**Data Types and Operators**

**(1)** A variable can hold more than one value at a time.

(a) True **(b) False**

**(2)** Multiplication, division and remainder always take place after addition or subtraction in an expression.

(a) True **(b) False**

**(3)** The term parse means to break into component parts.

**(a) True** (b) False

**(4)** Once a variable has been declared and initialized, new values may not be assigned to the variable.

(a) True **(b) False**

**(5)** The expression **boolean isTenLarger = (10 < 5)** will produce a value of true.

(a) True **(b) False**

**(6)** In Java, you use variables of type \_\_\_\_ to store integers, or whole numbers.

(a) num (b) var

(c) double **(d) int**

**(7)** A(n) \_\_\_\_ variable can hold only one of two values: true or false.

(a) integer (b) true

**(c) boolean** (d) comparison

**(8)** In Java, when a numeric variable is concatenated to a String using the \_\_\_\_ , the entire expression becomes a String.

**(a) plus sign** (b) concatenate statement

(c) equal sign (d) string statement

**(9)** What is the value of result after the following statement is executed?

**int result = 2 + 3 \* 4;**

(a) 9 **(b) 14**

(c) 10 (d) 20

**(10)** Which of the following is NOT the name of a Java primitive data type?

(a) int (b) float

(c) double **(d) String**

Match each term with the correct statement below.

(a) operand (f) primitive

(b) cast operator (g) float

(c) assignment (h) boolean

(d) operator precedence (i) escape sequence

(e) garbage

**(h)** **(11)** true or false

**(c)** **(12)** The operator that is represented by an equal sign (=)

**(e)** **(13)** A programming term for an unknown value

**(f)** **(14)** Java consistently specifies their size and format

**(a)** **(15)** A value that can be used on either side of an operator

**(d)** **(16)** Rules for the order in which parts of a mathematical expression are evaluated

**(g)** **(17)** A floating - point data type

**(b)** **(18)** Created by placing the desired result type in parentheses

**(i)** **(19)** Begins with a backslash followed by a character

**(20) 100 = salesAmount;**

In terms of assignment operators, why is the above statement illegal?

|  |
| --- |
|  |

**This assignment operator has a right-to-left associativity. Associativity is the order in which values are used with operators. Since 100 is a numeric constant, it is an rvalue, which is an item that can appear only on the right side of the assignment operator. An identifier that can appear on the left side of an assignment operator is referred to as an lvalue (left-to-right associativity).**

**(21)** Write the statement that will declare and assign two integer variables, **salesAmt** and **costAmt**, in a single statement. Assign values of your choice to the variables.

ANS: **int salesAmt = 100, costAmt = 15;**

**(22)** Write the statement that will declare a char data type named **testScore** that will hold a letter grade of your choice.

ANS: **char testScore = 'A';**

**(23) public class YourGrade**

**{**

**public static void main(String[] args)**

**{**

**int projectPoints = 89;**

**System.out.print("Your grade for this class is ");**

**System.out.print(projectPoints);**

**System.out.println("%");**

**}**

**}**

Given the above code, what will be the output at the command prompt?

ANS: Output will be as follows:

**Your grade for this class is 89%**

A blank line will follow the output.

**Decision - Making Code Blocks**

**(1)** A(n) \_\_\_\_ consists of written steps in diagram form, as a series of shapes connected by arrows.

(a) pseudocode chart (b) sequence structure

**(c) flowchart** (d) decision structure

**(2)** A logical structure called a(n) \_\_\_\_ structure is when one step follows another unconditionally.

(a) straight **(b) sequence**

(c) decision (d) unconditional

**(3)** The simplest statement you can use to make a decision is the \_\_\_\_ statement.

(a) this (b) true / false

(c) Boolean **(d) if**

**(4)** When an expression containing a \_\_\_\_ is part of an if statement, the assignment is illegal.

**(a) single equal sign** (b) double equal sign

(c) greater than sign (d) Boolean value

**(5)** A(n) \_\_\_\_ statement is the decision structure you use when you need to take one or the other of two possible courses of action.

(a) Boolean (b) single - alternative if

**(c) dual - alternative if** (d) if . . . else

**(6)** When you execute an if . . . else statement, only one of the resulting actions takes place depending on the evaluation of the \_\_\_\_ following the if .

**(a) Boolean expression** (b) else statement

(c) keyword (d) independent statement

**(7)** When you place a block within an if statement, it is crucial to place the \_\_\_\_ correctly.

(a) periods (b) commas

(c) angle brackets **(d) curly braces**

**(8)** The AND operator is written as two \_\_\_\_ .

(a) plus signs **(b) ampersands**

(c) equal signs (d) asterisks

**(9)** You can use the \_\_\_\_ , which is written as || , if you want some action to occur when at least one of two conditions is true.

**(a) conditional OR operator** (b) range check

(c) logical AND operator (d) switch statement

**(10)** if (quotaAmt > 100 || sales > 100 && productCode == "C")

bonusAmt = 50;

When the above code is executed, which operator is evaluated first?

**(a) &&** (b) || (c) == (d) =

Match each term with the correct statement below.

(a) relational operator (f) pipes

(b) conditional operator (g) dual - alternative

(c) Boolean values (h) !

(d) equality (i) AND operator

(e) switch statement

**(c)** **(11)** true and false

**(d)** **(12)** A double equal sign

**(g)** **(13)** Two possible courses of action

**(i)** **(14)** Two ampersands

**(f)** **(15)** Used in the OR operator

**(a)** **(16)** > or <

**(e)** **(17)** An alternative to using a series of nested if statements

**(b)** **(18)** Separated with a question mark and a colon

**(h)** **(19)** The logical NOT operator

**(20)** What is wrong with the following statement? How could you correct it?

if(payRate < 5.85 && payRate > 60)

System.out.println("Error in pay rate");

**ANS: As a single variable, no payRate value can ever be both below 5.85 and over 60 at the same time. Therefore, the output statement can never execute, no matter what value the payRate has. In this case, you must write the following code to print the error message under the correct circumstances:**

**if(payRate < 5.85 || payRate > 60)**

**System.out.println("Error in pay rate");**

**(21)** Create an if statement that will check if the variable salesAmt is greater than the constant SALES\_QUOTA . If true, create a block statement that will calculate salesAmt \* BONUS\_PERCENT and store the result in a variable named bonusPay . The block statement will then compute totalPay by adding salesAmt to   
 bonusPay . Outside of the if structure, add a println statement that will output the totalPay value.

**ANS: if(salesAmt > SALES\_QUOTA) {**

**bonusPay = salesAmt \* BONUS\_PERCENT**

**totalPay = salesAmt + bonusPay**

**}**

**System.out.println("Total pay is " + totalPay);**

**(22)** Create an if statement that checks if the variable salesAmt is greater than or equal to the QUOTA\_AMT constant. Use the logical OR operator to also check if salesAmt is greater than or equal to the HIGH\_SALES constant. If either expression is true, assign the value of the LARGE\_QUOTA constant to the variable bonusAmt .

**ANS:**

**if(salesAmt >= QUOTA\_AMT || salesAmt >= HIGH\_SALES)**

**bonusAmt = LARGE\_QUOTA;**

**More on If Statements**

**(1)** You write pseudocode in everyday language, not the syntax used in a programming language.

**(a) True** (b) False

**(2)** An alternative to using a Boolean expression, such as someVariable == 10, is to store the Boolean expression’s value in a Boolean variable.

**(a) True** (b) False

**(3)** When you create a block, you must place multiple statements within it.

(a) True **(b) False**

**(4)** When you use nested if statements, you must pay careful attention to placement of any else clauses.

**(a) True** (b) False

**(5)** In the switch structure, break is followed by one of the possible values for the test expression and a colon.

(a) True **(b) False**

**(6)** Computers contain switches that are set to on or off.

**(a) True** (b) False

**(7)** When writing a statement with the two - line format, you must be sure to type a semicolon at the end of the first line in order to ensure accurate results.

(a) True **(b) False**

**(8)** Although it is possible to block statements that depend on an if, you cannot likewise block statements that depend on an else.

(a) True **(b) False**

**(9)** Although not required, it is common procedure to align the keyword if with the keyword else.

**(a) True** (b) False

**(10)** Range checking and the switch statement are tools programmers can use for effective decision making.

**(a) True** (b) False

**(11)** Describe how a sequence structure works.

**ANS:**

**In a sequence structure, one step follows another unconditionally. A sequence structure might contain any number of steps; but when one task follows another with no chance to branch away or skip a step, you are using a sequence.**

**(12)** An alternative to using a Boolean expression, such as quizScore == 10 , is to store the Boolean expression’s value in a Boolean variable. How could you express this in Java?

**ANS:**

**For example, if isPerfectScore is a Boolean variable, then the following statement compares quizScore to 10 and stores true or false in isPerfectScore :**

**isPerfectScore = (quizScore == 10);**

**Then, you can write the if as:**

**if(isPerfectScore)**

**System.out.println("The value is 10");**

**(13)** Explain the use of the NOT operator.

**ANS:**

**You use the NOT operator, which is written as the exclamation point   
 ( ! ) , to negate the result of any Boolean expression. Any expression that evaluates as true becomes false when preceded by the NOT operator; and accordingly, any false expression preceded by the NOT operator becomes true.**

**(14)** public class Student

{

private int studentNum;

private int studentScore;

public int MAX\_NUM = 500;

public int MAX\_SCORE = 100;

Student(int num, int score)

{

}

}

Decision making can be used to control the allowed values in an object’s fields. In the above code, the Student class contains two fields that hold a student number and a score. A constructor accepts values for these fields as parameters. Write the code between the curly brackets that will determine whether the value of num is less than the MAX\_NUM constant. If true, assign the value of num to studentNum . Otherwise, assign the value of MAX\_NUM to the studentNum . Then check if the value of score is less than or equal to MAX\_SCORE . If true, assign the value of score to studentScore . Otherwise, assign 0 to studentScore .

**ANS:**

**if(num <= MAX\_NUM)**

**studentNum = num;**

**else**

**studentNum = MAX\_NUM;**

**if(score <= MAX\_SCORE)**

**studentScore = score;**

**else**

**studentScore = 0;**

**(15)** Write a single - alternative if statement that will check if a variable named maxValue is equal to 100 . If true, use a println statement to output " You have reached the limit " .

**ANS:**

**if (maxValue == 100)**

**System.out.println ("You have reached the limit");**

**( Be sure there is no semicolon at the end of the first line. )**

**(16)** Create an if . . . else statement that will check if the maxValue variable is equal to 100 . If true, create a println statement that will output " Max reached " . If false, create a println statement to output " Keep trying " .

**ANS:**

**if(maxValue == 100)**

**System.out.println("Max reached");**

**else**

**System.out.println("Keep trying");**

**When you execute an if . . . else statement, only one of the resulting actions takes place depending on the evaluation of the Boolean expression. Each statement, the one following the if and the one following the else, is a complete statement, so each ends with a semicolon.**

**(17)** if(qtySold > QUOTA)

bonusPay = 50;

totalPay = regPay + bonusPay;

System.out.println("Your totalPay amount is " + totalPay);

Why is the above code incorrect? What would be required to make the code execute correctly?

**ANS:**

**When you place a block within an if statement, it is crucial to place the curly braces correctly. In the above code, the curly braces have been omitted. When qtySold > QUOTA is true, bonusPay is calculated and the if expression ends. The next statement that computes totalPay executes every time the program runs, no matter what value is stored in qtySold . This last statement does not depend on the if statement; it is an independent, stand-alone statement. The indentation might be deceiving; it looks as though two statements depend on the if statement, but indentation does not cause statements following an if statement to be dependent. Rather, curly braces are required if multiple statements must be treated as a block.**

**Looping Code Blocks**

**(1)** A \_\_\_\_ is a structure that allows repeated execution of a block of statements.

(a) body **(b) loop**

(c) Boolean expression (d) loop control

**(2)** A loop controlled by the user is a type of \_\_\_\_ loop.

**(a) indefinite** (b) counter - controlled

(c) definite (d) incrementing

**(3)** A(n) \_\_\_\_ loop is a special loop that is used when a definite number of loop iterations is required.

(a) while (b) else

**(c) for** (d) do . . . while

**(4)** A for loop provides a convenient way to create a(n) \_\_\_\_ loop.

**(a) counter - controlled** (b) while

(c) posttest (d) infinite

**(5)** How many times will outputLabel be called?

for(customer = 1; customer <= 20; ++customer)

for(color = 1; color <= 3; ++color)

outputLabel();

(a) 0 (b) 20 (b) 3 **(d) 60**

**(6)** The order of the conditional expressions in the following is most important within a(n) \_\_\_\_ loop.

while(requestedNum > LIMIT || requestedNum < 0) . . .

**(a) nested** (b) pretest

(c) posttest (d) indefinite

**(7)** In the expressions b = 8 and c = --b , what value will be assigned to the variable c ?

(a) 8 (b) 9 **(c) 7** (d) 10

**(8)** When creating a for loop, which statement will correctly initialize more than one variable?

(a) for a = 1, b = 2 **(b) for(a = 1, b = 2)**

(c) for(a = 1; b = 2) (d) for(a = 1 && b = 2)

**(9)** As long as methods do not depend on one another, \_\_\_\_ is a technique that can improve loop performance by combining two loops into one.

**(a) loop fusion** (b) short - circuit evaluation

(c) prefix incrementing (d) do - nothing looping

**(10)** Which of the following is NOT a valid method to increase a variable named score by 1 ?

(a) ++score **(b) ++score = score + 1**

(c) score++ (d) score = score + 1

Match each term with the correct statement below.

(a) prefix ++ (f) loop fusion

(b) block (g) decrementing

(c) definite loop (h) for loop

(d) loop control variable (i) sleep() method

(e) binary operators

**(b)** **(11)** Multiple statements within curly braces

**(d)** **(12)** The value that determines whether loop execution continues

**(g)** **(13)** Subtracting 1 from a variable

**(c)** **(14)** A counter - controlled loop

**(a)** **(15)** A shortcut for incrementing and accumulating

**(e)** **(16)** Operate on two values

**(f)** **(17)** The technique of combining two loops into one

**(i)** **(18)** Part of the Thread class in the java.lang package

**(h)** **(19)** Within parentheses are three sections separated by exactly two semicolons

**(20)** public class DoWhileExample {

public static void main(String[] args) {

int currentValue;

\_\_\_\_

\_\_\_\_

\_\_\_\_

\_\_\_\_

\_\_\_\_

}

}

Complete the code above by writing a do . . . while posttest loop that will output the value of the variable currentValue in a println statement. After

the output statement, add a decrement statement to decrease the value of   
 currentValue by 1 . Continue the loop until currentValue is equal to 0 .

**ANS:**

**do**

**{**

**System.out.println("The value of currentValue is " + currentValue);**

**++currentValue;**

**}while(response > 0);**

**( Other incrementing options are possible. )**

**(21)** Write a definite while loop that initializes a loop control variable named decreaseOne to 10 and continues until decreaseOne > 0 . Decrement the loop control variable by 1 and include the println output " keep going " within the loop.

**ANS:**

**decreaseOne = 10;**

**while(decreaseOne > 0);**

**{**

**System.out.println("Keep going");**

**decreaseOne = decreaseOne - 1;**

**}**

**(22)** Loop control variables can be evaluated at the start or the end of the loop. Describe both pretest loops and posttest loops. How are do . . . while loops execute?

**ANS:**

**In a pretest loop, the loop control variable is evaluated at the top of the loop before the body has a chance to execute. Both while loops and for loops are pretest loops in that the loop control variable is tested before the loop body executes.**

**A posttest loop is used if you need to ensure that a loop body executes at least one time. In this case, you write a loop that checks at the bottom of the loop after the first iteration. The do . . . while loop is a posttest loop that tests the loop control variable after the loop body executes.**

**Characters and Strings**

**(1)** A(n) \_\_\_\_ is a variable that holds a memory address.

**(a) reference** (b) buffer

(c) Character (d) immutable

**(2)** The String class \_\_\_\_ method evaluates the contents of two String objects to determine if they are equivalent.

**(a) equals()** (b) charAt()

(c) toString() (d) replace()

**(3)** When the String class \_\_\_\_ method is used to compare two Strings, it provides additional information to the user in the form of an integer value.

(a) toString() (b) equals()

**(c) compareTo()** (d) equalsIgnoreCase()

**(4)** The \_\_\_\_ method returns the length of a String .

(a) getSize() (b) size()

(c) getLength() **(d) length()**

**(5)** The \_\_\_\_ method allows you to replace all occurrences of some character within a String .

(a) substring() (b) replaceCharacter()

(c) toString() **(d) replace()**

**(6)** The \_\_\_\_ method converts any object to a String.

(a) convertString() (b) replace()

**(c) toString()** (d) substring()

**(7)** To convert a String to an integer, you use the valueOf() method of the \_\_\_\_ class.

(a) StringBuilder **(b) Integer**

(c) String (d) Convert

**(8)** The \_\_\_\_ method takes a String argument and returns its double value.

(a) parseString() **(b) parseDouble()**

(c) parseInt() (d) returnDouble()

**(9)** You can tell that the equals() method takes a \_\_\_\_ argument because parentheses are used in the method call.

**(a) String** (b) Double

(c) Null (d) Boolean

**(10)** Which of the following correctly declares and initializes a String object?

**(a) String greeting = "Hello";** (b) new String = Hello

(c) String greeting == "Hello"; (d) String new = "Hello"

**(11)** System.out.println("Your name is " + yourName);

The above statement is an example of \_\_\_\_ , which is used to join Strings.

**(a) concatenation** (b) parsing

(c) referencing (d) buffering

Match each term with the correct statement below.

(a) parseDouble() (f) buffer

(b) toLowerCase() (g) setLength()

(c) substring() (h) indexOf()

(d) String variable (i) insert()

(e) append()

**(b)** **(12)** Returns the lowercase equivalent of the argument

**(d)** **(13)** A named object of the String class

**(g)** **(14)** Change the length of a string in a StringBuilder object

**(h)** **(15)** Determines whether a specific character occurs within a String

**(c)** **(16)** Takes two integer arguments: a start position and an end position

**(a)** **(17)** Takes a String argument and returns its double value

**(f)** **(18)** A memory block

**(e)** **(19)** Add characters to the end of a StringBuilder object

**(i)** **(20)** Add characters at a specific location within a StringBuilder object

**(21)** StringBuilder greeting = new StringBuilder("Hello, John");

char initial = greeting.charAt(7);

Using the above code and the charAt() method, what value will be assigned to the variable initial?

ANS:

**The charAt() method will return a "J", which is the seventh character in the String "Hello, John".**

**(22)**  String greeting = "Welcome back";

Using the above statement, write the length() method that will return the length of the greeting String . Store the length in an integer named greetingLength .

**ANS: int greetingLength = greeting.length();**

**(23)** StringBuilder greeting = new StringBuilder("Welcome");

Create the append() method to add the characters "home" to the end of the StringBuilder object created above.

**ANS: greeting.append(" home");**

**( Be sure that the String in double quotes has a blank space at the start. Otherwise, the Strings "Welcome" and "home" will be appended without a blank space. )**