

## CS61B: Administration

- Lab reader and Java reference manual available at Vick Copy, 1879 Euclid.
- Labs start immediately. Get an account (if needed) and register electronically *this week*
- Account forms available in labs.
- Class web page and newsgroup set up: read them regularly!
- Classes crowded: we're working on it. Go to any sections, labs where you fit.
- Enrollment problems, appeals: Michael-David Sasson, 379 Soda, *NOT US!*
- Concurrent enrollment: must wait for regular students (sorry).
- Reading for today's lecture: *On To Java*, chapters 2–7, 22–24 (they're short).
- Reading for Wednesday: chapters 8–11, 15.

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 1

## CS61B: Course Organization

- You read; we illustrate.
- Labs are important: practical dirty details go there.
- Homework is important, but really not graded: use it as you see fit and *turn it in!*
- Individual projects are *really* important! Expect to learn a lot.
- Tests are challenging: better to stay on top than to cram.
- Tests, 90%; Projects, 90%; HW, 20%
- Stressed? Tell us!
- Opportunity to decide.

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 2

## Programming, not Java

- Here, we learn *programming*, not Java (or Unix, or NT, or ...)
- Programming principles span many languages
  - Look for connections.
  - Syntax ( $x+y$  vs.  $(+ x y)$ ) is superficial.
  - E.g., Java and Scheme have a lot in common.
- Whether you use GUIs, text interfaces, embedded systems, important ideas are the same.

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 3

## Really simple example

```
public class Greet {  
  
    /** Print a greeting message  
     * on the standard output. */  
    public static void main (String[] args) {  
        System.out.print ("Hello, ");  
        if (args.length > 0)  
            System.out.println (args[0]);  
        else  
            System.out.println ();  
    }  
}
```

---

```
% javac -g Greet.java      # Creates Greet.class  
% java Greet world        # Interpreter calls Greet  
Hello, world              # Output  
% java Greet me warmly   # Another run  
Hello, me                  # args[0] = "me"  
%
```

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 4

## Lessons from Simple Example

- All definitions are inside some class.
- Syntax  $A.B$  means “the  $B$  defined inside  $A$ ,”
  - E.g., `System.out.println`, `Greet.main`
- Ordinary function is *static method*, like `Greet.main`.
- Methods declare what kind of arguments they take, and what kind of value they return (`void` means “no value”).
- Method calls use familiar prefix syntax.
- Command-line arguments become an *array of strings*.
- Array is indexed sequence: `args[0]`, `args[1]`, ..., `args[args.length-1]`
- Conditional statement: `if ...then ...else`.
- Access control: `public` and others.

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 5

## Prime Numbers

**Problem:** want java `PrintPrimes0 L U` to print prime numbers between  $L$  and  $U$ .

**Definition:** A *prime* number is an integer greater than 1 that has no divisors smaller than itself other than 1.

**Useful Facts:**

- If  $k \leq \sqrt{N}$ , then  $N/k \geq \sqrt{N}$ , for  $N, k > 0$ .
- $k$  divides  $N$  iff  $N/k$  divides  $N$ .

**So:** Try all potential divisors up to and including the square root.

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 6

## Plan

```
class PrintPrimes0 {
  /** Print all primes in the range given by ARGS
   * on the standard output (ARGS consists of
   * 1 or 2 numerals: an upper bound or lower
   * and upper bounds). */
  static void main (String[] args) {
    ...
  }

  /** True iff X is prime. */
  static boolean isPrime (int x) {
    ...
  }

  /** Print all primes in the range LOW .. HIGH,
   * inclusive, on the standard output. */
  static void print (int low, int high) {
    ...
  }
}
```

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 7

## Testing for primes

```
/** True iff X is prime. */
static boolean isPrime (int x) {
  return x == 2 ||
    (x > 1 &&
     ! isDivisible (x, 2, 1 + (int) Math.sqrt(x)));
}

/** True iff X is divisible by any integer in the
 * range L .. U. */
static boolean isDivisible (int x, int L, int U) {
  if (L > U)
    return false;
  else
    return x % L == 0 || isDivisible (x, L+1, U);
}

• boolean is true/false
• &&, ||, ! are 'and', 'or', 'not'.
• Could have written body as just
  return L <= U && (x % L == 0 || ...);
• % is remainder.
• (T) means "convert to T".
```

Last modified: Thu Sep 6 11:28:14 2001

CS61B: Lecture #1 8