2910109 Introduction to Java and object-oriented programming

Examiner's report: Zone A

Question 1

a. The majority of candidates offered correct answers in part (i). One way of commenting out a code fragment in Java is to employ /* and */ to signal the beginning and the end of the comment respectively. The programmer can furthermore conveniently use // for commenting out entire code lines. Almost all candidates found the compilation error in part (ii). The type declared for x is int, therefore variable x cannot hold boolean values. A handful of candidates failed to realise that the expression 1==1 is a boolean value. The overwhelming majority of candidates answered part (iii) correctly, demonstrating a solid understanding of precedence of arithmetic operations. When evaluating the expression 3+(2+2)*3 the addition inside the bracket is performed first followed by the multiplication operation and finally the other addition operation. The resulting value is then input to println, so that the given Java program prints out the number 15.

b. Most candidates rightly observed that the body of the main method is not enclosed by braces and the program will thus produce a compilation error. Assuming that these braces are added the two main ingredients of a correct answer in this section are a good understanding of all the parts of loops and the order in which these parts are being executed. Most candidates demonstrated a solid understanding of while constructs. The first statement in part (i) initialises the control variable j to 0. Next the condition j < 5 is evaluated and if it is false the loop terminates, otherwise the print-statement is executed and the control variable is incremented by 1. The whole process is then repeated, resulting in the program printing out the numbers 0, 2, 4, 6, 8. Almost all candidates produced equivalent programs in part (ii) employing a for-loop as follows: for(int j=0; j<5; j++) System.out.println(i).

c. Although a small number of candidates were unable to produce a method for sorting a vector of ints into descending order, the majority of answers were perfect. Unfortunately a couple of answers used excessively long or inefficient programs. As code compactness is an important aspect of software development, quite a few marks were lost due to this. Furthermore a handful of candidates produced code for sorting an array rather than a vector of ints. Many candidates left the complexity part of the question blank or gave instead an answer demonstrating that they had very little knowledge of algorithmic time complexity. Time complexity of algorithms is a fundamental aspect of program design and students are advised to focus more on this in the future.

Question 2

a. The majority of candidates gave the answer 97 in part (i), and this is correct since the ASCII code for the character a is two less than the ASCII code for character c. The output of the Java program in part (ii) is simply the ASCII code of character b, namely 98. The

   Java program in (iii) prints out the character that corresponds to ASCII code 101, so the correct answer here is thus e. The Java program in part (iv) prints out the third character contained in the file 'letters', namely c. This is easily worked out by first observing that a FileReader f is used to access the file 'letters', then its read-method is executed three times reading in one character at a time and storing its corresponding ASCII code into a variable x. Finally, prior to executing the print-method, the integer variable x is typecast into a char, and what is printed out is thus the actual character whose ASCII code is the value stored in x. Some candidates lost marks by answering that the program prints out the third character of the string "letters", failing to notice that 'letters' is simply the file name and not the file contents.

b. The majority of candidates demonstrated a good understanding of the use of a while construct for reading in file contents on a character-by-character basis. To achieve this a FileReader f must first be declared and initialised appropriately for accessing the file 'letters'. The while loop's control variable x must then be declared as int and initialised to 1. The condition x!=-1 is then evaluated and if it is false the while loop terminates, otherwise a read is executed to read one further character from the file. In part (i) a variable n (declared as int and initialised to 0 prior to the loop execution) is also increased each time a character is read. The result is that when the end of the file is reached the value of this variable n is the number of characters in the file 'letters', which is printed out with the println statement. In part (ii) the character read inside the while loop is immediately printed out if its corresponding ASCII code is less than 101, thus the output is abcd.

c. The majority of candidates attempted this question. Most successfully set up an instance of a FileReader for accessing 'apple' and provided some code for reading in characters. However, a handful of candidates were unable to provide Java code and resorted to some rough textual description of the code functionality accompanied by specific examples. Some candidates lost quite a few marks due to lengthy code. Others instead of counting the number of occurrences of each lower case letter wrongly counted all lower case letters. A simple way to answer this question would be to declare an array of int values to hold the number of occurrences of file characters. The underlying assumption is that the array component i holds the number of times the character with ASCII code i occurs in 'apple'. The ith component of the array must be initialised to 0 to cater for the case that no occurrences of character with ASCII code i are found in the file. By using a while-loop we could then successively read in the ASCII codes of file characters and incrementing after each read the value in the corresponding array component. To print out the number of occurrences, we simply print out the array contents.

Question 3

a. Most candidates correctly answered that the Java program in part (i) outputs [[12,17]] although a handful failed to realise that left and right square brackets are being printed. The Java program in part (ii) prints simply [[1,2]](3). The most common error here was failure to realise that left and right brackets (just before 1 and after 2) are also printed because the employed class triple is an extension of the class pair which was defined in part (a) of the question.

b. Many candidates attempted to extend the class pair in order to address this part, but this was not required. A fresh class, called quadruple, was needed here, based on two variables, say one and two. The constructor of this new class would accept as arguments four objects, say o1, o2, o3 and o4, and initialise variable one using the constructor of pair based on objects o1, o2 and variable two using the constructor of pair based on objects o3, o4.

c. A handful of candidates attempted this question. The required equality instance method for pairs, say isEqual, should simply accept as input a variable p of type pair and return the boolean value of the expression one==p.one && two==p.two. The equality instance method for triples is then built upon isEqual. Assuming that its input is a variable t of type triple, it returns the boolean value of the expression super.isEqual(t) && three==t.three. Finally, for the quadruple case, the boolean of an expression is returned, the expression being based on two calls to isEqual as the class quadruple consists of two pairs.

Question 4

a. The majority of candidates correctly pointed out that the given program unfortunately contains syntax errors, (e.g. the curly bracket in the second for-loop construct). Assuming that errors are corrected this program first declares an array of ints, initialising the ith component to the value i. Then a for-loop is executed that effectively shifts the values of the array components one position from right to left, that is the value of the component i prior to loop-execution is assigned into component i-1, and the value of the first component is assigned to the last component. Finally, a loop prints out the contents of the array, using a new line for each component.

b. A large number of candidates answered this question correctly, showing a good understanding of the use of logical operators. Students should note the importance of reading questions very carefully before attempting to answer them. A number of candidates produced complete Java programs instead of just boolean expressions as requested. Others offered program fragments based on if-statements, such as, for example, if (x == 5) return true; else return false. In either case valuable examination time was wasted. A sizeable number of scripts wrongly employed the equal sign '=' instead of the comparison operator '=='. An equal sign is used in an assignment statement to assign a value to a variable. So, for example x = y assigns the value of y to variable x. On the other hand the comparison operator is used to compare the results of expressions, so, for example, x == y has the value true or false depending upon whether variables x and y are equal or not. A handful of candidates had difficulty in using the correct symbol for the logical operators (e.g. using & (|) instead of && (||)). The most common problem in part (iv) was the use of only two conditions e.g. x!=y && y!=z. A good answer here would be x!=y && y !=z && x!=z.

c. A fairly large number of candidates attempted this question. It is important for candidates to read the question very carefully before attempting to answer it. Some candidates just printed out the characters of the string in reverse order, while others used built-in methods for reversing strings, thus gaining no marks at all. A simple answer to this question would be a for-loop with control variable i addressing all elements of the original string from the highest order (that is length of the string minus 1) down to 0. Then the reverse string could be constructed by starting from an empty string t (declared and initialised prior to the execution of the loop) and for each execution of the for-loop add into t the character of the original string that corresponds to index i. Some candidates offered very good, compact solutions based on recursion. The most common problem for candidates who adopted this approach was the omission of a test for terminating the recursive process.

Question 5

a. Many candidates explained well the purpose of using the import-statement in Java namely that it saves programmers from referring to things by their full package.class name.

b. The majority of candidates answered parts (i) and (ii) satisfactorily. They showed a good understanding of the purpose of packages in Java. A few students had severe problems with expressing their answers in good English. Also a handful of candidates failed to realise that java.io should be used as a prefix to IOException in part (ii). A couple of candidates have wrongly attached the java.io-prefix to methods (e.g. readLine).

c. A large number of candidates attempted this question and most offered correct answers. The simple answer in part (i) is 2^32, but some students lost valuable examination time here in attempting to compute the value of this. The program in part (ii) simply prints out 3, namely the contents of variable x which is declared and initialised in the main method. The program in part (iii) however prints out 2, as x is a global variable and although it is initially assigned with the value 3, after execution of method G it will be assigned with the value 2. The program in part (iv) declares two arrays of ints, x and y. All components of array y are initialised to 17 and the ith component of array x is initialised to i. This latter array initialisation however occurs after the assignment statement y=x, which effectively ensures that y refers to the components of x. The println statement will thus output 1, namely the value stored in x[1].

d. Very few candidates offered complete answers to this question though the majority managed to set up an appropriate buffer reader together with a call to the readLine method to read the first line of the file. The general strategy here is to keep reading lines from the file until the end-of-file marker is reached. Whenever the string stored in the line last read starts with the string typed in on the command line (assigned to args[0]) the entire line is printed out. Some candidates made no use of command line arguments but attempted to define a string variable inside the program and assign the value "cat" to it. Some candidates first built a vector containing the strings stored in the lines of the file; this vector was then inspected to see if any of its lines starts with the input string in the command line. This approach is rather inefficient as all functionality required can be implemented into a single while-loop.

Question 6

a. A fairly large number of candidates realised in part (i) that calling method f with argument the string 'MCMXIII' would generate an exception, thus printing out the string 'bad'. The output of the Java program in part (ii) is 26. Following the declarations and initialisations for variables a and b their values are 5 and 26 respectively. Now the assignment statement a=b changes the value of a to 26, leaving b's value unchanged. Almost all candidates answered this correctly, except a few who thought that an assignment a=b changes the values of both a and b. Finally, the assignment statement b=a assigns the value of a to b, so b has the value 26 once again. The value of b is then printed out. The output of the Java program in part (iii) is: 8 7. After the declaration and initialisation of variables a and b the pair (a,b) = (7,8). Then variable c is declared and initialised to the value of b, namely 8. Two successive assignments b=a and a=c then change the value of the triple to (a,b,c) = (8,7,8). The program prints the value of variables a and b, inserting a space in between. Almost every candidate who had an attempt at it answered this question very well.

b. The Java program in part (i) prints the output of calling the method f with the value 6. Note that method f is recursive and calls itself six times before returning a value, so f(6) = 6 +f(5)=6 +(5+f(4))=6 +(5+(4 +f(3))) =6 +(5 +(4+(3 +f(2)))) =6+(5 +(4+ (3 +(2+f(1))))) =6+(5 +(4+(3 +(2 +(1 +f(0)))))=6 +(5+(4 +(3 +(2+(1 + 1))))) = 22. Many candidates attempted part (ii) but few offered complete solutions. The most common omission was a base case to stop the recursion, this can be implemented by using an if-statement, for example, if (n==0 || n==1) return 1, where n is the parameter of the function.

c. Many candidates attempted this question. Assume that we want to check whether strings t and s are anagrams of one another. A general approach would be to first convert t and s into character arrays and check if their lengths are different in which case they are not anagrams. Then for each element of the first array, a search is performed in the second array. If not found, t and s are not anagrams. But if found the element in the second array is replaced by a value which is known not to be in the string. Continuing in this fashion if the end of the first array is reached with all elements being in the second array, then t and s are anagrams. Some candidates followed other approaches e.g. by sorting the strings t and s first, to obtain t1 and s1, then check if t1 and t2 are identical to verify if t, s, are anagrams.