CP2406 Quiz 2

**Question 1:**

Briefly explain what is meant by the *syntax* and the *semantics* of a programming language. Give an example to illustrate the difference between a syntax error and a semantics error.

* A language’s syntax is its grammar, and its semantics is its meaning. A programme that has a syntax mistake cannot be compiled. A programme with a semantic mistake can be built and run but the output is erroneous. A missing semicolon in a programme is an example of a syntax mistake, as the compiler will detect and report the issue. If N is an integre variable, the phrase “frac = 1/N;” is most likely a semantic mistake. For every N bigger that one, the value of 1/N will equal 0. The coder most likely intended to say 1.0/N.

**Question 2:**

What does the computer do when it executes a variable declaration statement. Give an example.

* A variable is a named “box” or region in the computer’s memory. The box contains a value of a specific kind. A variable declaration statement is something like this:

Int x;

* This produces the variable x. When variable declaration is executed, the computer constructs the box in a memory and assigns a name (in this example, x) with that box. The variable can be referred to by name later in the programme.

**Question 3:**

What is a *type,* as this term relates to programming?

* A “type” is a collection of potential values. When you declare a variable’s type, you are indicating what values it can hold. When you state an expression is of a given type, you’re referring to the values that the expression can have. For example, stating that a variable is a type int implies that it may contains integer values within a specific range.

**Question 4:**

One of the primitive types in Java is *boolean.* What is the boolean type? Where are boolean values used? What are its possible values?

* True and False are the only Boolean values. Boolean expressions are used in situations where true/false values are expected, such as a while loop and if statements.

**Question 5:**

Give the meaning of each of the following Java operators:

a) ++

b) &&

c) !=

1. To add one to the value of a variable, use the operator ++. “count++”, for example, has the same effect as “count = count + 1”.
2. The operator && stands for the word and. It may be used to combine two Boolean values, such as “(x>0 &&y>0)”, which means “x is more than 0 and y is greater than 0”.
3. The operator != indicates “is not equals to”, as in “if (x!=0)”, meaning “if x is not equals to zero”.

**Question 6:**

Explain what is meant by an *assignment statement,* and give an example. What are assignment statements used for?

* An assignment statement computes a value and assigns it to a variable.
* Example:
  + X = 2; assign a constant value to the variable, x.
  + newVariable = variable; Copying the value, variable, into the newVariable
  + ans = 2\*x + 42; compute the value of the expression and store it in ans
* When the programme is executing, an assignment statement is used to alter the value of a variable. Because the variable’s value can be another variable or an expression, assignments statements can be used to copy data from one location in the computer to another as well as to do sophisticated computations.

**Question 7:**

What is meant by *precedence* of operators?

* Whenever an expression has two or more operators and there are no parentheses indicating the order in which the operators should be evaluated, the computer must choose which operator to evaluate first. The precedence of the operators determines the sequence. Since \* take priority over +, the phrase 3 + 5 \* 7 is evaluated as if it were written 3+(5\*7)

**Question 8:**

What is a *literal*?

* A literal is a string of letters that is used to represent a constant value in a programme. For example, ‘A’ is a char literal that represents the value A, while 17L is a long literal that represents the integer 17. A literal is a method of expressing a value that should not be mistaken with the value itself.

**Question 9:**

In Java, classes have two fundamentally different purposes. What are they?

* A class can be used to aggregate variables and subroutines included within the class. They are known as the class’s static members. For instance, the subroutine Math.sqrt is a static member of the Math class. A static member of a class is also the primary routine in any programme. A class’s second conceivable role is to describe and generation things. The class defines the variables and subroutines such object posses. Classes are utilised in this role in object-oriented programming (which we haven’t explored in depth yet).

**Question 10:**

What is the difference between the statement "x = TextIO.getDouble();" and the statement "x = TextIO.getlnDouble();"

* Either statement will read a user-supplied real number and save it in the variable x. They would both read and provide the same result. The difference is that in the second sentence (using getlnDouble), the computer will continue reading characters from input up to and including the next carriage return after reading the value. These superfluous characters will be removed.

**Question 11:**

Explain why the value of the expression 2 + 3 + "test" is the string "5test" while the value of the expression "test" + 2 + 3 is the string "test23". What is the value of "test" + 2 \* 3 ?

* The explanation is a little technical. The sequence of evaluation accounts for the discrepancy. When many + operators are used in a row without parenthesis, they are evaluated left to right 2 + 3 + “test” is understood as (2 + 3) + “test”, therefore 2 and 3 are put together, yielding 5, and then the 5 is concatenated onto the string “test”. “test” + 2 + 3 is understood as (“test” + 2) + 3, thus the 2 is concatenated onto the “test” first, yielding “test2”, and then the 3 is concatenated onto that. The precedence rules for + and \* apply in the instance of “test” + 2 \* 3. Since \* takes priority, this equation is evaluated as “test” + (2\*3), which equals “test6”.

**Question 12:**

Integrated Development Environments such as Eclipse often use syntax coloring, which assigns various colors to the characters in a program to reflect the syntax of the language. A student notices that Eclipse colors the word *String* differently from int, double, and boolean. The student asks why *String* should be a different color, since all these words are names of types. What's the answer to the student's question?

* Although String is a type name, like int, double, and Boolean, there is a fundamental distinction between String and the other kinds. String is a class name, whereas int, double, and Boolean are primitive types. Eclipse colours all class names in the same manner that it colours String, and it colours primitive kinds differently. (While the distinction between classes and primitive types may not appear to be significant right now, it is, and Eclipse is justified in using different colours for the two ideas.)

**Question 13:**

What is the purpose of an import directive, such as import textio.TextIO or import java.util.Scanner?

* TextIO and Scanner are specified in “packages” called textio and java.util. This implies that you cant just type “TextIO” or “Scanner” into a programme and expect the machine to understand what they represent. These name must be imported from the appropriate packages before they can be used in a programme

**Question 14:**

Write a complete program that asks the user to enter the number of "widgets" they want to buy and the cost per widget. The program should then output the total cost for all the widgets. Use System.out.printf to print the cost, with two digits after the decimal point. You do not need to include any comments in the program.

import java.util.Scanner;  
  
public class quiz {  
 public static void main(String[] args) {  
 int numberOfWidgets;  
 double costPerWidget;  
 double totalCost;  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Enter the number of widget you want: ");  
 numberOfWidgets = in.nextInt();  
 System.*out*.print("Enter how much does each widget cost: ");  
 costPerWidget = in.nextDouble();  
 totalCost = numberOfWidgets \* costPerWidget;  
 System.*out*.printf("The total cost is $%1.2f%n", totalCost);  
 }  
}