Quiz on Chapter 3

**Question 1:**

What is an *algorithm*?

* An algorithm is a clear, step-by-step technique for completing a certain task that is guaranteed to complete after a set number of steps. (An algorithm is not the same as a programme, but it might be the inspiration for one.)

**Question 2:**

Explain briefly what is meant by "pseudocode" and how is it useful in the development of algorithms.

* Pseudocode is an informal description of an algorithm written in a language that mimics the structure of a programming language but lacks the rigid grammar. Pseudocode can be utilised in the development of an algorithm through progressive refining. You may begin with a simple pseudocode description of the algorithm and gradually add depth to it through a series of revisions until you have something that can be readily converted into a programme written in a real programming language.

**Question 3:**

What is a *block statement?* How are block statements used in Java programs?

* A block statement is just a series of Java statements separated by and. A block statement is the body of a subroutine. In control structures, block statements are often utilised. A block statement is typically used to combine together numerous statements so that they may be utilised in situations when just one statement is required. A while loop, for example, requires only one statement: "while (condition) do statement." The statement, on the other hand, can be a block statement, with the structure: "while (condition) { statement; statement; ...}".

**Question 4:**

What is the main difference between a while loop and a do..while loop?

* Both forms of loops repeat a set of statements until a condition is met. The primary distinction is that the condition in a while loop is checked at the beginning of the loop, but the condition in a do..while loop is tested at the conclusion of the loop. It is possible that the body of a while loop will never be performed. The body of a do..while loop, on the other hand, is run at least once since the condition for stopping the loop is not verified until the body of the loop is executed.

**Question 5:**

What does it mean to *prime* a loop?

* The condition at the start of a while loop must make sense even when verified for the first time before the body of the loop is run. To prime the loop, you put things up ahead of time so that the test makes sense (that is, the variables that it contains have reasonable values). For example, if the loop's test is "when the user's answer is yes," you must prime the loop by receiving a response from the user (or making one up) before running the loop.

**Question 6:**

Explain what is meant by an *animation* and how a computer displays an animation.

* An animation is made up of a succession of "frames." Each frame is a still image, although there are small changes between them. When pictures are shown quickly one after the other, the eye sees motion. A computer shows an animation by displaying one image on the screen, then replacing it with the next image, and so on.

**Question 7:**

Write a for loop that will print out all the multiples of 3 from 3 to 36, that is: 3 6 9 12 15 18 21 24 27 30 33 36.

* Here are two possible answers. Assume that N has been declared to be a variable of type int:

for ( N = 3; N <= 36; N = N + 3 ) {

System.out.println( N );

}

or

for ( N = 3; N <= 36; N++ ) {

if ( N % 3 == 0 )

System.out.println( N );

}

**Question 8:**

Fill in the following main() routine so that it will ask the user to enter an integer, read the user's response, and tell the user whether the number entered is even or odd. (You can use TextIO.getInt() to read the integer. Recall that an integer n is even if n % 2 == 0.)

public static void main(String[] args) {

// Fill in the body of this subroutine!

}

* The problem already gives an outline of the program. The last step, telling the user whether the number is even or odd, requires an if statement to decide between the two possibilities.

import textio.TextIO;

public static void main (String[] args) {

int n; // the number read from the user

TextIO.put("Type an integer: "); // ask the use to enter an integer

n = TextIO.getInt(); // read the user's response

if (n % 2 == 0) // tell the user whether the number is even or odd

System.out.println("That's an even number.");

else

System.out.println("That's an odd number.");

}

**Question 9:**

Write a code segment that will print out two *different* random integers selected from the range 1 to 10. All possible outputs should have the same probability. Hint: You can easily select two random numbers, but you have to account for the fact that the two numbers that you pick might be the same.

* The code for selecting two random integers has to be wrapped in a loop that will end only when the two selected numbers are different. This can be done easily with a do..while loop. A while loop can also be used, but it must be "primed" in some way. Note that by using a loop to choose the numbers, we can be absolutely sure that after the loop ends, the two numbers are different. Here are three possible solutions:

(1) int x,y; // Two random integers.

do {

x = (int)(10\*Math.random() + 1);

y = (int)(10\*Math.random() + 1);

} while (x == y); // continue if x and y are the same number.

System.out.println(x + " " + y);

(2) int x,y; // Two random integers.

x = y = 0; // Give them junk value to "prime" the loop to make sure it runs.

while (x == y) {

x = (int)(10\*Math.random() + 1);

y = (int)(10\*Math.random() + 1);

}

System.out.println(x + " " + y);

(3) int x,y; // Two random integers.

x = (int)(10\*Math.random() + 1); // Pick x, and keep its value.

do { // The loop finds a y with a different value from x.

y = (int)(10\*Math.random() + 1);

} while (x == y);

System.out.println(x + " " + y);

**Question 10:**

Suppose that s1 and s2 are variables of type *String*, whose values are expected to be string representations of values of type int. Write a code segment that will compute and print the integer sum of those values, or will print an error message if the values cannot successfully be converted into integers. (Use a try..catch statement.)

* The function Integer.parseInt can be used to convert the strings into integers. This function will throw an exception of type *NumberFormatException* if the conversion fails. A try..catch statement can catch this exception and print an error message. So, the code segment can be written:

try {

int n1, n2; // The values of s1 and s2 as integers.

int sum; // The sum of n1 and n2.

n1 = Integer.parseInt(s1);

n2 = Integer.parseInt(s2);

sum = n1 + n2; // (If an exception occurs, we don't get to this point.)

System.out.println("The sum is " + sum);

}

catch ( NumberFormatException e ) {

System.out.println("Error! Unable to convert strings to integers.");

}

**Question 11:**

Show the exact output that would be produced by the following main() routine:

public static void main(String[] args) {

int N;

N = 1;

while (N <= 32) {

N = 2 \* N;

System.out.println(N);

}

}

* The exact output printed by this program is:

2

4

8

16

32

64

**Question 12:**

Show the exact output produced by the following main() routine:

public static void main(String[] args) {

int x,y;

x = 5;

y = 1;

while (x > 0) {

x = x - 1;

y = y \* x;

System.out.println(y);

}

}

The way to answer this question is to trace exactly what the program does, step-by-step. The output is shown below on the right. On the left is a table that shows the values of the variables x and y as the program is being executed.

value of x | value of y OUTPUT

--------------|-------------- -------------

5 | 1 [ before loop]

4 | 4 [ = 1\*4 ] 4

3 | 12 [ = 4\*3 ] 12

2 | 24 [ = 12\*2 ] 24

1 | 24 [ = 24\*1 ] 24

0 | 0 [ = 24\*0 ] 0

**Question 13:**

What output is produced by the following program segment? **Why?** (Recall that name.charAt(i) is the i-th character in the string, name.)

String name;

int i;

boolean startWord;

name = "Richard M. Nixon";

startWord = true;

for (i = 0; i < name.length(); i++) {

if (startWord)

System.out.println(name.charAt(i));

if (name.charAt(i) == ' ')

startWord = true;

else

startWord = false;

}

This is a tough one! The output from this program consists of the three lines:

R

M

N

As the for loop in this code segment is executed, name.charAt(i) represents each of the characters in the string "Richard M. Nixon" in succession. The statement System.out.println(name.charAt(i)) outputs the single character name.charAt(i) on a line by itself. However, this output statement occurs inside an if statement, so only some of the characters are output. The character is output if startWord is true. This variable is initialized to true, so when i is 0, startWord is true, and the first character in the string, 'R', is output. Then, since 'R' does not equal ' ', startWorld becomes false, so no more characters are output until startWord becomes true again. This happens when name.charAt(i) is a space, that is, just *before* the 'M' is processed and again just before the 'N' is processed. In fact whatever the value of name, this for statement would print the first character in name and every character in name that follows a space. (It is *almost* true that this for statement prints the first character of each word in the string, but something goes wrong when the first character is a space or when there are two spaces in a row. What happens in these cases?)

**Question 14:**

Suppose that numbers is an array of type int[]. Write a code segment that will count and output the number of times that the number 42 occurs in the array.

Use a for loop to go through the array and test each array element. If the value is 42, add 1 to a counter:

int count42; // The number of 42s in the array

count42 = 0;

int i; // loop control variable

for ( i = 0; i < numbers.length; i++ ) {

if ( numbers[i] == 42 ) {

count42++;

}

}

System.out.println("There were " + count42 + " 42s in the array.")

**Question 15:**

Define the range of an array of numbers to be the maximum value in the array minus the minimum value. Suppose that raceTimes is an array of type double[]. Write a code segment that will find and print the range of raceTimes.

We need both the minimum and the maximum value in the array. We can compute both using one for loop.

double max; // maximum value from the array

double min; // minimum value from the array

double range; // the range of the array, max - min

int i; // for-loop variable

max = min = raceTimes[0]; // start with both equal to the first element

for ( i = 1; i < raceTimes.length; i++ ) {

if ( raceTimes[i] > max )

max = raceTimes[i];

if ( raceTimes[i] < min )

min = raceTimes[i];

}

range = max - min;

System.out.println("The range is " + range);