Project 1 Demos

- Before you can demo your project for me:
  - UML and Cover Page must be turned in
  - Code stubs and javadocs must be submitted to D2L
- When you do give a demo, no more submissions for the Design stage will be accepted after that time

Project 1 Demos cont.

- I have a zip drive with different sets of valid instructions
- One will be chosen at random and placed in the proper directory so your code and open and read it
- Your program will be told to run “ALL” and a specific instruction set

Project 1 Demos cont.

- Your group has until 5 pm on the 24th to have a correct demo run
- If you do not pass a demo the first time, you can come back after figuring out what went wrong
- You have 4 tries to get a successful demo, and each run will be with a different set

Lab 3 Objectives

1. Analyze the class structure of an existing java program using UML diagrams,
2. Extract and store sensor data from the Finch,
3. Employ abstract classes to provide generic programming functionality, and
4. Search the Finch “data streams” for key values

Sensor Samples

- We will take a sample of data at regular (50 ms) intervals for about 5 seconds (so, 100 samples total).
- Each sample is a tuple that contains the values from the light, acceleration, obstacle and temperature sensors.

Queries

Goal: find and report the minimum, maximum and median data sample

- One way to do this: sort the samples and then take the first, last and middle samples

- How do we sort the samples?

Queries

How do we sort the samples?

- Use the sort method from the book
- But: we need some way of telling sort() to use a particular class variable (such as the Y component of the accelerometer)

One Solution…

Answer: define a new interface Comparable2

- Requires that the following is provided by the implementing class:

```
public int compareTo2(Object obj, VariableType var);
```

- VariableType is an enumerated type that tells the implementation of compareTo2() which class variable to compare

Enumerated Types

- An enumerated type variable is a means of storing one of several values
- Values are typically symbolic:
  - TRUE and FALSE
  - TEMPERATURE, LIGHT_LEFT, etc.
- Values are often non-ordered
  - The “equals” operator is meaningful
  - Greater-than and less-than are not meaningful

Provided Classes and Interfaces

- VariableType: a generic interface for specifying which variable (or “key”) to do the comparison on
- SensorType: a specific interface that provides an enumeration of the different sensor channels
- FinchSensor: a class that stores a single sensor sample
- Comparable2: an interface similar to Comparable
- sensorDriver: the top level program

General “To Do”

- Get the Finch talking to your computers
- Download Lab3.zip
- Analyze program (and draw the UML diagram)
- Provide implementation for compareTo2()
- Perform quick experiments

(try to get through all of these by the end of lab)