Introduction

The New CSE 102

(actually section 3 of CSE 132)

Instructional Staff

- Instructors –

Roger Chamberlain Ed Richter Ron Cytron

  - Bryan 509 Bryan 201 Bryan 525
  - roger@wustl.edu erichter@wustl cytron@wustl
- TA – Ben Stolovitz
  - Email: bstolovitz@wustl.edu
- Office hours: MW 4-5pm for Roger, others TBD
- Other appointments for Roger: see Jayme in dept. office (Bryan 509)

Course Web Page

- cse.wustl.edu/~roger/102
- Will contain calendar
- Will contain studio and lab assignments
- Documents grading, collaboration, and late policies
- Contains documentation on languages (Java, C) and tools (Eclipse, Subversion, Arduino)

What is this class about?

- Organization will be like CSE 131, 132
  - 1.5 hrs/wk lecture
  - 1.5 hrs/wk studio
  - 1.5 hrs/wk lab
- The material includes
  - Basic computer capabilities (I/O, esp. custom I/O)
  - Demystifying how computer systems operate
  - More than one machine, more than one type of machine
  - Design decisions that include both software and hardware

Some High-level Goals for CSE 102

- Introduce students to CoE concepts (so those who should be CoE students know what that is)
  - Do this while ensuring relevance to CS students
- Introduce students to concept that not all computers are desktop/laptop class machines
  - Computing happens in many different form factors
  - Vehicle for 102 will be an 8-bit microcontroller + standard desktop environment (Java/Eclipse from CSE 131)
- Introduce students to distributed concurrency
- Recurring theme throughout semester will be the representation of information

Typical Module Sequence

- Lecture
  - Here in Lopata 509
- Studio
  - Studio exercises in Urbauer 115 lab (attendance is required!)
- Lab
  - Lab demos in Urbauer 115
- Skills Lecture (less regular)
  - Here in Lopata 509
  - You can influence topics chosen (in studio Friday)

Early Topics

- Mod. 1 – platform intro, timing, digital output
- Mod. 2 – digital input, analog input (measuring properties of the physical world)
- Mod. 3 and 4 – information representation (numbers, characters, strings, images, etc.)
- Mod. 5 and 6 – fetch-execute cycle (how does a processor really work?)
- Following – distributed computing

Two Compute Platforms

- Java on laptop or lab machines, using Eclipse as the development environment (just like 131)
- ``C'' on Arduino machine
  - Actually a subset of C, and subset is *very* close to the Java you are familiar with
  - Physical computer is 8-bit machine running at only 16 MHz (over 100 times slower than desktop PC)
    - 16 Kbytes of program memory
    - 2 Kbytes of data memory
    - No keyboard or display
  - Wonderful community of users, doing lots and lots!

Timing

- Passage of time (in Java)
- Use Thread.sleep()
  - Argument is integer number of milliseconds before the method returns

```
for (int i=0; i < endTime; i++) {
    Thread.sleep(1000);
    System.output.println(i + " seconds have elapsed");
}
```

Exceptions

- Deviations from the normal flow of control
- "Old style" error checking:

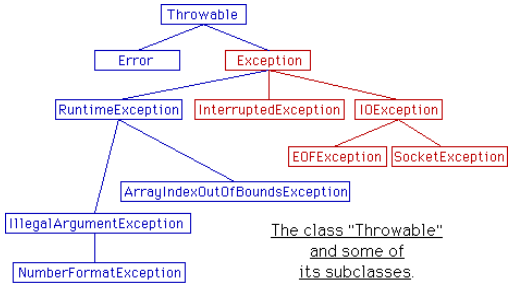
```
if (i < 0 || i >= A.length) {
        // handle out of range index
}
else {
        // access array element A[i]
}
```

- Exceptions allow us to be a bit more general

Try/Catch Block

```
try {
    // arbitrary code that might throw an
    // exception when something goes wrong
}
catch (Exception e) {
    // handle the thrown exception
}
```

Unchecked / Checked



Back to Java Timing

- The Thread.sleep() method can throw the "InterruptedException," so we enclose it in a try/catch block

```
for (int i=0; i < endTime; i++) {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        // default action
        e.printStackTrace();
    }
    System.output.println(i + " seconds have elapsed");
}
```

Arduino Programs

- Community calls them "sketches"
- Composed of the basic structure below

```
void setup() {
    // insert startup code here, will execute once
}

void loop() {
    // insert main code here, will execute over and over
}
```

Arduino Timing

- Use delay() library routine
  - Argument is integer number of milliseconds
- How do we print?
  - Serial.print() and Serial.println()
  - Argument can be any type
    - Serial.println("String to print");
    - Serial.print(14); // no newline included
    - NOTE: cannot do this – Serial.println("X = " + x); because string concatenation is not supported
    - Do this instead –

```
Serial.print("X = ");
Serial.print(x);
```

Better Timing

- What are possible issues with this code?

```
while (true)
    wait for 1 second
    output time information
end while
```

- How about using a free-running timer?
  - unsigned long millis()
  - Returns # of milliseconds since reset
  - Rolls over to zero after about 50 days

Hello World

- First complete Arduino program

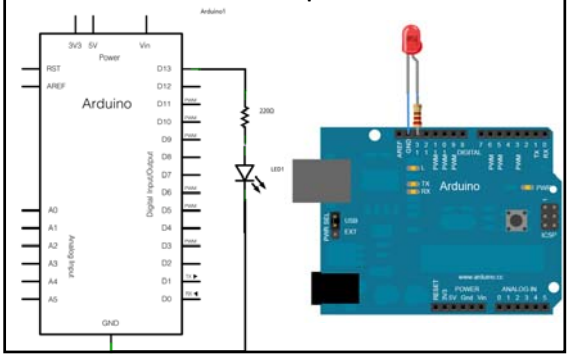
```
void setup() {
    Serial.begin(9600); //startup comm. link to PC
    Serial.println("Hello world!");
}
void loop() {
}
```

Arduino Input/Output

- 20 pins on physical chip can be configured to do digital input, digital output, analog input, analog output (not all pins can do each function)
- We must configure pins at startup, then use them

```
const int myPin = 13;
void setup() {
    pinMode(myPin, OUTPUT);
}
void loop() {                    //generates square wave
    digitalWrite(myPin, LOW);
    digitalWrite(myPin, HIGH);
}
```

Digital Output LED



Studio Friday

- Come to Urbauer 115
- Form groups of 2
- Do the exercises
  - Simple heartbeat on PC in Java and on Arduino
  - Metronome
  - Stoplight controller
- Get signed out by Ben, Ed, or Roger