School of Business, Leadership & Enterprise

FdSc Communication Technologies:

Software Engineering

Level: 4

Module: Introduction to Programming (Java)

Assessment: 1 Knowledge and Programming

Module Tutor: Dr Nicholas Caldwell

Weighting in Module:40%

Hand out: 7th March 2014

Hand in: **on or before noon on Monday 19th May 2014**

Assessment Centre, WF 1st floor or SafeAssign.  
Please make sure you obtain and keep a receipt

**+What is required?**

A word processed document that contains the answers to the questions set, including screenshots of program outcomes and code listings. You may find it easier to bundle source code files and the Word document into a compressed archive. In all cases, the Word document and any overarching zip should have only your UCS studentid as its identifier.

**Learning outcomes to be assessed:**

1. to introduce the concept of a computer program, programming languages and program production methods;
2. to introduce generic high-level programming language concepts;
3. to introduce the concepts and structures of structured programming;
4. to introduce the concepts of object-oriented development and programming;

**Graduate Headstart**

C1 Reading, selecting, analyzing and synthesizing information from a range of sources

C2 Producing different types of documents

IT1: Preparing information

**Assessment & Grading Criteria:**

See attached.

**Assessment Brief.**

See attached.

**Assessment Criteria**

To achieve a Pass in this Assignment, the stated Pass criteria must be achieved

|  |  |
| --- | --- |
| **Learning Outcomes Assessed in this Assessment** | **PASS criteria** |
| 1. Demonstrate knowledge and understanding of computer program, programming languages and program production methods; | In order to be awarded a pass grade (P− or above) for this assignment you must meet the following criteria.   * You must score a minimum of 40% (40 marks) for the 28 questions * You must submit your work: * anonymously (i.e. you must identify yourself using only your student number); * word processed; * with the answers to all questions appropriately identified. |
| 2. Select appropriate structured, object-oriented high-level programming and formulate effective software solutions |
| 1. Employ structured and high-level programming constructs to implement effective software solutions |
| 3. Employ object-oriented programming constructs to implement effective software solutions |

Grading criteria follow…….

To achieve a higher grade it is the quality of work that will be considered, rather than the amount of work done, and will be assessed against the given criteria:

**Generic Grading criteria for Level 4**

|  |  |
| --- | --- |
| **Good Pass** | In order to be awarded a grade of “Good Pass” (G−, G= or G+)  your score for the 20 questions must greater than or equal to 50% (50 marks) and less than 60% (50 marks) |
| **Merit** | In order to be awarded a grade of “Merit” (M−, M= or M+)  your score for the 20 questions must greater than or equal to 60% (60 marks) and less than 70% (70 marks) |
| **Distinction** | In order to be awarded a grade of “Distinction” (D−, D= or D+)  your score for the 20 questions must greater than or equal to 70% (70 marks) |

**Assignment Brief**

Study the lecture slide sets, the example sheets, the book “Java: How to Program” by Deitel & Deitel (Ebook), and any other Java documentation that you personally find helpful in order to answer the questions posed.

Submit word processed answers to the questions, making sure that each answer is appropriately identified and properly corresponds to the question set. You should include source code listings (either included in the Word document or as distinct carefully named .java files), screenshots showing program outcomes and any diagrams such as flowcharts.

Modify the Word document by filling in each answer, appropriately identified as the answer, directly below its corresponding question. Rename the file to have your student userid in its filename.

Assignment Questions

1. (2 marks) Explain how Java compilation differs from standard high-level compilation

**Standard high-level compilation, for example what happens in a C compiler, involves using a compiler program to translate a text file written in the high-level programming language and outputting binary machine codes that will have the a specific processor execute the logical equivalent of the high-level program.**

**Java compilation involves an extra step, where the Java code is compiled to a special bytecode that will be executed on a virtual machine, not a specific processor. The Java Virtual Machine then runs this byte code, translating the bytecode into machine codes for the local processor as it goes along.**

1. (2 marks) What is the output produced by the following lines of program code?

char x, y;   
x = ′y′;   
System.out.println(x);   
y = ′z′;   
System.out.println(y);

x = y;   
System.out.println(x);

**Output:**

**y**

**z**

**z**

1. (2 marks) What is the output produced by the following and why?

System.out.println(″1 + 3 = ″ + (1 + 3));   
System.out.println(″1 + 3 = ″ + 1 + 3);

**Output:**

**1 + 3 = 4**

**1 + 3 = 13**

**Java will convert integers to strings before adding them to other strings, but brackets are executed before any other operation. In the first output the brackets dictate that the two integers 1 and 3 must be added together to get 4. Then 4 is added to the string “1 + 3 = ” to give the output above.**

**In the second output java proceeds with addition from left to right, as is the default mode when no brackets are present. In this case a string must be added to an int, giving “1 + 3 = 1”, then that string must be added to another int (3), giving the output above.**

1. (3 marks) Write a complete Java program that reads in a line of keyboard input containing three values of type float, separated by one or more spaces, and outputs them as well as their average.

**import java.util.Scanner;**

**public class ThreeFloats {**

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

**Scanner input = new Scanner(System.*in*);**

**String floats\_as\_string = "";**

**while (floats\_as\_string.length() < 1) {**

**System.*out*.println("Please enter three numbers:");**

**floats\_as\_string = input.nextLine();**

**}**

**input.close();**

**String[] strings = floats\_as\_string.split(" +");**

**System.*out*.println("You input:");**

**float total = 0;**

**for (String string : strings) {**

**System.*out*.println(string);**

**float number = Float.*parseFloat*(string);**

**total += number;**

**}**

**float average = total/strings.length;**

**System.*out*.printf("Average: %.4f\n", average);**

**}**

**}**

1. (3 marks) What output is produced by the following code for key having a value of 1, 4, 5, and 7 and what is the bug in the code?

1 int key = 1;   
2 switch (key \* 1) {   
3 case 2:   
4 System.out.println(“Binary”);   
5 break;   
6 case 8:   
7 System.out.println(“Octal”);   
8 break;   
9 case 10:   
10 System.out.println(“Denary”);   
11 case 16:   
12 System.out.println(“Hexadecimal”);   
13 break;   
14 default:

15 System.out.println(“Confused”);

16 }

**Bug 1: ‘key’ is multiplied by 1 on line 1, which will have no effect. ‘key’ should be multiplied by 2.**

**Bug 2: the ‘case 10’ is missing a ‘break’ statement (line 9-10). This means that if this case is activated, “Denary” will be printed, then “Hexadecimal”, then the case 16 ‘break’ will exit the case statement.**

**A ‘break;’ should be added on a new line just after line 10.**

**Assuming these are fixed, the output will be as follows:**

**Output given 1:**

**Binary**

**Output given 4:**

**Octal**

**Output given 5:**

**Denary**

**Output given 7:**

**Confused**

1. (2 marks) Suppose elementA and elementB are two variables that have been given values. How would you test whether they are equal when they are of type double? How would you test whether they are equal when they are of type String? Explain any difference in your test.

**For doubles: elementA == element**

**For strings: elementA.equals(elementB)**

**Comparing using ‘==’ compares whether two variables are of equal value. In the case of doubles, this will compare their values for equality, because doubles are a primitive type in Java. Strings are objects in Java, so using ‘==’ on string variables will check if the variables point to the same object. Two strings that have the same content will still be stored as separate objects. So ‘==’ will not compare the content of the strings. To do that, we call the ‘equals’ method on one of the strings, passing the second string as a argument. This will compare the contents of the strings.**

1. (3 marks) Determine the value of each of the following Boolean expression, assuming that the value of the variable counter is 0 and the value of the variable sentinel is 20. (Give your answers as one of the values true or false).

counter is 0

sentinel is 20

* 1. (counter == 0) && (sentinel < 20)

True && False

**False**

* 1. counter == 0 && sentinel < 21

True && True

**True**

* 1. (sentinel > 20) || (counter < 5)

False || True

**True**

* 1. !(counter == 7)

!False

**True**

* 1. ((sentinel/counter) > 5) || (sentinel < 30)

**Runtime Error – divide by 0**

* 1. (sentinel < 20) | ((sentinel/counter) > 5)

**Runtime Error – divide by 0**

1. (5 marks) Write code that sums the values 1 to 10 and outputs the answer. Create equivalent solutions using for loops, while loops and do while loops.

For loop:

**int** total = 0;

**for** (**int** i = 0; i <= 10; i++) {

total += i;

}

System.*out*.println(total);

While loop:

**int** counter = 0;

**int** total = 0;

**while** (counter <= 10) {

total += counter;

counter += 1;

}

System.*out*.println(total);

Do-While loop:

**int** counter = 0;

**int** total = 0;

**do** {

total += counter;

counter += 1;

} **while** (counter <= 0);

System.*out*.println(total);

1. (4 marks) In the array declaration

Double[] reading = new Double[100];

what is

1. the array name? **reading**

b. the base type? **double**

c. the length of the array? **100**

d. the range of values an index accessing this array can have? **0-99**

e.. Is it legal to write reading[100] = 79.45? (Explain your answer)

**Writing this line will cause a ArrayIndexOutOfBoundsException as the 100th index in this array does not exist. The array contains 100 items, but it is indexed from 0 so the last index will be 99.**

1. (2 marks) Predict the output from the following:

Double a[] = new Double[5];

int i;

for (i = 0; i < 5; i++)

a[i] = 2.0 \* i;

System.out.println(a[i – 1]);

**8.0**

What would be the output if the last line were changed to

System.out.println(a[i]); ?

**An ArrayIndexOutOfBoundsException would be thrown, because ‘i’ would have come out of the loop as 5, having been incremented as part of the ‘for’ loop. 5 is an invalid index for a 0 indexed array of 5 items.**

1. (2 marks) Write an assertion check that checks to see that the value of the variable monitor is greater than or equal to the value of the variable threshold. Both variables are of type float.

**monitor >= threshold**

1. (3 marks) Explain the differences between a class method and an instance method.

**An instance method has access to the instance variables and methods attached to the instance from which it is called. An instance method also has access to any class variables or methods its class has. Methods are defined as instance methods by default in Java.**

**A class method only has access to the class variables for a particular class. It does not have access to variables stored in instances of the class. In Java, class methods and variables are declared using the ‘static’ modifier.**

1. (4 marks) Explain the differences between public, private and protected access modifiers when applied to variables.

**A public variable is accessible from outside the enclosing class.**

**A private variable is accessible only from within the enclosing class.**

**A protected variable has similar accessibility to public variable, but access is restricted to objects within the same package as the enclosing class.**

1. (3 marks) What is the signature of each of the following method headings?

public void lookupName(String name)

public double calculateCommission(double sales, double commissionRate)

public boolean exceededThreshold(double commission)

public int shares(float commission, String jobrole, int duration)

public float shares(float commission, String jobrole, int duration)

**lookupName(String name)**

**calculateCommission(double sales, double commissionRate)**

**exceededThreshold(double commission)**

**shares(float commission, String jobrole, int duration)**

**shares(float commission, String jobrole, int duration)**

1. (4 mark) Explain the similarities and differences between classes and interfaces.

**A class defines an object type. It consists of a set of variables and methods. Classes can be instantiated, creating objects that are new instances of that class. Instances can assign distinct values to the variables of their class, and can have all the class’s non-static methods called on them.**

**Interfaces define a set of abstract methods. They contain no implementation of these methods, just the types of parameter the methods accept, and the return type of those methods. Classes can ‘implement’ interfaces. This means they have one method definition for each of the abstract method definitions set by the interface.**

**The main difference between a class definition and an interface definition is a class can be instantiated, whereas an interface can only specify aspects of a class. Class definitions and interface definitions are similar because they both define a set of methods that can be called on an object, and can also define static, final variables for a given class. Like interfaces, classes can also contain abstract methods. If a class contains at least one abstract method, the whole class is declared abstract. An abstract class cannot be instantiated. It can be inherited from by other classes, who in turn will become abstract unless they define the abstract methods.**

1. (3 marks)

public class Book

{ private String title;

private int code;

private String author;

public Book (String t, int c, String a)

{

**<Details not shown>**

}

public Book()

{

**<Details not shown>**

}

public void getDetails()

{

**<Details not shown>**

}

}

Which of the following are legal in a program that uses this class and if not, why not? Assume that each line of code follows in order.

Book Javain21Days = new Citizen(“Java in 21 Days”, 17, “Roger Cadenhead”);

**Illegal – the Citizen class is not defined.**

Book ThirtyNineSteps = new AnotherClass(“Thirty-Nine Steps”,78.9);

**Illegal – the AnotherClass class is not defined.**

Book b = new Book();

**Legal**

b.getDetails();

**Legal**

Book TopBestSeller;

**Legal, but TopBestSeller has not been initialised.**

TopBestSeller.changeDetails();

**Illegal – this method is not defined for the Book class, also TopSeller has not been assigned a Book instance.**

1. (3 marks) Suppose Novel is a derived class of a class called Book and further suppose that the class Book has instance variables Title and Author and a method checkLoanStatus. Will an object of the class Novel have instance variable Title and Author? Explain your answer.

**Yes. All variables will be inherited by children of the Book class.**

Will the Novel class also have a method named checkLoanStatus and if so, how may it differ between the Novel and Book classes. Explain your answer.

**Yes. All instance methods will be inherited by children of the Book class. ‘checkLoanStatus’ may be overridden by the Novel class. In this case the Novel class’s ‘checkLoanStatus’ may produce a different result or have a different implementation than the one defined by Book. Different behaviour may be produced if ‘checkLoanStatus’ uses instance variables that differ for a particular instance of Novel and a particular instance of Book.**

1. (15 marks) In the Week 5 exercises, you developed code to encrypt text using the Caesar cipher. Now design using a flowchart or pseudocode an algorithm to decrypt a ciphertext which has been encoded by a Caesar cipher of unknown offset. Implement this as a complete program – you may assume that the ciphertext is available to the program as a String of multiple uppercase 5-character blocks separated by spaces.

**Determine the most common letter**

**Make an array of 26 zeros**

**For each letter in message:**

**Get alphabetical position of the letter**

**Add one to the number in the array at the position determined by the alphabetical position**

**Find the alphabetical position with the highest number in the array.**

**Get the character code from the alphabetical position.**

**offset = (character code of most common letter) – (character code of ‘E’)**

**For each letter in message:**

**Get alphabetical position of the letter**

**Add offset to alphabetical position of the letter**

**Use modulus to ‘wrap’ alphabetical position**

**If alphabetical position is negative add 26**

**Convert alphabetical position to character code**

**Return message**

1. (15 marks) Design (using flowcharts or pseudocode) and implement classes and methods which perform matrix multiplication on square matrices of size 2, 3 and 4. Matrix cells may be integer or double-valued

**Arrays of doubles:**

**Let C = a blank array of the same dimensions as A**

**for each column in A:**

**for each row in B:**

**result = multiply all the cells in As columns with all the rows in B**

**store result in the equivalent cell of C**

**return C**

**Arrays of integers:**

**Convert each integer to a double.**

**Perform the above operations on this array of doubles.**

**Convert each double to an integer.**

1. (20 marks) Design and implement (using flowcharts or pseudocode) a suitable (set of) class(es) and methods to implement a birthday and address book.

Each entry in the book should contain a first name, a surname, a valid email address, and a birthday.

A valid email address should have one or more letters or digits followed by an @ character followed by one or more letters or digits – there may be one or more periods between letters or digits.

Create a constructor to create instances of the book entries

Accessor methods to retrieve and display names, email addresses, and birthdays using family names and email addresses as search keys

Mutator methods to change names, email addresses, and birthdays using family names and email addresses as search keys

A search method to test if there is an entry matching a given birthday

A sorting routine that retrieves all the entries ordered by date (1st Jan first to 31st Dec).

AddressBook

ArrayList addresses

searchByDateOfBirth()

sort()

getAddressesFromEmailAddress()

getAddressesFromSurname()

searchByDateOfBirth(Date target)

for each address in addresses:

get date of birth from address

if date of birth matches the target date of birth

add address to result list

return result list

dateComparator

compare()

EmailAddressInvalidException

Address

Constructor returns blank address

String firstName

String surname

String emailAddress

Date dateOfBirth

Normal getters and setters for each, except for setting email address:

setEmailAddress()

getAddressesFromEmailAddress(target)

for each address in addresses:

get email address from address

if email address == target

add address to result list

return result list

getAddressesFromSurname() similar but by surname

setEmailAddress(newEmail)

if newEmail doesn’t match the email regular expression:

raise EmailAddressInvalidException

else:

emailAddress = newEmail

compare(d1, d2)

compares only the month and the day of the date

returns 1 if d1 > d2

returns 0 if d1 == d2

returns -1 if d1 < d2

sort()

call Collection.sort on addresses, passing in the an instance of dateComparator

return sorted list