Lesson 40: Book Pricing (W12D1)

Balboa High School

Michael Ferraro

October 30, 2015

Do Now

1. Read Litvin Ch. 6, #17 from here.
2. The publisher's website, `http://skylit.com/`, used to take online orders. They've stopped doing so, but a replica of the old order form is here.
   - Insert a quantity near one of the titles, scroll to the bottom, and click $\boxed{\text{subtotal}}$.
   - Notice that, for certain titles, there's a volume discount.
   - Update the quantity for books, causing the subtotal to have to calculate a discount.
   - *How does it work?* Check the source code for the page! (`CTRL-U` in Firefox)

Aim

Students will use conditionals (e.g., `if()-else if`) to compute an order subtotal for a book order in which books are discounted under certain conditions.

Going over the Do Now. . .

Let's discuss the Do Now. . .

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if `y` is. . .

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if `y` is. . .

- evenly divisible by 4 **and** not evenly divisible by 100, **or**

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if `y` is. . .

- evenly divisible by 4 **and** not evenly divisible by 100, **or**
- evenly divisible by 4 **and** is evenly divisible by 400

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if...

- `y % 4 == 0` **and** `y % 100 != 0`, **or**
- evenly divisible by 4 **and** is evenly divisible by 400

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if. . .

- `y % 4 == 0` **and** `y % 100 != 0`, **or**
- `y % 4 == 0` **and** `y % 400 == 0`

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if...

- `y % 4 == 0 && y % 100 != 0` ||
- `y % 4 == 0 && y % 400 == 0`

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if...

```
y%4==0 && y%100!=0 || y%4==0 && y%400==0
```

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if...

```
y%4==0 && y%100!=0 || y%4==0 && y%400==0
```

Approach to LeapYear

A year, y, is considered a *leap* year if. . .

```
y%4==0 && y%100!=0 || y%4==0 && y%400==0
```

A && **B** || **A** && **C**

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if...

`y%4==0 && y%100!=0 || y%4==0 && y%400==0`

A && **B** || **A** && **C**

$$\mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}$$

Approach to `LeapYear`

A year, `y`, is considered a *leap* year if...

```
y%4==0 && y%100!=0 || y%4==0 && y%400==0
```

A && **B** || **A** && **C**

$$\mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}$$

$$\mathbf{A} \cdot (\mathbf{B} + \mathbf{C})$$

Approach to LeapYear

A year, `y`, is considered a *leap* year if...

`y%4==0 && y%100!=0 || y%4==0 && y%400==0`

A && **B** || **A** && **C**

$$\mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}$$

$$\mathbf{A} \cdot (\mathbf{B} + \mathbf{C})$$

A && (**B** || **C**)

Approach to LeapYear

A year, y, is considered a *leap* year if...

`y%4==0 && y%100!=0 || y%4==0 && y%400==0`

A && **B** || **A** && **C**

$$\mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}$$

$$\mathbf{A} \cdot (\mathbf{B} + \mathbf{C})$$

A && (**B** || **C**)

`y%4==0 && (y%100!=0 || y%400==0)`

LeapYear Solution

One way:

```
public static boolean isLeapYear(int y) {
    if ( y%4==0 && (y%100!=0 || y%400==0) ) {
        return true;
    } else {
        return false;
    }
}
```

LeapYear Solution

Another, equivalent way:

```
public static boolean isLeapYear(int y) {
    if ( y%4==0 && (y%100!=0 || y%400==0) ) {
        return true;
    }
    return false;
}
```

LeapYear Solution

Slickest way:

```
public static boolean isLeapYear(int y) {
    return y%4==0 && (y%100!=0 || y%400==0);
}
```

Attacking Ch. 6, #17

- Understand the problem!

Attacking Ch. 6, #17

- Understand the problem!
- Come up with some test cases — quantities of each book and the total cost per the problem's specification. Add these cases as comments inside a separate tester class, `BookOrderTester`. (Minimum of 4 test cases!)

Attacking Ch. 6, #17

- Understand the problem!
- Come up with some test cases — quantities of each book and the total cost per the problem's specification. Add these cases as comments inside a separate tester class, `BookOrderTester`. (Minimum of 4 test cases!)
- Write a class — `BookOrder` — containing the method `getOrderTotal()`.

Attacking Ch. 6, #17

- Understand the problem!
- Come up with some test cases — quantities of each book and the total cost per the problem's specification. Add these cases as comments inside a separate tester class, `BookOrderTester`. (Minimum of 4 test cases!)
- Write a class — `BookOrder` — containing the method `getOrderTotal()`.
- Implement the tester class.

Attacking Ch. 6, #17

- Understand the problem!
- Come up with some test cases — quantities of each book and the total cost per the problem's specification. Add these cases as comments inside a separate tester class, `BookOrderTester`. (Minimum of 4 test cases!)
- Write a class — `BookOrder` — containing the method `getOrderTotal()`.
- Implement the tester class.
- Run the tester class, comparing the return value of `getOrderTotal()` to the values you predetermined; If those values are equal, print the test case and that the method returned the correct amount.

Attacking Ch. 6, #17

- Understand the problem!
- Come up with some test cases — quantities of each book and the total cost per the problem's specification. Add these cases as comments inside a separate tester class, `BookOrderTester`. (Minimum of 4 test cases!)
- Write a class — `BookOrder` — containing the method `getOrderTotal()`.
- Implement the tester class.
- Run the tester class, comparing the return value of `getOrderTotal()` to the values you predetermined; If those values are equal, print the test case and that the method returned the correct amount.
- `main()` in the `BookOrder` class will address part (b), which polls user for keyboard input.

HW

- Finish today's in-class problem, Litvin Ch. 6, #17.
- §4 of PS #6 — Programming Exercises

Coming soon: In-class *Craps* lab!