Comp 110/401

Documentation: Annotations

Instructor: Prasun Dewan
PREREQUISITE

- Documentation Comments
PROGRAMMING STYLE: THE ART OF PROGRAMMING

- Use Interfaces
- Efficiency
- Document Program
- Avoid Code Repetition
- Giving Least Privilege
- Use Named Constants
DOCUMENTATION

Comments are one form of documentation about a program

External descriptions of a program are also useful

{@link http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#exampleresult}

Annotations: Internal descriptions available at runtime (to runtime tools) also useful
import util.annotations.Explanation;
@StructurePattern(StructurePatternNames.NO_PATTERN)
@Explanation("Height is in meters, weight in centimeters")
public class AnAnnotatedBMICalculator {
    public double calculateBMI(double height, double weight) {
        return weight/(height*height);
    }
}

Annotation not removed by compiler
Available to ObjectEditor
EXPLANATION ANNOTATION?

Calculates BMI from height in metres and weight in kgs. Stores previously computed BMI value in a variable.

Weight is in Kilograms. Changing weight automatically recalculates BMI.
@Explanation("Calculates BMI from height in metres and weight in kgs."")
public interface AnnotatedBMISpreadsheet {
    public double getHeight();
    public void setHeight(double newVal);
    @Explanation("Weight is in Kilograms. ")
    public double getWeight();
    @Explanation("Changing weight automatically recalculates BMI.")
    public void setWeight(double newVal);
    public double getBMI();
}

@Explanation("Stores previously computed BMI value in a variable.")
public class AnAnnotatedBMISpreadsheet implements AnnotatedBMISpreadsheet {
    AnnotatedBMISpreadsheet {...}
}
EXPLANATION AND UI DOCUMENTATION

- Menu item tooltip text
  - Explanation annotation of associated method
- Property tooltip text
  - Sum of getter and setter explanations
- Object tooltip text
  - Sum of interface and class explanations

Some documentation is outside a class

In PPT, PDF, and other Web Docs
public interface AnnotatedBMISpreadsheet
{
    ...
}

4. State

The computations performed by the code we wrote in the previous chapter were essentially the kind we would do with a calculator. The difference was that our code was tailored to a particular kind of calculation. While users of a calculator are responsible for entering the entire formula that is to be computed, the users of our classes were responsible only for entering the data — the formulas were built into the code. We essentially learnt how Java can be used to define the kinds of functions some calculators allow us to program.

A computer is, of course, more than a programmable calculator. Here we will see another, related use of it — to implement custom spreadsheets, which, like the custom calculators we saw before, have formulae burnt into them, requiring the users to enter only the data.

If you are familiar with how to program a calculator or a spreadsheet, Java might seem much too complicated for programming these examples. Languages tailored to a particular kind of computation are bound to be simpler than a general-purpose language such as Java. The purpose of these introductory chapters is to use these simple domains to explain aspects of programming that will be crucial for implementing more complex domains, which we will see later.
COMMON MENU

[AnAnnotatedBMISpreadsheet]

Common

- Refresh
- Tree
- New Editor: 99205209

Demo Font Size

- About Object Editor: Makes font size big enough to be easily seen in a demo projected to a medium-sized room.
- Documents
- Source

Common

- Height: 1.77
- Weight: 75.0
- BMI: 23.939480992052

10
COMMON MENU
Tooltip Texts in Common and Object Menu

- ObjectEditor used to build itself
- Meta circularity – a la Interpreter “interpreting” itself
- Bootstrapping – a la OS loading itself
- Requires a layered implementation
public class ADemoFontOperationsModel implements FrameModel {
    @Explanation ("Makes font size big enough to be easily seen in a 
demo projected to a medium-sized room.")
    public void demoFontSize() {
        ...
    }
    ...
}
import bus.uigen.ObjectEditor;
public class AnObjectEditorEditor {
    public static void main (String[] args) {
        ObjectEditor.edit(new ObjectEditor());
    }
}