Instructor: Joel Castellanos
e-mail: joel@unm.edu
web: http://cs.unm.edu/~joel/
Course website: http://cs.unm.edu/~joel/cs259/

Upcoming Schedule

Today:
- Lab 10: Loops and Numerical Integration (due Thursday night).
- Homework 2 (due Friday): Carefully Read textbook Chapter 5: Inheritance, pages 171-185.
  Only read “C++ notes” if you know C++.

Wednesday: Retest

Friday: Quiz on Reading
How Far does Alex Drive

- Alex is driving in cruise control on Interstate 40.
- His speed remains constant at 75 miles / hour from 5:00 PM through 7:00 PM.
- How much distance does Alex drive from 5:27 PM to 5:57 PM?

\[ 75 \text{ mi/hr} \times 0.5 \text{ hr} = 36.5 \text{ miles} \]

Velocity Time Curve

At a given time, let there be a function that returns Alex’s instantaneous velocity. Time is plotted on the \( x \)-axis. Velocity = \( f(t) \) is plotted on the \( y \)-axis. The distance traveled from \( t_1 \) to \( t_2 \) is equals the area under the curve between \( t_1 \) and \( t_2 \).

From \( t=0 \) to \( t=2 \), Alex drives \( \frac{1}{2}(2)(8) = 8 \text{ meters} \).

From \( t=2 \) to \( t=10 \), Alex drives \( (8)(8) = 64 \text{ meters} \).

From \( t=10 \) to \( t=12 \), Alex drives \( \frac{1}{2}(2)(4) + 4(2) = 12 \text{ meters} \).
Upper and Lower Sum Approximations

Finding area under the velocity-time graph is tricky when the plot is not made of straight lines.

The **lower sum** is the sum of the area of rectangles with a corner touching the plot and never crossing **above** it.

The **upper sum** is the sum of the area of rectangles with a corner touching the plot and never crossing **below** it.

Lab 10: How Far Did Alex Drive?

Given a method that returns the instantaneous velocity:

Write a method that uses numerical integration to find either the upper or lower bound of the distance Alex travels from one given time and another given time using a given number of rectangular steps.

Handling of Negative Numbers:

- If Alex misses an exit, gets off the highway and drives back, his velocity in the backward direction would be negative.

- In this assignment, we want the total distance traveled, not the net distance. Therefore, even when the velocity is negative, the distance is still positive.
**Lab 10 API**

```java
public class MovingObject
{
    double startTime = 0; //start time for the object
    public MovingObject(double t0)
        { startTime = t0; }
    // Returns velocity of this object in meters/second at the given number of
    // seconds, t, after this.startTime.
    public double getInstantaneousVelocity(double t)
        { //Unknown method you must call. }

    public double findDistanceUpperorLowerBound(double time1, double time2, int steps, boolean upper)
        { //This is the part you write }

    public static void main(String[] args)
        { //Used to test the class. }
}
```

**Error Checking**

```java
throw new IllegalArgumentException(message);

public double findDistanceUpperorLowerBound(
    double time1, double time2, int steps, boolean upper)
    { //This is the part you write }

Input Values must be:
- steps >= 1
- time1 <= time2
- time1 >= this.startTime
```
Example: `getInstantaneousVelocity`

```java
public double getInstantaneousVelocity(double t) {
    return (t-startTime)*2.0;
    //return 5.0;
}
```

Example: Constant Velocity (steps=2)

```java
public double getInstantaneousVelocity(double t) {
    return 2.5;
}
```

```java
public static void main(...) {
    time1 = 0.0;
    time2 = 2.0;
    steps = 2;
    ...
    // Since the velocity is constant, the upper and lower sums are the same.
    deltaTime = 1.0
    areaSum = 2.5 + 2.5 = 5.0
```
Example: Constant Velocity (steps=4)

```java
public double getInstantaneousVelocity(double t) {
    return 2.5;
}
```

```java
public static void main(...) {
    ... 
    time1 = 0.0;
    time2 = 2.0;
    steps = 4;
    ... 
}
```

- Since the velocity is constant, the upper and lower sums are the same.
- deltaTime = 0.5
- areaSum = 0.5×2.5 + 0.5×2.5 + 0.5×2.5 + 0.5×2.5 = 5.0

Example: Linear Velocity (steps=2)

```java
public double getInstantaneousVelocity(double t) {
    return 2.0*t;
}
```

```java
public static void main(...) {
    ... 
    time1 = 0.0;
    time2 = 2.0;
    steps = 2;
    ... 
}
```

- deltaTime = 1.0
- Lower Sum = 1.0×0.0 + 1.0×2.0 = 2.0
- Upper Sum = 1.0×2.0 + 1.0×4.0 = 6.0
- Exact Answer:
  triangle area = ½ bh = ½(2)(4) = 4
Example: Linear Velocity (steps=4)

```java
public double getInstantaneousVelocity(double t)
{ return 2.0*t;
}
```

```java
public static void main(...
{ ... 
    time1 = 0.0; 
    time2 = 2.0; 
    steps = 4; 
    ... 
    deltaTime = 0.5 
    Lower Sum = 0.5×0.0 + 0.5×1.0 + 0.5×2.0 + 0.5×3.0 = 3.0 
    Upper Sum = 0.5×1.0 + 0.5×2.0 + 0.5×3.0 + 0.5×4.0 = 5.0
```

Example: Linear Velocity (steps=10)

```java
public double getInstantaneousVelocity(double t)
{ return 2.0*t;
}
```

```java
public static void main(...
{ ... 
    time1 = 0.0; time2 = 2.0; 
    steps = 10; 
    ... 
    deltaTime = 0.2 
    Lower Sum = 0.2×0.0 + 0.2×0.4 + 0.2×0.8 + 0.2×1.2 + 0.2×1.6 + 0.2×2.0 + 0.2×2.4 + 0.2×2.8 + 0.2×3.2 + 0.2×3.6 + 0.2×4.0 = 3.6 
    Upper Sum = 0.2×0.4 + 0.2×0.8 + 0.2×1.2 + 0.2×1.6 + 0.2×2.0 + 0.2×2.4 + 0.2×2.8 + 0.2×3.2 + 0.2×3.6 + 0.2×4.0 = 4.4
```
Example: Quadratic Velocity

```java
public double getInstantaneousVelocity(double t) {
    return 3.0 * (t * t);  // v = 3t^2
}
```
time_1 = 2 seconds, time_2 = 4 seconds

Exact Solution:

\[
\int_{2}^{4} 3t^2 \, dt = \left. t^3 \right|_{2}^{4} = 64 - 8 = 56
\]

Numerical Results for steps=5:
upper=false, deltaTime=0.4: 4.80 + 6.91 + 9.41 + 12.29 + 15.55 = 48.96
upper=true, deltaTime=0.4: 6.91 + 9.41 + 12.29 + 15.55 + 19.20 = 63.36
t1=2.0, t2=4.0, steps=5: >48.96 <63.36

Algorithm

1. check for errors
2. deltaTime = (time2-time1)/steps
3. Loop i=0 to the number of steps
4. { speed1 = abs(getInstantaneousVelocity(t))
5.    speed2 = abs(getInstantaneousVelocity(t+deltaTime))
6.    speed = either max or min of speed1 and speed2 depending on value of upper.
7.    areaOfRectangle = speed*deltaTime
8.    t=t+deltaTime
}
9. return sum of all area of rectangles
Coding `getInstantaneousVelocity()`

How can this velocity-time graph be coded?

```java
public double getInstantaneousVelocity(double t)
{
    if (t<2.0) return t*4.0;
    if (t<10.0) return 8.0;
    if (t<12.0) return 8.0 - (t-10.0)*2.0;
    return 4.0;
}
```

Trapezoid Method

Calculate the lower rectangle, then add the area of the right triangle between the lower and upper sum.

Triangle area:

\[
\frac{1}{2} bh = \frac{1}{2}(\text{deltaTime})h
\]

\[
h = \max(v(t_1),v(t_2)) - \min(v(t_1),v(t_2))
\]

\[
= 0.5(0.0) + \frac{1}{2}(0.5)(1) + \\
0.5(1.0) + \frac{1}{2}(0.5)(1) + \\
0.5(2.0) + \frac{1}{2}(0.5)(1) + \\
0.5(3.0) + \frac{1}{2}(0.5)(1) = 4.0
\]