

# Organizing Programs

- Consider the following issues:
  - As programs increase in complexity and size, how can we break them up into more manageable pieces?
  - How can we avoid duplicating code to perform the same action in various places in our programs?

# Methods

- One of the mechanism provided by Java to organize program structure is *static/class* methods
- Informally, a method is a sequence of statements that performs some task. These statements are grouped together and given a name (the name of the method)

## A Simple Example

- Let's consider a program that, after asking the user for two positive numbers, *width* and *height*, outputs a rectangle of '+'s of the given width and height.

## Anatomy of a Method

```
private static void procedureName(parameters)
{
    // sequence of statements
}

private static returnType functionName(parameters)
{
    // sequence of statements
    return value;
}
```

## A Method (Procedure)

```
private static void outputOneRow(int w)
{
    int column = 0;
    while (column < w)
    {
        System.out.print('+');
        column = column + 1;
    }
    System.out.println();
}
```

## Another Method (Function)

```
private static int inputWidth(Scanner in)
{
    System.out.print("Enter width > 0: ");
    int width = in.nextInt();
    return width;
}
```

# Parameters

- We need a mechanism to provide a method with the information it needs to perform its task, e.g.,
  - What will a method to compute the area of a rectangle need?
  - What will a method to print the average of two integers need?
  - What will a method to count the number of occurrences of a character in a string need?

# Parameters cont.

- In a method declaration, we specify the *formal parameter list*, which looks like a list of variable declarations separated by commas, e.g.,  
`int a, int b, double c, char d, String e`
- You can have 0 or more parameters for your methods and they can be of any data type
- Choose meaningful names for the formal parameters, just like you would for variables

## Examples of Method Headers

- `private static double area(  
                                  double width, double height)`
- `private static void printAverage(  
  int x, int y)`
- `private static int occurrences(  
  char ch, String str)`

## Examples of Method Calls

```
Scanner keyboard = new Scanner(System.in);  
double w = keyboard.nextDouble();  
double h = keyboard.nextDouble();  
double myArea = area(w, h);  
  
int i = 21, j = 13;  
printAverage(i, j);  
  
String s = "This couldn't be more fun!";  
int count = occurrences('u', s);
```

# What Happens?

```
import java.util.Scanner;
public class ComputeArea
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        double w = keyboard.nextDouble();
        double h = keyboard.nextDouble();
        double myArea = area(w, h);
        System.out.println(myArea);
    }
    private static double area(double width, double height)
    {
        double a = width * height;
        return a;
    }
}
CSE 201 – Elementary Computer Programming
```

91

# Procedures vs. Functions

- Procedures are declared with void; functions are declared with a return type
- Procedures perform an action; functions compute a value
- Procedures do not return a value; functions must return a value
- Procedure calls are statements; function calls are expressions

## Why Methods?

- Methods are needed for at least two different reasons:
  - They allow us to better structure and organize our programs by breaking up possibly large pieces of code into smaller, more manageable pieces
  - When the same code solving some task appears in multiple places in our program, we can write the code once in a method, and execute the code in multiple places simply through a method call

## Your Turn

- For each of the following tasks design an appropriate method header:
  - The task is to compute the integer average of three given integer numbers
  - The task is to output to the screen the information of a doctor's patient, given the name, age, weight (in pounds), and height (in feet) of the patient
  - Given the first name, middle initial, and last name of a person, the task is to concatenate them and return the result (e.g., "Earl E. Bird")

## Your Turn cont.

- Compute integer average of three integers
- Print information of patient, given name, age, weight (in pounds), height (in feet) of patient
- Concatenate first name, middle initial, and last name of a person, and return the result

## Your Turn, Still

- Design and implement a class method that inputs a sequence of non-negative real numbers from a given Scanner, and returns the average of the numbers read. The method stops reading numbers when a negative number is entered.

## Method: average

## Your Turn, Last Time

- Design and implement a class method that given a `String` and a character, returns the index of the last occurrence of the character in the string (or `-1` if the character does not occur in the string).

# Method: indexOfLast