CompSci 6
Programming Design and Analysis

September 3, 2009

Prof. Rodger and Prof. Forbes

\[
\begin{align*}
  t_0 &= 2.0 \\
  t_1 &= \frac{1}{2}(t_0 + \frac{3}{t_0}) = 1.5 \\
  t_2 &= \frac{1}{2}(t_1 + \frac{3}{t_1}) = 1.4166666666666665 \\
  t_3 &= \frac{1}{2}(t_2 + \frac{3}{t_2}) = 1.4142156862745097 \\
  t_4 &= \frac{1}{2}(t_3 + \frac{3}{t_3}) = 1.4142135623746899 \\
  t_5 &= \frac{1}{2}(t_4 + \frac{3}{t_4}) = 1.414213562373095
\end{align*}
\]
Announcements

• Reading for next time
  – Chap. 4.1-4.5, Chap 5, Chap 7.1-7.5
  – Reading Quiz due before next class
Top 10 list for surviving in CompSci 6

• 10. Read the Book
• 9. Keep Pizza number handy
• 8. Learn how to spell Rodger
• 7. Ask questions
• 6. Keep working until it is correct
Top 10 list (cont)

• 5. Get the easy points! (reading quizzes, READMEs, etc.)
• 4. Visit your professor, TA and/or UTA
• 3. Read the CompSci 6 Bulletin Board
• 2. Seek help when stuck (1 hour rule)!
• 1. Start programming assignments early!
Estimation

• Square Root:
  – Given a real number $c$ and some error tolerance $\epsilon$
  – Estimate $t$, the square root of $c$

• $\pi$:
  – Estimate $\pi$ with a given number of Monte Carlo trials
While Loops: Square Root

Q. How might we implement `Math.sqrt()`?

A. To compute the square root of `c`:
   - Initialize `t_0 = c`.
   - Repeat until `t_i = c / t_i`, up to desired precision:
     set `t_{i+1}` to be the average of `t_i` and `c / t_i`.

<table>
<thead>
<tr>
<th><code>t_i</code></th>
<th>=</th>
<th><code>c / t_i</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>t_0</code></td>
<td>=</td>
<td>2.0</td>
</tr>
<tr>
<td><code>t_1</code></td>
<td>=</td>
<td>1.5</td>
</tr>
<tr>
<td><code>t_2</code></td>
<td>=</td>
<td>1.4166666666666665</td>
</tr>
<tr>
<td><code>t_3</code></td>
<td>=</td>
<td>1.4142156862745097</td>
</tr>
<tr>
<td><code>t_4</code></td>
<td>=</td>
<td>1.4142135623746899</td>
</tr>
<tr>
<td><code>t_5</code></td>
<td>=</td>
<td>1.414213562373095</td>
</tr>
</tbody>
</table>

computing the square root of 2
Newton-Raphson Method

- Square root method explained.
  - Goal: find root of function $f(x)$.
  - Start with estimate $t_0 = c$.
  - Draw line tangent to curve at $x = t_i$.
  - Set $t_{i+1}$ to be x-coordinate where line hits x-axis.
  - Repeat until desired precision.

$$f(x) = x^2 - c$$ to compute $\sqrt{c}$
Buffon Needle Experiment

Figure 3 The Buffon Needle Experiment
Needle Position

• Needle length = 1, distance between lines = 2
• Generate random \( y_{low} \) between 0 and 2
• Generate random angle \( \alpha \) between 0 and 180 degrees
• \( y_{high} = y_{low} + \sin(\alpha) \)
• Hit if \( y_{high} \geq 2 \)

Figure 4
When Does the Needle Fall on a Line?
Constructing objects/Applying methods

• Class Rectangle in Chapter 2
• Creating a Rectangle object with x, y, width, and height
  Rectangle box = new Rectangle(5, 10, 20, 30);
• Applying Methods
  box.translate(15, 25);       // move the rectangle
  System.out.println("x: ", box.getX());   // print x
  System.out.println("y: ", box.getY());   // print y
Parts of a Class

• State
  – Data
• Constructors
  – Initialize state when object is created
• Accessor methods
  – Accessing data
• Mutator methods
  – Modify data – change the state
Class Example

- **Needle class** – Needle.java
  - Defines state and behavior of Needle
  - Keeps track of the number of times needle hits the line
  - Use drop() method to simulate dropping needle

- **java.util.Random class** in Java library
  - nextDouble() generates pseudo-random numbers in [0,1]
import java.util.Random;

/**
 * This class simulates a needle in the Buffon needle experiment.
 */

public class Needle
{
    /**
     * Constructs a needle.
     */
    public Needle()
    {
        hits = 0;
        tries = 0;
        generator = new Random();
    }

    /**
     * Drops the needle on the grid of lines and remembers whether the needle hit a line.
     */

public void drop()
{
    double ylow = 2 * generator.nextDouble();
    double angle = 180 * generator.nextDouble();

    // Computes high point of needle
    double yhigh = ylow + Math.sin(Math.toRadians(angle));
    if (yhigh >= 2) myHits++;
    tries++;
}

/**
 * Gets the number of times the needle hit a line.
 * @return the hit count
 */
public int getHits()
{
    return myHits;
}
/**
 * Gets the total number of times the needle was dropped.
 * @return the try count
 */
public int getTries()
{
    return myTries;
}

private Random myGenerator;
private int myHits;
private int myTries;

Intended Output:
Tries = 10000, Tries / Hits = 3.08928
Tries = 1000000, Tries / Hits = 3.14204

Big Java by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved
Classwork Today – Loops/Classes

• Snarf the `classwork/04_loops_cps006_fall09` project
• Complete Sqrt
  – Finish `estimate` method
  – Print results
• Complete Needle
  – Finish `main` method
  – Print results
• Classwork handout has all the details
• Submit under assignment name `classSep03`