class Primes {
    public static void main(String[] args) {
        // Create array and initialize all entries.
        boolean[] marks = new boolean[1000];
        for (int i = 0; i < marks.length; i = i+1)
            marks[i] = true;
        // For each number k >= 2
        for (int k = 2; k < marks.length; k = k+1)
            // If it is prime, cross out its multiples.
            if (marks[k])
                for (int j=2; j*k< marks.length; j =j+1)
                    marks[j*k] = false;
        // Display results
        System.out.println("Primes: ");
        for (int i = 0; i < marks.length; i = i+1)
            if (marks[i]) System.out.println(i);
    }
}
• This is a **class** structure with one **function** in it:

```java
class Primes {
    public static void main(String[] args) {
        // Create array and initialize all entries.
        boolean[] marks = new boolean[1000];
        for (int i = 0; i < marks.length; i = i+1)
            marks[i] = true;
        // For each number k >= 2
        for (int k = 2; k < marks.length; k = k+1)
            // If it is prime, cross out its multiples.
            if (marks[k])
                for (int j=2; j*k< marks.length; j =j+1)
                    marks[j*k] = false;
        // Display results
        System.out.println("Primes: ");
        for (int i = 0; i < marks.length; i = i+1)
            if (marks[i]) System.out.println(i);
    }
}
```
Functions I

- A function allows *commands to be grouped* for use once or many times.
- A function can *receive arguments* that give values to be worked on.
- A function can *return a result* to be used by the caller.
- A function can have its own *local variables*.

```java
class Example2 {
    ...
    public static void main(String[] args) {
        int m = factorial(7);
        ...
    }
    public static int factorial(int n) {
        int prod = 1;
        for (int i = 1; i <= n; i = i+1)
            prod = prod*i;
        return prod;
    }
}
```
Functions II

• In applications, a program begins by calling the function called “main” and terminates when “main” is finished.

• A function should have a clearly defined-purpose, *doing one thing well.*

• A function should be short, if possible. This might mean defining and calling other functions.

• A function should **not** do input or output, **unless** that is its *sole purpose.*
A function’s arguments must be declared, like variables, separated by commas.

The type of the value returned must be declared in the function header.

There is some boilerplate we will explain later.

```java
public static   int power(int base, int exponent) {
    int prod = 1;
    for (int i = 0; i < exponent; i = i + 1)
        prod = prod * base;
    return prod;
}
```
Functions and Arrays

- Functions may take arrays as arguments or return arrays as results.

```java
public static double average(double[] numbers) {
    int n = numbers.length;
    double total = 0.0;
    for (int i = 0; i < n; i = i + 1)
        total = total + numbers[i];
    return total/n;
}

public static int[] reverse(int[] numbers) {
    int n = numbers.length;
    int[] result = new int[n];
    for (int i = 0; i < n; i = i + 1)
        result[i] = numbers[n-1 - i];
    return result;
}
```
class Primes {
    public static boolean[] makeSieve(int n) {
        boolean[] marks = new boolean[n];
        for (int i = 0; i < n; i++) marks[i] = true;
        return marks;
    }

    public static void doCancel(boolean[] marks, int n) {
        if (!marks[n]) return;  // ! means “not”
        for (int k = 2*n; k < marks.length; k += n) marks[k]=false;
    }

    public static void printPrimes(boolean[] marks) {
        for (int i = 2; i < marks.length; i++)
            if (marks[i]) System.out.print(" "+i);
    }

    public static void main(String[] args) {
        boolean[] sieve = makeSieve(100);
        for (int i = 2; i < sieve.length; i++) doCancel(sieve, i);
        System.out.print("Primes:");
        printPrimes(sieve);
        System.out.println(".");
    }
}
Other Assignments

• There are some short forms of assignment:

\[
\begin{align*}
a &=+ b; & \quad & // \quad a = a + b \\
a &=* b; & \quad & // \quad a = a * b \\
a &=/ b; & \quad & // \quad a = a / b \\
\ldots \quad & \\
f(i++) & & // \quad t = i; \quad i = i+1; \quad f(t) \\
f(++)i) & & // \quad i = i+1; \quad f(i)
\end{align*}
\]
Side-Effects

• When a function
  – modifies one of its arguments
  – modifies a global variable
  – does input or output
that is called a "side-effect"

• Side-effects should be used sparingly, if at all, and in a very carefully thought out, well-specified manner.

• If a function has side effects, it should probably not return a value.

• A function that does not return a value must specify `void` as its return type.