) can disable a widget (e.g. menu item/text widget) for invoking a write method if its precondition is false

A user-interface class (ObjectEditor, manual controller or view) should not call a method if its precondition is false

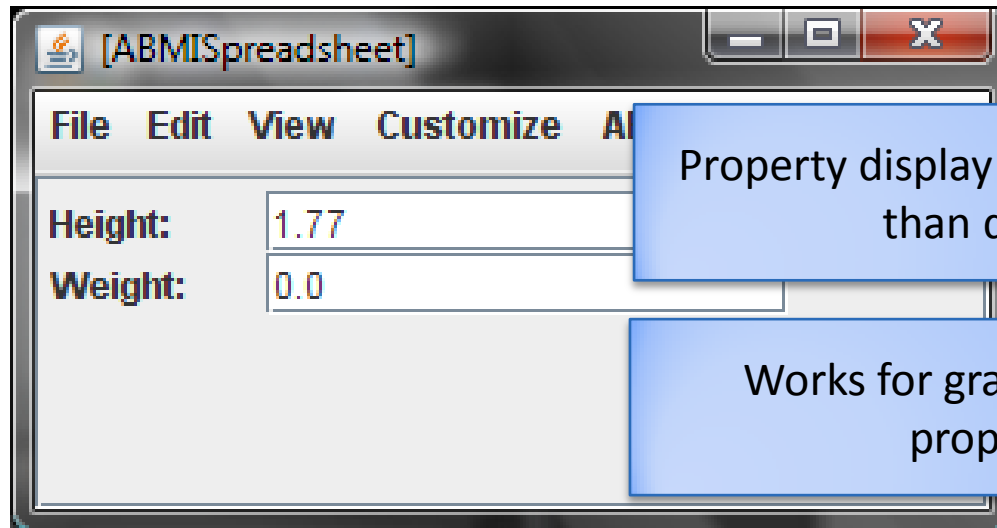
OE Convention: Precondition of method M() is preM(). M could be a read, write or some other method



PRECONDITION OF BMI IS TRUE (FALSE): DISPLAY SHOWN (FALSE)



The screenshot shows a window titled "[ABMISpreadsheet]*" with a menu bar containing "File", "Edit", "View", "Customize", and "ABMISpreadsheet". Below the menu bar, there are three input fields: "Height:" with the value "1.77", "Weight:*" with the value "0", and "BMI:" with the value "23.93948099205209".



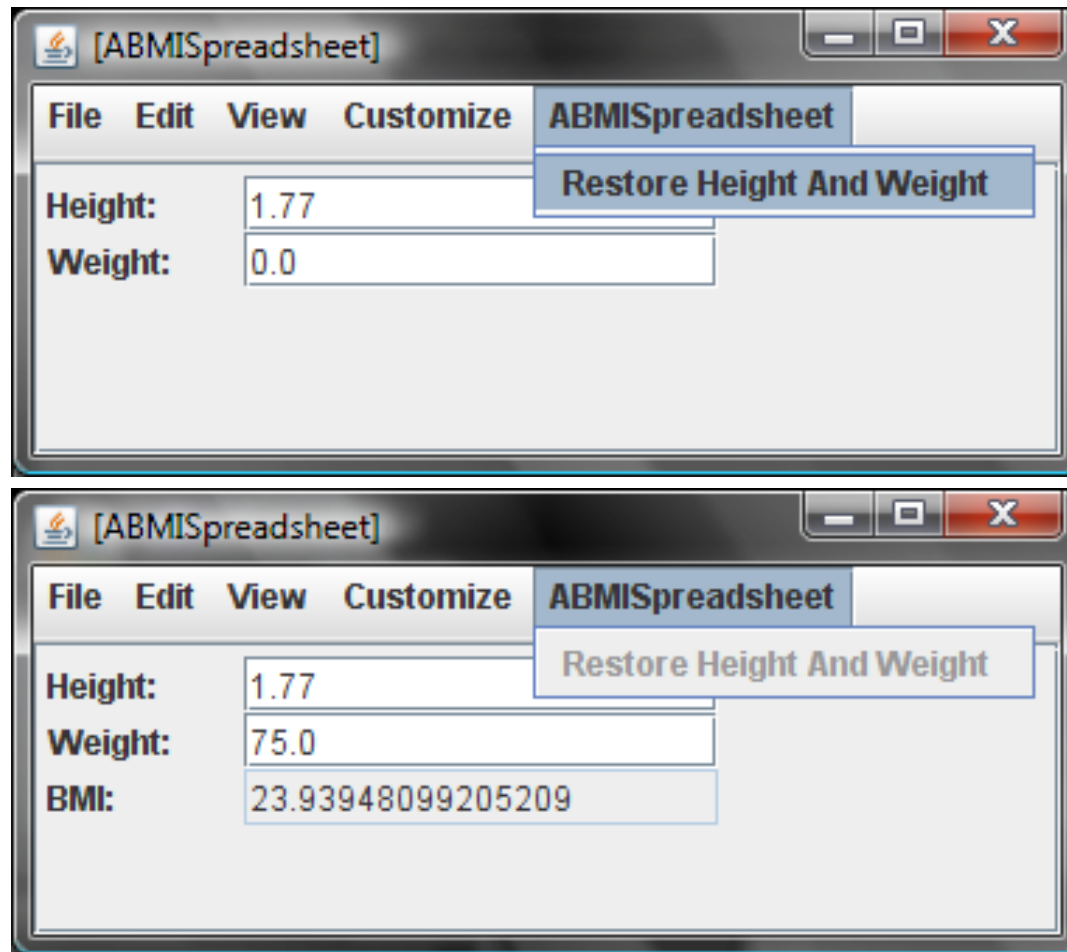
The screenshot shows the same window as above, but with a blue callout box overlaid on the right side. The callout box contains the text: "Property display is removed rather than disabled".

Property display is removed rather than disabled

Works for graphics and text properties



INPUT UI ITEM DISABLED/ENABLED



The menu item for a method is disabled when its precondition not met



RESTORING HEIGHT AND WEIGHT

```
public class AnAssertingBMISpreadsheet implements BMISpreadsheet {
    double height;
    double weight;
    double initialHeight, initialWeight;
    public AnAssertingBMISpreadsheet(
        double theInitialHeight, double theInitialWeight) {
        setHeight(theInitialHeight);
        setWeight(theInitialWeight);
        initialHeight = theInitialHeight;
        initialWeight = theInitialWeight;
    }
    public boolean preRestoreHeightAndWeight() {
        return height != initialHeight || weight != initialWeight;
    }
    public void restoreHeightAndWeight() {
        assert preRestoreHeightAndWeight();
        height = initialHeight;
        weight = initialWeight;
    }
    ...
}
```



METHOD PRECONDITION STYLE RULE

- If a method `M(...)` has a precondition that should be checked by another class, write a precondition boolean method, `preM()` for it that takes no arguments. (For overloaded methods there is a special rule we will not cover.)
- ObjectEditor will not invoke a method whose precondition is false and will not give the user a way to invoke it. If the method is a getter for a property, OE will not display the property.
- Call the precondition method in an **assert** statement before executing the method body.



METHOD ASSERTIONS

- Precondition: assertion true before the method is executed (regardless of parameters)
- Post condition: assertion true after the method is executed.
- Invariant: a precondition that is also a post condition



CLASS ASSERTIONS

- Class precondition: precondition of all public methods
- Class post condition: post condition of a all public methods
- Class invariant: invariant of all public methods (weight and height ≥ 0)



