**ADMINISTRATION**

Welcome to Week 2!

Homework 1 and Lab 1 – due next Wednesday (of Week 3)

Quiz 1 – next week Wednesday

**Operator Precedence**

See Lecture Notes for Document on Operator Precedence

**More Useful Links**

See Lecture Notes for Document on useful hyperlinks for JAVA!

**Some more on Data, Data types, Variable Naming Conventions and Declarations, Scanner Class**

JoptionPane – for Dialog Box -> JoptionPane.showMessageDialog()

Concatenation – use + operator to concatenate number with String (use println())

Relational Operators ==, >=, >, <=, <, != Boolean True or Boolean False

if(x==3) ---> ask: is x equal to 3?

if(x>=3) ---> ask: is x greater than OR equal to 3?

if(x>3) ---> ask: is x greater than to 3?

if(x<=3) ---> ask: is x less than OR equal to 3?

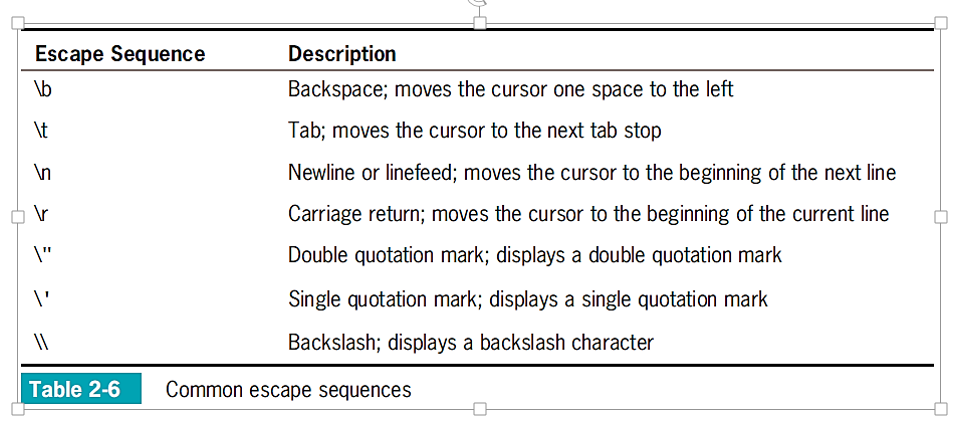
if(x<3) ---> ask: is x less than 3?

if(x!=3) ---> ask: is x not equal to 3?

Relational operators are usually used for selection control statements (e.g., the if statements)

Significant Digits – mathematical accuracy (and physical measurement accuracy)

Escape Sequences -



Input Dialog Box – allows user to input data when prompted

Confirm Dialog Box – allows user to input YES, NO, or CANCEL

Arithmetic Operators + - \* / %

Watch for Integer Division!

int n = 10

float r = 5.6

z = r/n

z = n/r

Automatic Type Conversion – converts nonconforming operands to unifying type

In order: double float long int

When in doubt for a datatype for numbers… use double

x = 4.5

metrics: time, correctness, efficiency

Efficient coding (clean) versus inefficient coding (works but not clean)

Lines of code

algorithms

**A Review of Methods in Java – Chapter Three**

• A method provides a way to encapsulate code in one place so that it can be reused. Its general form is

*Method Heading*

*Method Body*

where the heading has the general form

**modifiers returnType methodName(parameterDeclarations)**

and the body is a sequence of statements enclosed in curly braces ( { and } ):

**{**

**statements**

**}**

• Values ( called arguments ) can be passed into a method via parameters.

• Execution transfers from a method back to the caller when a return statement is encountered or the end of the method is reached.

• A return statement is not required for methods whose return type is void.

• A specification for a method should include descriptions of values it receives – parameters and their types; values input to the method; the value it returns; values it outputs; preconditions; and postconditions.

• A method’s documentation should include a comment that describes its specifications.

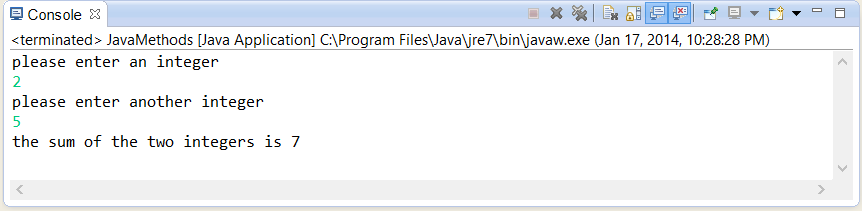
• Locals exist only while a method is executing and thus can be accessed only within the method. This means that other methods may reuse the name of a local for some other purpose without causing a conflict.

• When a method is called within the class in which it is defined, its name can be used without qualification.

• When a method is called, the arguments are associated with the parameters from left to right - the first argument with the first parameter, the second argument with the next parameter, and so on, until the matching is complete. There must be the same number of arguments as parameters and each argument’s type must be compatible with the type of the corresponding parameter.

**An Example of a void Method in Java**

**Sample Output**



**The Source Code**

|  |
| --- |
| **import java.util.Scanner; //package**  **public class JavaMethods**  **{**  **//declare a Scanner class object**  **static Scanner sc = new Scanner(System.in);**  **// define a method**  **public static void AddNumbers()**  **{**  **// the variables that are local to the method**  **int num1 = 0, num2 = 0, sum = 0;**    **//prompt user for a number**  **System.out.println("please enter an integer");**  **num1 = sc.nextInt();**    **//prompt user for another number**  **System.out.println("please enter another integer");**  **num2 = sc.nextInt();**    **// add the integers – processing step**  **sum = num1 + num2;**    **// output the sum of the two integers**  **System.out.println("the sum of the two integers is " + sum);**  **}**    **public static void main(String[] args)**  **{**  **// call the method**  **AddNumbers();**  **}**  **}** |

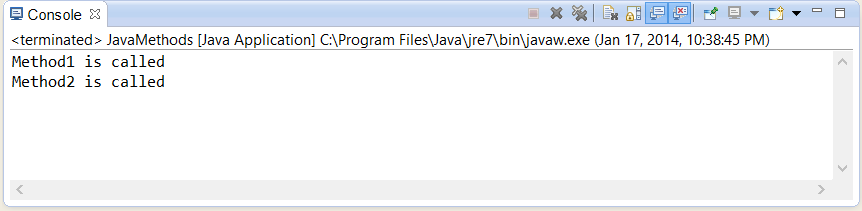
**The Source Code Dissected**

**For the above example, a void method is defined within the class. The method uses two local variables to read and assign the values of two integers.**

**The method is called in main() and then computes the sum of the integers.**

**An Example of a void Method in Java Calling Another void Method**

**Sample Output**



**The Source Code**

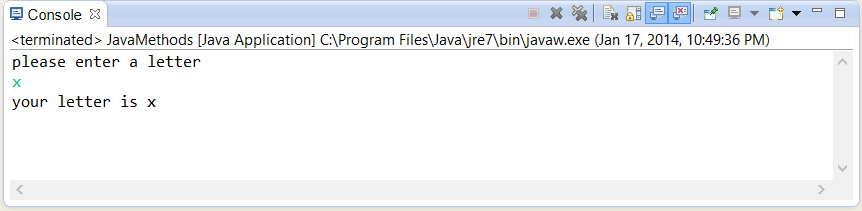
|  |
| --- |
| **public class JavaMethods**  **{**  **// define a method**  **public static void Method1()**  **{**  **// display a message to the program user**  **System.out.println("Method1 is called");**  **// call another method within this method**  **Method2();**  **}**  **// define another method**  **public static void Method2()**  **{**  **// display a message to the program user**  **System.out.println("Method2 is called");**  **}**  **public static void main(String[] args)**  **{**  **// call a method**  **Method1();**  **}**  **}** |

**The Source Code Dissected**

**For the above example, a two void methods are defined within the class. The methods merely display a message that they have been called. One method calls the other method. One of the methods is called in main().**

**An Example of a char Method in Java**

**Sample Output**



**The Source Code**

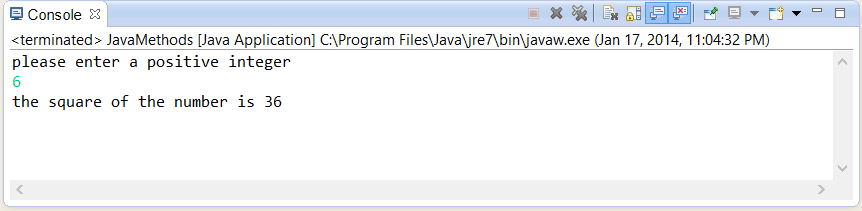
|  |
| --- |
| **import java.util.Scanner;**  **public class JavaMethods {**  **//declare a Scanner class object**  **static Scanner sc = new Scanner(System.in);**  **// define a method**  **public static char DisplayChar()**  **{**  **// declare a string and a character variable**  **String letter = "";**  **char myChar = '\0';**    **//request user for a character**  **System.out.println("please enter a letter");**  **letter = sc.next();**  **myChar = letter.charAt(0);**    **return myChar;**  **}**  **public static void main(String[] args)**  **{**  **// call the method**  **System.out.println("your letter is " + DisplayChar());**  **}**  **}** |

**The Source Code Dissected**

**For the above example, a char method is defined within the class. The method requests and receives a letter from the user. The method is called in main() via an output statement.**

**An Example of an int Method in Java that Accepts a Single Parameter**

**Sample Output**



**The Source Code**

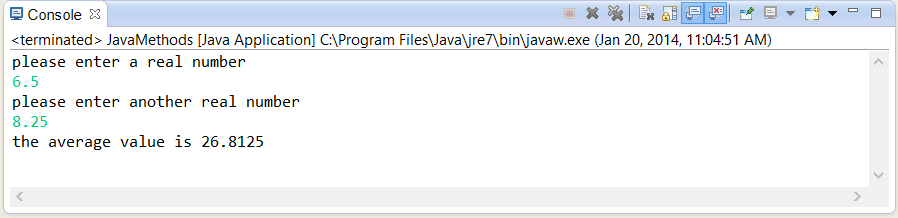
|  |
| --- |
| **import java.util.Scanner;**  **public class JavaMethods**  **{**  **//declare a Scanner class object**  **static Scanner sc = new Scanner(System.in);**  **// define a method**  **public static int SquareValue(int a)**  **{**  **return a \* a;**  **}**  **public static void main(String[] args)**  **{**  **// declare a local variable**  **int myNum = 0;**    **// request and receive an integer**  **System.out.println("please enter a positive integer");**  **myNum = sc.nextInt();**    **// call the method**  **System.out.println("the square of the number is " + SquareValue(myNum));**  **}**  **}** |

**The Source Code Dissected**

**For the above example, an int method is defined within the class. The method requests and receives an integer from the user. The method is called in main() via an output statement and returns the square of the given integer.**

**An Example of a double Method that Accepts Two double Parameters**

**Sample Output**



**The Source Code**

|  |
| --- |
| **import java.util.Scanner;**  **public class JavaMethods {**  **//declare a Scanner class object**  **static Scanner sc = new Scanner(System.in);**    **// define a method**  **public static double AverageValue(double a, double b)**  **{**  **return (a \* b) / 2.0;**  **}**    **public static void main(String[] args)**  **{**  **// declare local variable(s)..also initializing to zero**  **double num1 = 0, num2 = 0;**  **// request and receive two decimal values**  **System.out.println("please enter a real number");**  **num1 = sc.nextDouble();**  **System.out.println("please enter another real number");**  **num2 = sc.nextDouble();**    **// call the method**  **System.out.println("the average value is " +   AverageValue(num1, num2));**  **}**  **}** |

**The Source Code Dissected**

**For the above example, a double method is defined within the class. The method requests and receives two real numbers from the user. The method is called in main() via an output statement and returns the average value of the given decimal values.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Java Exercises for Methods**

Answer each of the following as True or False.

**(1)** **True** In the following, **buildVerse()** could be said to be the " caller. "

**buildVerse( "dog" , "bow" )**

**(2)** **True** If a method has the **return** type **void** , then a **return** statement is not required.

**(3)** **True** In the following, **yards** is a " parameter. "

**public static double yardsToFeet(double yards)**

**(4)** **True** The following is an example of a method " specification. "

**public static double yardsToFeet(double yards)**

**(5)** **False** Effects produced by a method are called " preconditions. "

**(6)** **True** " Driver " programs test methods.

**(7)** **False** The following method returns a value of type **double** .

**public static int yardsToFeet(double yards)**

**(8)** **True** Given the following method,

**public static int aMethod(int n)**

**{**

**return n \* 2;**

**}**

the statement **int anInteger = aMethod(2);**

would assign **anInteger** the value **4** .

**(9)** **False** Given the following method,

**public static int aMethod(int n)**

**{**

**return n \* 2;**

**}**

the statement **int aNum = aMethod( aMethod(4) );**

would assign **aNum** the value **8** .

**(10)** **True** A variable declared within a method cannot be accessed by the **main()** method.

Choose from (a) , (b) , (c) , (d) or (e) to answer the given problem.

**(11)** The first line of a method is called its " heading. " The rest of the method is called its \_\_\_\_\_\_\_\_\_\_ .

**(a) body** (b) constructor (c) parameter (d) argument

**(12)** A method’s body begins and ends with the characters \_\_\_\_\_\_\_\_\_\_ , respectively.

(a) [ ] (b) < > (c) ( ) **(d) { }** (e) ==

**(13)** Values that exist only while a method is executing are called \_\_\_\_\_\_\_\_\_\_ .

(a) drivers (b) static (c) preconditions **(d) locals**

**(14)** When a method is called, \_\_\_\_\_\_\_\_\_\_ are copied into the method’s \_\_\_\_\_\_\_\_\_\_ .

(a) parameters, arguments (b) arguments, body

(c) parameters, body **(d) arguments, parameters**

**(15)** Restrictions on method parameters that must be observed in order for a method to work correctly are called \_\_\_\_\_\_\_\_\_\_ .

(a) postconditions **(b) preconditions** (c) drivers (d) arguments

**(16)** A class method is defined using the keyword \_\_\_\_\_\_\_\_\_\_ .

(a) **void** (b) **final** (c) **class** **(d) static**

**(17)** A method whose return type is " **void** " returns \_\_\_\_\_\_\_\_\_\_ .

(a) **null** (b) **void**

(c) zero **(d) does not return a value**

**(18)** In the statement **double cost = price(itemNumber);**

**itemNumber** is called \_\_\_\_\_\_\_\_\_\_ in this method call.

(a) a parameter (b) a driver **(c) an argument** (d) an alias

**(19)** A method that can only be accessed within the class containing it must be specified with the \_\_\_\_\_\_\_\_\_ keyword.

(a) **void** (b) **final** (c) **private** **(d) static**

**(20)** A method’s " specification " often includes \_\_\_\_\_\_\_\_\_\_ .

(a) a precondition (b) a postcondition (c) a return value **(d)** All of these

**More Methods in Java**

Choose from (a) , (b) , (c) , (d) or (e) to answer the given problem.

**public static int minimum(int x, int y)**

**{**

**int smaller;**

**if(x < y)**

**smaller = x;**

**else**

**smaller = y;**

**return smaller;**

**}**

**(1)** What is the name of the method above?

(a) **x** (b) **smaller**

(c) **y** **(d) minimum**

**(2)** Based on the figure above, what would be the output of the following statement?

**int s = minimum(5, minimum(3, 7));**

**(a) 3** (b) **5**

(c) **7** (d) There would be no output as this is not a valid statement.

**(3)** Which of the following is NOT part of the heading of the method above?

(a) **public** **(b) int smaller;**

(c) **static** (d) **minimum(int x, int y)**

**(4)** What is the **return** type of the method above?

(a) **public** **(b) int**

(c) **void** (d) Nothing is returned.

**(5)** Which of the following is a valid call to the method above?

**(a) minimum(5, 4);** (b) **public static int minimum(5, 4);**

(c) **minimum(int 5, int 4);** (d) **minimum(int x, int y);**

The following exercises refer to the program code on the following page.

**(6)** Which of the following identifiers is visible in method **one**?

**(a) rate ( before main )** (b) **x** ( block three’s local variable )

(c) local variables of method **two** (d) **one** ( method **two**’s formal parameter )

**(7)** Which of the following identifiers is NOT visible in method **two**?

(a) **rate** ( before **main** ) (b) **w** ( before method **two** )

**(c) x ( variable in block three )** (d) **one** ( method **name** )

**(8)** Which of the following identifiers is visible in **main**?

**(a) z ( before main )** **(b) w ( before method two )**

(c) local variables of method **two** (d) all identifiers are visible in **main**

**(9)** Which of the following identifiers is visible in block three?

**(a) z ( before main )** **(b) t ( before main )**

(c) local variables of **main** (d) **x** ( **one**’s formal parameter )

**(10)** Which of the following identifiers is NOT visible in block three?

**(a) z ( before main )** **(b) t ( before main )**

(c) **main** (d) local variables of method **two**

**(11)** Where is identifier **x** ( block three’s local variable ) visible?

(a) in one and block three **(b) in block three only**

(c) in two and block three (d) in block three and **main**

**(12)** Which of the following is an example of a local identifier in the example below?

(a) **rate** ( line 3 ) (b) **w** ( line 20 )

(c) **t** ( line 5 ) **(d) a ( line 25 )**

|  |
| --- |
|  |

**public class ScopeRules // line 1**

**{**

**static final double rate = 5.10;**

**static int z;**

**static double t;**

**public static void main(String args[]) // line 7**

**{**

**int num;**

**double x, z;**

**char ch;**

**// main block...**

**}**

**public static void one(int f, char g) // line 15**

**{**

**// block one...**

**}**

**public static int w; // line 20**

**public static void two(int one,int i) // line 22**

**{**

**char ch;**

**int a;**

**//block three**

**{**

**int x = 12; // line 29**

**//...**

**}//end block three**

**// block two... // line 32**

**}**

**}**

|  |
| --- |
|  |