Abstraction and Encapsulation
javadoc
More About if Statements

Lecture 16
Some slides borrowed from Kurt Eiselt and Beth Simon.

Readings
This Week: Ch 5.1-5.4 (Ch 6.1-6.4 in 2nd ed).
(Reminder: Readings are absolutely vital for learning this stuff!)

Labs and Tutorials
This week is Lab #5.

Midterms – Save the Dates!
- Midterm #1 is 5:30-6:30pm on February 10 (Tuesday) in Woodward IRC 2 (Done!)
- Midterm #2 is 6-7pm on March 11 (Wednesday) in Woodward IRC 2

Midterm Study Tips:
- Old midterms on-line.
- Try programming without notes.
- Try programming without computer!

Programming Assignment 1
- Assignment 1 is up on WebCT!
  - Click on the “Assignments” icon.
- Due at NOON, February 17 (Tuesday), via electronic hand in.
- Start early!

Learning Goals
By the end of the next several lectures you will be able to...
- Create your own classes, with:
  - Public and private fields and methods
  - Helpful documentation that works with javadoc
- Basic principles of abstraction and encapsulation (information hiding)
- Explain why abstraction and information hiding are important.
Learning Goals
By the end of class today you will be able to...

- Explain what “abstraction” is, and why it’s important.
- Explain what “information hiding” (also known as “encapsulation” or “data protection”) is, and why it’s important.
- Use javadoc to generate nice-looking documentation webpages.
- Write complicated, nested if statements.

Review: Mileage Computer
- Design a class for a trip computer in a car that computes the gas mileage
  - Every 1m, a distance sensor sends a signal to the computer.
  - Every 1 second, a fuel sensor sends a signal to the computer indicating how many ml of gas were used during the last second.
  - The measurements can be reset
  - We can query the computer for the gas mileage (reported as litres/100km) averaged since the last reset.

If you want, try doing “updated” specifications in my slides.

Managing Complexity
- Computer science creates the most complex artifacts ever created by humans:
  - Windows Vista has 50 million lines of code.
  - Current mainstream Intel processors (Core 2 Duo) have 410 million transistors; quad-core Core i7 chips have 731 million transistors.
- How do we manage to do this?
- Answer: Abstraction

Abstraction
- Abstraction means creating higher-level ways to think about things, so you can ignore lots of lower-level details.

Abstraction in Real Life
- Abstraction means creating higher-level ways to think about things, so you can ignore lots of lower-level details.
- You already do this all the time to manage complexity:
  - How do you get home after school?
  - How do you manage your time in a day?
  - How do you learn biology or chemistry or music or literature or …?

Abstraction in Software
- Abstraction means creating higher-level ways to think about things, so you can ignore lots of lower-level details.
- Creating new classes is a common and powerful way to create new abstractions:
  - How do you print information to the console window?
  - How do you make a Java program that can compute with arbitrarily big numbers?
  - How do you compute a logarithm?
Encapsulation
- Why is the gas cap on the outside of a car, while the places to add oil, coolant, brake fluid, etc., are under the hood?
- Why are the on/off, channel, and volume switches of a TV on the front, while other controls are hidden away?
- Why is the keyboard on a laptop obvious, while the jumpers to configure the disk drive are hidden inside?

Encapsulation
- A car, a TV, a laptop, these are all abstractions!
- They were intended by their creators to be used in certain ways, and not in other ways:
  - My mom adding water to the engine oil filler.
  - Phone phreaking
- The designer encapsulates the product to hide the details, to make it harder to misuse.

Encapsulation, Information Hiding, Data Protection
- In computer software, the same idea is also called data protection or information hiding.
- Parnas’s Law: Only what is hidden can be changed without risk:
  - If I change problem 2 on the final, does that upset you?
  - If I move the final to be RIGHT NOW instead, does that upset you?

Encapsulation, Information Hiding, Data Protection
- In computer software, the same idea is also called data protection or information hiding.
- Parnas’s Law: Only what is hidden can be changed without risk:
  - If Java 7 implements System.out.println slightly differently, does that upset people?
  - If Java 7 changes the name of System.out.println to be System.out.writeln, does that upset people?

Encapsulation, Information Hiding, Data Protection
- By making the instance fields private, we protect them from misuse.
  - They are accessible to others only through the public methods that we provide.
  - If we write these methods carefully, we can help prevent mistakes.

Designing a Class
- You want to provide a useful abstraction. It should let the user think about higher-level things, without worrying about the details.
- You use encapsulation to prevent other programmers (or yourself!) from misusing your class.
Questions?

Commenting Code

- Conventions
  - explain what classes and methods do
  - plus anywhere that you’ve done something non-obvious
    - usually better to say why than what
      - not useful
        int wishes = 3; // set wishes to 3
      - useful
        int wishes = 3; // follow fairy tale convention

---

**javadoc Comments**

- Specific format for method and class header comments
  - running javadoc program will automatically generate HTML documentation
- Rules
  - /** to start, first sentence used for method summary
  - @param tag for parameter name and explanation
  - @return tag for return value explanation
  - other tags: @author, @version, etc.
  - */ to end
- Running
  1. javadoc Die.java
  2. javadoc *.java

---

**javadoc Method Comment Example**

```java
/**
 * Sets the die shape, thus the range of values it can roll.
 * @param numSides the number of sides of the die
 */
public void setSides(int numSides) {
    sides = numSides;
}
```

---

**javadoc Class Comment Example**

```java
/**
 * Die: simulate rolling a die
 * @author: CPSC 111, Section 206, Spring 05-06
 * @version: Jan 31, 2006
 * This is the final Die code. We started on Jan 24,
 * tested and improved in on Jan 26, and did a final
 * cleanup pass on Jan 31.
 */
```
Java's if statement

- Syntax:
  ```java
  if ( condition ) statement;
  ```
- Examples:
  ```java
  if (tax < 0) tax = 0;
  
  if (age < 19)
      System.out.println("Sorry, you may not buy alcohol.");
  ```

Java's if statement

- Intuitively, a lot like English:
  If you're hungry, have a sandwich.
  ```java
  if (tax < 0) tax = 0;
  ```

Java's if statement with else

- Syntax:
  ```java
  if ( condition ) statement;
  else statement;
  ```
- Example:
  ```java
  if (age < 19) {
      System.out.println("Sorry, ");
      System.out.println("you may not ");
      System.out.println("buy alcohol.");
  }
  ```

Java's if statement with else

- Syntax:
  ```java
  if ( condition ) statement;
  else statement;
  ```
- Sort of like English:
  If it's sunny, we'll go hiking,
  or else we'll go see a movie.
  (It'd make more sense if the reserved word
   had been otherwise, but that's too long to
   type.)
Nested If Syntax

- Statements within if-else statements can themselves be if-else statements.
- Use curly braces to be clear.

```java
public class NestTest {
    public static void main (String[] args) {
        int x = 1; int y = 3; int z = 2;
        if (x == y)
            if (y == z)
                System.out.println("all three values the same");
            else
                System.out.println("y is not equal to z");
        else
            System.out.println("x is not equal to y");
    }
}
```

Body Mass Index (BMI) Calculator

- The Body Mass Index (BMI) is a standard medical test to give a quick and approximate indicator of healthy body weight.
- Your BMI is defined as your “weight” in kilograms divided by the square of your height in meters.
  - BMI < 19: Underweight, higher risk of disease
  - 19 <= BMI <= 25: Normal
  - 25 < BMI <= 30: Overweight, higher risk
  - 30 < BMI: Obese, much higher risk of disease

Nested if Syntax

- A common solution (others are possible):
  ```java
  if ( bmi < 19 )
      // underweight statements
  else if ( bmi <= 25 )
      // normal statements
  else if ( bmi <= 30 )
      // overweight statements
  else
      // obese statements
  ```

else if Syntax

- Easier to read indented like this:
  ```java
  if ( bmi < 19 )
      // underweight statements
  else if ( bmi <= 25 )
      // normal statements
  else if ( bmi <= 30 )
      // overweight statements
  else
      // obese statements
  ```

Dangling else

- What does this code do?
  ```java
  // print warnings only if BMI not normal
  if (bmi >= 19)
      if (bmi > 25) System.out.println("Over");
  else System.out.println("Under");
  ```

Beware the Dangling else

- What does this code do?
  ```java
  // print warnings only if BMI not normal
  if (bmi >= 19)
      if (bmi > 25) System.out.println("Over");
  else System.out.println("Under");
  ```
  ```java
  else goes with nearest if
      // (but doesn’t look inside curly braces)
      // common mistake
      // whitespace ignored by compiler
  ```
Use Curly Braces to Control Nesting

- What does this code do?
  // print warnings only if BMI not normal
  if (bmi >= 19) {
    if (bmi > 25) System.out.println("Over");
  }
  else System.out.println("Under");
- else goes with nearest if
  - but doesn't look inside curly braces