Class Design, Conditionals, Loops: Putting it all Together

Lecture 19

Borrowing from slides by Alan Hu, Kurt Eiselt, Paul Carter, and Tamara Munzner

News
- Assignment 2 is out
  - Due next Friday, Oct 30
    (a week from today)
- Midterm 2 coming up the week after
  - Wednesday, Nov 4, 6:30-7:30
  - Note: the rooms will be different
- Material: conditionals, loops
  - Arrays will NOT be on the midterm after all!
  - Arrays are next topic (starting Monday)

Reading Assignments
- Reading for this week: looping/iterations
  - Edition 3: Ch. 6.1-6.5
  - Edition 2: Ch. 7.1-7.5
- Reading for next week: arrays
  - Edition 3: Ch. 7.1, 7.5-7.7
  - Edition 2: Ch. 8.1, 8.5-8.7

Objectives for Today
- Gain more practice in class design using loops and conditionals

Recap: Monte Hall Problem
- From old TV game show “Let's Make a Deal”...
- There are three doors, labeled 1, 2, and 3.
- Behind one door is a new car.
- Behind the other two are goats.
- You pick a door.
- Monte Hall opens another door and shows you a goat. He asks you if you want to keep your current pick, or switch to the other unopened door.
- You win what’s behind the door you pick.
- Should you switch doors?

Recap: To Switch Or Not To Switch
- Does it make a difference?
- Try playing the game a million times, either switching each time, or not.
- Better yet, have the computer simulate playing a million times!
Recap: Monte Hall Game Experiment
- What will this program look like?
- main method will loop 1,000,000 times
  - Each iteration, create a new game instance
  - Play the game.
    - Switch or not, depending on which test.
    - If we won, count it.
- Print the result.

Monte Hall Game Class Design
- What will this program look like?
- main method will loop 1,000,000 times
  - Each iteration, create a new game instance
  - Play the game.
    - Switch or not, depending on which test.
    - If we won, count it.
- Print the result.

Generalized Monte Hall
- What happens if we have 4 doors?  5 doors?  n doors?
- Should we switch?

One More Example:
Loan Amortization
- How do we compute the right payment amount to pay off a loan after some number of years?

We Have Seen This Code Before
- How about computing your account balance after some number of years?
  double balance = initBalance;
  int years = 0;
  while (years < term) {
    intAccrued = balance*intRate/100;
    balance += (intAccrued - payment);
    years++;
  }

Loan Amortization
- How do we compute the right payment amount to pay off a loan after some number of years?
- Use “Nested Intervals” approach:
  - Have two guesses: tooLow and tooHigh
  - Try a guess halfway in between.
  - Compute the loan balance using guess.
    - If guess was too high, then tooHigh = guess
    - else tooLow = guess.
  - Repeat
Loan Amortization

- How do we know that our program will terminate?
- How do we know that our program will compute the correct result?

Inner Loop:
years starts at 0 and counts up to term. Will always execute exactly term times.

Outer Loop: The gap between tooLow and tooHigh gets cut in half each iteration.

These are termination arguments (aka ranking functions). You should always know why your loops will terminate.

Loan Amortization

How do we know that our program will compute the correct result?

Inner Loop: At each iteration, balance is always the correct value after years years.

Outer Loop: At each iteration, tooLow is always less than the correct value, and tooHigh is always greater than the correct value.

These are called “loop invariants”. Very helpful to understand loops. (Great to put in comments!)