The Plan

- Basic principles of design
- Motivate why MVC is needed
- Model/View/Controller (MVC) overview
- Model (today)
- View (when we get to Graphical User Interfaces)
- Controller (when we get to Event Handling)

Basic Principles of Design

We want our programs to be

- simple
- functional
- fast (to code and to execute)
- correct
- easy to modify
- easy to extend
- capable of reuse
Why Simple?

- Simple is easy to understand and therefore easier to reuse, modify, and extend.
- Simple has less likelihood for program errors and is therefore more likely to be correct.
- Simple may be faster than complex, certain to code and perhaps in execution time.

Why functional?

- It has to work.
- How well it works is negotiable considering:
  - speed to release
  - cost to release
  - lifetime of code

Why fast?

- Who wants to wait?
- Works better on slower, more out-of-date machines.
- Sometimes crucial to application’s purpose
  - air control
  - nuclear plant facilities
  - weather prediction
  - stock prediction

Why correct?

- In some cases, incorrect execution is unacceptable
  - access control
  - online sales
- In some cases, incorrect execution can be remedied by re-execution
  - operating system locks up, so just reboot
  - database goes down, just restart
- impacts time coding
Why easy to modify?
- Users rarely can tell you exactly what they want
- Even if they could tell you, what users want changes
- Changes in hardware and software mandate code modification (think Y2K)

Why easy to extend?
- Why not make a good thing better?
- Enables multiple releases of functional programs
  - Windows 3.1, 95, 98, 2000, NT, etc.
  - Java 1.0, 1.1, 1.2, 1.3, 1.4 and now 1.5 (5.0)
- Keep up with increasing competition and demand

Capable of Reuse
- No need to reinvent the wheel
- Easier to build with tools than starting from scratch
- C was used to make C++ compiler, C++ used to make Java, Java used to make Java compiler!
- Reduce, reuse, recycle is good for coding too!

Why we need MVC?
MVC assists in achieving design principles:
- simple idea
- fast to code with a clear ready-made design
- structure makes reuse, modification and extension simple
- MVC design easy to test for correctness
What is MVC?

- **Model**
  - representation of the data
  - has no visual component

- **View**
  - visual display of the data

- **Controller**
  - specifies ways in which the model can be changed
  - between model and view

Simplified 1D Model

- Recall collision detection simulation.
- We’re going to apply MVC to solve the problem more generally.
- Were going to use Object Oriented Programming
  - what are the objects?
  - what value do they have?
  - how do the values change?
  - how are the values accessible?

What are our objects?

We’re going to model 1D particles and segments.

- **Particle** (similar to the Sprite class)
- **Segment** (similar to extending the Sprite class)
- **Tracker1D** (similar to Tracker)
- **VelocityTracker** (similar to ProjectileTracker)
- **ModelTest** (similar to our JFrame)

class Particle

Attributes
- double position
- Tracker1D

Behavior
- double getPosition()
- void setPosition(double)
- Tracker1D getTracker()
- void setTracker(Tracker1D)
class Segment extends Particle

Attributes
- double position
- double width
- Tracker1D

Behavior
- double getPosition()
- void setPosition(double)
- double getWidth()
- void setWidth(double)
- Tracker1D getTracker()
- void setTracker(Tracker1D)

interface Tracker1D

Attributes
- none

Behavior
- double getPosition()
- void advanceTime(double)

class VelocityTracker

Attributes
- double velocity
- double position

Behavior
- double getPosition()
- void advanceTime(double)

class ModelTest

Attributes
- none

Behavior
- public static void main(String[] args)
More to come....