Estimation

- **Square Root:**
  - Given a real number $c$ and some error tolerance $\epsilon$
  - Estimate $t_i$, the square root of $c$
- **$\pi$:**
  - Estimate $\pi$ with a given number of *Monte Carlo* trials

**Announcements**

- Reading for next time
  - Chap. 4.6, Chap 7.5, Chap 11.1
  - Reading Quiz due before next class
- Assignment 3 due tonight!
- Assignment 4 out.

**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>$t_i$</th>
<th>Computed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0 = \frac{c}{2}$</td>
<td>2.0</td>
</tr>
<tr>
<td>$t_1 = \frac{t_0 + \frac{c}{t_0}}{2}$</td>
<td>1.5</td>
</tr>
<tr>
<td>$t_2 = \frac{t_1 + \frac{c}{t_1}}{2}$</td>
<td>1.416666666666665</td>
</tr>
<tr>
<td>$t_3 = \frac{t_2 + \frac{c}{t_2}}{2}$</td>
<td>1.4142156862745097</td>
</tr>
<tr>
<td>$t_4 = \frac{t_3 + \frac{c}{t_3}}{2}$</td>
<td>1.4142135623746899</td>
</tr>
<tr>
<td>$t_5 = \frac{t_4 + \frac{c}{t_4}}{2}$</td>
<td>1.414213562373095</td>
</tr>
</tbody>
</table>

© Sedgewick & Wayne
Newton-Raphson Method

• Square root method explained. \( f(x) = x^2 - c \) to compute \( \sqrt{c} \)
  – 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.

Buffon Needle Experiment

Needle Position

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

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]

---

```java
import java.util.Random;

public class Needle {
    public Needle() {
        hits = 0;
        tries = 0;
        generator = new Random();
    }

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

        double yhigh = ylow + Math.sin(Math.toRadians(angle));
        if (yhigh >= 2) myHits++;
        tries++;
    }

    public int getHits() {
        return myHits;
    }
}
```

Continued
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

Classwork Today – Loops/Classes

• Snarf the classwork 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 Class07-Feb04