**Method**

- A *method* is a set of program statements that can be executed as a unit.
- The statements are executed by *invoking* the method.

**Why Write Methods?**

- Methods allow us to hide the details of a set of statements (this is an example of *abstraction*).
- Methods allow us to write a complicated set of statements once, and then execute them multiple times in our programs.
- EX: say we had a method that gets and returns a validated positive number; we could invoke it whenever our program needs to input a positive number.

**Overview**

- Every method must be defined within a class.
- Can be defined in a different class than the main method (EX: Input and Output classes).
- Invoke using the syntax: `ClassName.methodName(arguments)`
Single Class

- Can instead be defined in the *same* class with the **main** method
- Invoke in **main** using the syntax:
  - `methodName(arguments)`
  - Current class name is assumed

Method Parts

- Method Header
  - `public static void main (String [] args)`
  - `Output.showMessage("Hello world!");`
  - `} // method main`

- Method Body

Example 17.1

```
public class Simple {
    public static void methodA () {
        Output.showMessage("In methodA");
    } // method methodA

    public static void main (String [] args) {
        Output.showMessage("Before methodA");
        methodA();
        Output.showMessage("After methodA");
    } // method main

} // class Simple
```
Statements

• the method body can contain any valid statements
• EX:
  • variable declarations
  • while-loops
  • invocations of other methods
  • etc.

Local Variables

• variables declared within a method body are local variables
• local variables can only be used within the method where they are declared
• local variables cease to exist when the method they are declared in exits
• methods cannot “see” local variables declared in other methods

Example 17.2

```java
public class LocVar {
    public static void someMethod () {
        int num;
        num = 5;
    } // method someMethod
    public static void main (String [] args) {
        int num;
        num = 0;
        someMethod();
        IO.showValue("num is ", num);
    } // method main
} // class LocVar
```

Info In & Out

• when your program invokes a method it can pass data into, and get data out of, the method
  • pass data into the method via arguments
  • get data out of the method via its return value
Arguments

Arguments act exactly like local variables except that they are:

- declared in the argument list in the method header
- initialized with the corresponding values in the method invocation

Example 17.3

```java
public class Args {
    public static void adder (String label, int a, int b) {
        int sum;
        sum = a + b;
        Output.showValue(label, sum);
    } // method adder

    public static void main (String [] args) {
        int num = 5;
        adder("The total is: ", 10, num);
        Output.showMessage("After adder.");
    } // method main
} // class Args
```

Return Value

A method can return a single value to the invoking program.

- requires two things:
  - appropriate return type in the method header
  - a return statement in the method body

Return Statement

Syntax:

```
return expression;
```

- when executed, immediately exits the method
- returns the value of `expression` to the invoking program
- `expression` must have the same data type as the return type of the method
Example 17.4

```java
public class Return {
    public static double product (double x, double y) {
        double result;
        result = x * y;
        return result;
    } // method product
    public static void main (String [] args) {
        double p;
        p = product(6.5, 2.0);
        Output.showValue("p is ", p);
    } // method main
}
```

Example 17.5

**Problem:** output the secret ingredient of Tasti-Chips

**Abstract pseudocode:**
- confirm user agrees to terms
- display the secret ingredient
- confirm user agrees to terms

**Methods**

- each step could be very complex
- let’s create 2 methods named:
  - confirmAgreement
  - displaySecret
- these will only perform output
  - no arguments or return values needed

```java
public static void confirmAgreement () {
    String msg = "The secret ingredient is proprietary.\n";
    msg = msg + "By choosing OK, you agree:"
    msg = msg + "to hold this information confidential."
    Output.showMessage(msg);
} // method confirmAgreement
```
displaySecret

Method Header

public static void displaySecret ()
{
    String msg = "The secret ingredient in Tasti-Chips is extra MSG.";
    Output.showMessage(msg);
} // method displaySecret

Method Body

main

public static void main (String [] args)
{
    confirmAgreement();
    displaySecret();
    confirmAgreement();
} // method main

Another Method

let's make the user enter a pin number
create a method to verify the user's pin:
• verifyPin
takes as an argument the correct pin
returns whether or not the user entered the correct pin

verifyPin

public static boolean verifyPin (int goodPin)
{
    int pin;
    boolean isGood;
    pin = Input.readInt("Enter your pin: ");
    if (pin == goodPin)
        isGood = true;
    else // bad pin
        isGood = false;
    return isGood;
} // method verifyPin
public static void main (String [] args)
{
    int theRightPin = 4444;
    if (verifyPin(theRightPin))
    {
        confirmAgreement();
        displaySecret();
        confirmAgreement();
    } // if correct pin
    else
    Output.showMessage("INCORRECT PIN
" + "Sorry, you cannot see the secret.");
} // method main

Secret.java
* contains all methods (including main) in a single class
* take a moment to look at, compile, and experiment with this program

Library Class
* methods can be defined in a class without a main method, called a library class
  • these methods can then be used by any program
  • EX: Output, Input, and Math classes
* library classes that you write must be separately compiled
  • the compiled .class file must be in the same directory with any program that invokes any of its methods

Example 17.6
* the source code file Cylinder.java defines the methods given in the Cylinder class API
* the source code file CylTest.java contains a program that invokes Cylinder methods
Example 17.6 (cont.)

To run the example you must first:

- compile the class library (Cylinder.java)
- place the compiled file (Cylinder.class) in the same directory with the program source code file (CylTest.java)
- compile the program (CylTest.java)

Spend some time looking at and experimenting with this example.

Module 17 Vocabulary

- method
- local variable
- return statement

Questions?

Email Kevin at sahrk@sou.edu