CSE 1223: Introduction to Computer Programming in Java
Chapter 5 – Methods
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 a positive number *length*, outputs a square of ‘+’s with sides of the given length.
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    System.out.print("Enter length > 0: ");
    int length = Integer.parseInt(keyboard.nextLine());
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < length) {
        // Output one row of length '+'s
        int column = 0;
        while (column < length) {
            System.out.print('+');
            column = column + 1; // go to the next column
        }
        System.out.println(); // terminate the row
        row = row + 1; // go to the next row
    }
}
A Simple Example

- The code on the previous slide works, but we could make it better:
  - Instead of having all of our code in the main method, we can break it out into other methods
  - This allows us to more easily trace and reason about our code
    - Which in turn allows us to more easily debug our code!
  - It also allows us to more easily reuse our code in new applications
    - *Code reuse* is one of the key goals of good software design and engineering
Let’s walk through an example of how we can break our square printing code up using *methods*.

- The full example is available on the course website.
- You can download, compile and run it to see it work for yourself.
public static void outputOneRow(int length) {
    int column = 0;
    while (column < length) {
        System.out.print('+');
        column = column + 1; // go to the next column
    }
    System.out.println();
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    System.out.print("Enter length > 0: ");
    int len = Integer.parseInt(keyboard.nextLine());
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
}
public static void outputOneRow(int length) {
  int column = 0;
  while (column < length) {
    System.out.print('+');
    column = column + 1; // go to the next column
  }
  System.out.println();
}

public static void main(String[] args) {
  Scanner keyboard = new Scanner(System.in);
  // Input the length of a side
  System.out.print("Enter length > 0: ");
  int len = Integer.parseInt(keyboard.nextLine());
  // Output length rows of length columns of '+'s
  int row = 0;
  while (row < len) {
    outputOneRow(len);
    row = row + 1; // go to the next row
  }
}
public static void outputOneRow(int length) {
    int column = 0;
    while (column < length) {
        System.out.print('+');
        column = column + 1; // go to the next column
    }
    System.out.println();
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    System.out.print("Enter length > 0: ");
    int len = Integer.parseInt(keyboard.nextLine());
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
}

We call the method down here with the length we want it to print.
public static int promptForInteger(String msg, Scanner in) {
    System.out.print(msg);
    int value = Integer.parseInt(in.nextLine());
    return value;
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    int len = promptForInteger("Enter length > 0: ", keyboard);
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
}
public static int promptForInteger(String msg, Scanner in) {
    System.out.print(msg);
    int value = Integer.parseInt(in.nextLine());
    return value;
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    int len = promptForInteger("Enter length > 0: ", keyboard);
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
}
public static int promptForInteger(String msg, Scanner in) {
    System.out.print(msg);
    int value = Integer.parseInt(in.nextLine());
    return value;
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    int len = promptForInteger("Enter length > 0: ", keyboard);
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
}

We call it here, providing the message and the Scanner we want it to use.
public static int promptForInteger(String msg, Scanner in) {
    System.out.print(msg);
    int value = Integer.parseInt(in.nextLine());
    return value;
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    int len = promptForInteger("Enter length > 0: ", keyboard);
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
}
public static int promptForInteger(String msg, Scanner in) {
    System.out.print(msg);
    int value = Integer.parseInt(in.nextLine());
    return value;
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    // Input the length of a side
    int len = promptForInteger("Enter length > 0: ", keyboard);
    // Output length rows of length columns of '+'s
    int row = 0;
    while (row < len) {
        outputOneRow(len);
        row = row + 1; // go to the next row
    }
    We assign that returned value to the variable len.
    We can then use it like any other value – it is just another integer!
public static void outputSquare(int length) {
    int row = 0;
    // Output length rows of length columns of '+'s
    while (row < length) {
        outputOneRow(length);
        row = row + 1; // go to the next row
    }
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    int len = promptForInteger("Enter length > 0: ", keyboard);
    outputSquare(len);
}
public static void outputSquare(int length) {
    int row = 0;
    // Output length rows of length columns of '+'s
    while (row < length) {
        outputOneRow(length);
        row = row + 1; // go to the next row
    }
}

public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    int len = promptForInteger("Enter length > 0: ", keyboard);
    outputSquare(len);
}
Anatomy of a Method

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

26 statements executed by the method
Anatomy of a Method

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

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

return statement required in functions: returns a value of the returnType

statements executed by the method
A Method (Procedure)

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

  \[
  \text{int } a, \text{ int } b, \text{ double } c, \text{ char } d, \text{ String } e
  \]

- 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 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)

formal parameters
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);
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?

```java
import java.util.Scanner;

public class ComputeArea {

    public static void main(String[] args) {
        Scanner key = new Scanner(System.in);
        double w = Double.parseDouble(key.nextLine());
        double h = Double.parseDouble(key.nextLine());
        double myArea = area(w, h);
        System.out.println(myArea);
    }

    private static double area(double width, double height) {
        double a = width * height;
        return a;
    }
}
```
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.
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
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.
Your Turn, Again

- 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).
Your Turn, Once More

- Modify the code for printing a square to the screen so that it prints a rectangle instead.
  - You should prompt for both a *height* and a *width*
  - What do you need to change to make this work?