Public-Service Announcements

- "CSUA has a Welcome BBQ on Wednesday, 2 September at 7PM in the Woz. Open to anyone interested in computer science. Please drop by our office located in 311 Soda Hall"

Administrivia

- Please make sure you have used our "Account Administration" link to obtain an account and to register it,
- If you did not complete Lab #1, please try to do so by tonight. It is especially important to set up your central repository
- If you decide not to take this course after all, please tell Tele-BEARS ASAP, so that we have a reasonably accurate count of class membership.

Lecture #2: Let's Write a Program: Prime Numbers

Problem: want java Primes \( U \) to print prime numbers through \( U \).

You type: java Primes 101

It types: 2 3 5 7 11 13 17 19 23 29

31 37 41 43 47 53 59 61 67 71

73 79 83 89 97 101

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

Useful Facts:

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

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

Plan

```java
public class Primes {
    /** Print all primes up to ARG[0] (interpreted as an integer), 10 to a line. */
    public static void main(String[] args) {
        printPrimes(Integer.parseInt(args[0]));
    }

    /** Print all primes up to and including LIMIT, 10 to a line. */
    private static void printPrimes(int limit) {
        /* For every integer, x, between 2 and LIMIT, print it if isPrime(x), 10 to a line. */
    }

    /** True iff X is prime */
    private static boolean isPrime(int x) {
        return /*( X is prime )*/;
    }
}
```
Testing for Primes

```java
private static boolean isPrime(int x) {
    if (x <= 1)
        return false;
    else
        return !isDivisible(x, 2); // "!" means "not"
}
```

/** True iff X is divisible by any positive number >=K and < X, * given K > 1. */
private static boolean isDivisible(int x, int k) {
    if (k >= x) // a "guard"
        return false;
    else if (x % k == 0) // "%" means "remainder"
        return true;
    else // if (k < x && x % k != 0)
        return isDivisible(x, k+1);
}

Lesson: Comments aid understanding. Make them count!

Thinking Recursively

Understand and check isDivisible(13,2) by tracing one level.

• Call assigns x=13, k=2
• Body has form 'if (k >= x) S1 else S2'.
• Since 2 < 13, we evaluate the first else.
• Check if 13 mod 2 = 0; it's not.
• Left with isDivisible(13,3).

• Rather than tracing it, instead use the comment:
  Since 13 is not divisible by any integer in the range 3..12 (and 3 > 1), isDivisible(13,3) must be false, and we're done!
• Sounds like that last step begs the question. Why doesn't it?

Iteration

• isDivisible is tail recursive, and so creates an iterative process.
• Traditional "Algol family" production languages have special syntax for iteration. Four equivalent versions of isDivisible:

```java
if (k >= x) // a "guard"
    return false;
else if (x % k == 0) // "%" means "remainder"
    return true;
else // if (k < x && x % k != 0)
    return isDivisible(x, k+1);
```

int k1 = k;
while (k1 < x) {
    if (x % k1 == 0)
        return true;
    k1 += 1;
}
return false;
```java
for (int k1 = k; k1 < x; k1 += 1) {
    if (x % k1 == 0)
        return true;
    }
return false;
```

Using Facts about Primes

• We haven't used the Useful Facts from an earlier slide. Only have to check for divisors up to the square root.
• So, reimplement isPrime:

```java
private static boolean isPrime(int x) {
    if (x <= 1)
        return false;
    else
        return !isDivisible(x, 2, (int) (Math.round(Math.sqrt(x) + 1.0))); // "(int) E" is "convert to int". Math.round => a 'long'.
}
```

```java
private static boolean isDivisible(int x, int k, int lim) {
    if (k >= lim) // a "guard"
        return false;
    else if (x % k == 0)
        return true;
    else // if (x % k != 0) // "%" means "remainder"
        return isDivisible(x, k+1);
}
```
Final Task: printPrimes

/** Print all primes up to and including LIMIT, 10 to
 * a line. */
private static void printPrimes(int limit) {

}

printPrimes: One solution

/** Print all primes up to and including LIMIT, 10 to
 * a line. */
private static void printPrimes(int limit) {
    int np;
    np = 0;
    for (int p = 2; p <= limit; p += 1) {
        if (isPrime(p)) {
            System.out.print(p + " ");
            np += 1;
            if (np % 10 == 0)
                System.out.println();
        }
    }
    if (np % 10 != 0)
        System.out.println();
}