**COP 2250 Java Programming I - Chapter 6 – Methods**

**What a Method is**

* A method is a named block of code with braces that enclose one or more Java statements.
* Methods organize and simplify programs and make code reusable.
* There are many methods in the Java API. Each one is defined in a Java class.
* However, you can also craft your own methods. That is the subject of this chapter.
* View Figure 6.1 on page 205. It illustrates the terminology for the parts of a method.
* Note: All methods in this chapter will have the **static** modifier.
* Some methods **return a value** as indicated by the return value type.
* Note that a method definition has:
  + a method **header** line with:
    - two modifiers, public and static
    - a return value type
    - a name
    - parentheses that optionally may contain a list of parameters
  + a **body** consisting of statements inside curly braces
* The value returned by a method should be “caught” so it can become part of the program.
* Methods that return a value always have a **return statement** to do the job.
* Some methods do not return a value. These methods are always of return-type **void**.
* Some methods require one or more **parameters** inside parentheses when they are defined.
* Parameters are values that the method needs to use internally to do its job.
* Methods can be called (aka invoked or executed) by other methods. For now we do this in main.
* When a method is called a value called an **argument** is passed to each parameter. This is explained well by Figure 6.2 on page 207.

Try TestMax

Try TestVoidMethod

Try TestReturnGradeMethod

* Although a void method doesn’t require a return statement, a naked return statement can be used to halt a void method. See page 210.

**Exercise**

Write a program named Adding.java that has a method named addTwo(). The addTwo( ) method should take two **double** parameters and return their sum. Test the addTwo() method with several calls in the main method. Don’t forget the static modifier in the method header and be sure to output the sum in main.

**Passing Arguments by Values**

* In the definition of a method, the parameters must be **typed**.
* When the method is called, values passed to the method are known as **arguments**.
* The arguments must match the parameters in **order**, **number** and **data type.**
* A parameter is a variable with its own location in memory, and when a method finishes, the parameter is cleared from memory.
* Parameters are said to be **passed-by-value** to the method. See page 212.

Try Increment

Try TestPassByValue

**Modularizing Code**

* Programs can often be improved by organizing statements into methods.
* This modularization also promotes **reuse** and eliminates **redundancy**.
* Using loops to call one or more methods in every loop cycle can greatly simplify your code.

Try GreatestCommonDivisorMethod

Try PrimeNumberMethod

Try Hex2Dec

**Overloading Methods**

* Overloading means that one class can have more than one method with the **same name**.
* Each overloaded method will have its own body statements.
* Overloading is permissible only if:
  + The **number** of parameters is different.
  + The **data types** of the parameters are different.
* NOTE: Using a different return type or modifier is NOT permitted for overloading a method because of the ambiguity. See page 221.

Try TestMethodOverloading

**Exercise**

Create a new Java class called **Addem.java** that has a method named **addNumbers( )**. The

addNumbers( ) function should take 2 integers as parameters and return the **sum** of the 2 numbers. This method should also report (to the screen) the two numbers being added. Catch and display the value returned by this function. Test it with some low numbers so you can easily check the result. When done, overload addNumbers( ) to accept **doubles** and **bytes**. Test both new methods.

**Variable Scope**

* The visibility of a variable in a program is called the **scope** of the variable.
* A variable declared inside a method is visible (accessible) only inside that same method. It has **method scope.** A parameter passed into a method also has method scope.
* A variable cannot be used until it has been declared (by name and type) and assigned a value.
* See Figure 6.6 on page 223.

Generating Random Characters (see page 223)

* Recall that char is a numeric type, so random characters can be generated with Math.random().

Study RandomCharacter on page 224. Then try TestRandomCharacter

The chapter ends with a look at abstraction and stepwise refinement.

Try PrintCalendar