Java Language: Some Basic Elements

Numeric Operators

+	-	*
/	%	++
- -		

Numeric Comparators

< =	> =	= =
<	>	! =

Grouping & Punct. Symbols

{	}	;
(	)	,
[	]	

Boolean Operators

true	false
&&	| |
!	

Control Operators

if	else
for	while
return	break
continue	

Assignment Operators

=	+=
-=	

Data Types

int	double
long	String
char	boolean

Math Function Library

Math.sqrt( )	Math.pow( )
Math.abs( )	Math.PI
Math.sin( )	Math.cos( )
Math.exp( )	Math.log( )
Math.min( )	Math.max( )

String Operators & Methods

+	" "
length( )	substring( )
equals( )	equalsIgnoreCase( )
charAt( )	compareTo( )

Object Specifiers,
Operators & Methods

class	static
public	private
new	.
main( )	equals( )
toString( )	

System Print Methods

System.out.print( )
System.out.println( )
System.out.printf( )

Array Operators
& Methods

a [*i*]
new
length
clone( )
toString( )

Convert String to Number

Integer.parseInt( )
Double.parseDouble( )

Java Language: Logic 2

In this section we will extend our Java Logic development to include logical flow of control using **if**, **else**, and **return**. In Java parlance, a Java "method" is essentially a ***mathematical function*** or ***procedure*** that produces (returns) an answer based on given logical conditions and mathematical calculations. However, note that not all Java methods return an answer, e.g. a method may simply update an already existing internal variable such as an average or a total.

The problems in this section will address the construction of Java code fragments (statements) that could be used to define a Java method. These code fragments are not complete Java methods, but just part of them. Later, we will use these code fragments to create complete Java methods.

Problem 1. Given two integer parameters **a** and **b**, return their sum if they aren't equal, otherwise return double their sum. Construct a Java code fragment (statements) for a Java method that will return the correct answers under the given conditions.

general format	long way 1	long way 2	short way 1	short way 2
if (_condition_**)**	**int ans1 = a + b;**	**int ans1 = a + b;**	**if (a != b)**	**if (a == b)**
{	**int ans2 = 2*ans1;**	**int ans2 = 2*ans1;**	**return (a + b);**	**return 2*(a + b);**
....			**else**	**else**
....	**if (a != b)**	**if (a == b)**	**return 2*(a + b);**	**return (a + b);**
return _ans1_**;**	**{**	**{**		
}	**return ans1;**	**return ans2;**		
else	**}**	**}**		
{	**else**	**else**		
....	**{**	**{**		
....	**return ans2;**	**return ans1;**		
return _ans2_**;**	**}**	**}**		
}				

In the table above we have given examples of four different correct solutions. Notice that in the two long versions we have defined integer variables **ans1** and **ans2**, which strictly speaking isn't necessary here but often in more complex situations it will be very helpful to do so. Also, in the two long versions we have included the curly braces, which again strictly speaking isn't necessary here, since there is only one statement following the logical flow of control operators **if** and **else**. However, if there is more than one statement following one these logical operators then the curly braces must be included, otherwise the Java code won't be interpreted correctly and hence won't run correctly.

Problem 2. Given an integer parameter **n**, return the absolute difference between **n** and **21**, except return double the absolute difference if **n** is over **21**. Construct a Java code fragment (statements) for a Java method that will return the correct answers under the given conditions.

way 1	way 2
if (n > 21)	**if (n <= 21)**
return 2*Math.abs(n-21);	**return Math.abs(n-21);**
else	**else**
return Math.abs(n-21);	**return 2*Math.abs(n-21);**

Problem 3. Given two integer parameters **a** and **b**, and a boolean parameter **negative**, if **negative** is true then return true if **a** and **b** are both negative. If **negative** is false, return true if **a** and **b** have opposite signs. Otherwise, return false. Construct a Java code fragment (statements) for a Java method that will return the correct answers under the given conditions.

way 1	way 2

if (negative)
{
    if (a < 0 && b < 0)
      return true;
    else
      return false;
}
else
{
    if (a*b < 0)
      return true;
    else
      return false;
}
	if (negative)
	  return (a < 0 && b < 0);
	else
	  return (a*b < 0);

Notice that, since in each of the statements **if (a < 0 && b < 0)** and **if (a*b < 0)** we are testing a boolean (logical) condition and returning true or false based on whether or not the condition itself is true or false, we may simply return the boolean value of the logical condition itself ! As you can see, there are often several ways to write Java code that works. One of the most appealing and interesting aspects of Java programming is to try to find and create the most elegant (logically clear, concise) solution possible.

Homework. Here are some more problems for you to do on your own. In each one, construct a Java code fragment (statements) for a Java method that will return the correct answers under the given conditions.

Problem 4. Given three integer parameters **a**, **b**, and **c**, return the largest one.

Problem 5. Given two integer parameters **a** and **b**, return whichever value is nearest to the value 10, or return 0 in the event of a tie. Hint: Use **Math.abs(x)**.

Problem 6. Given two positive integer parameters **a** and **b**, return the larger value that is in the range from 10 to 20 inclusive, or return 0 if neither is in that range.

Problem 7. Given two integer parameters **a** and **b**, return false if one is negative. Return true if they aren't negative and have the same last digit, such as with 27 and 57. Otherwise, return false. Note that the "mod" operator, **%**, computes remainders, so **17 % 10** is **7**, i.e. 17 mod 10 is the remainder after dividing 17 by 10, which is 7.