YOUR SCHOOL BADGE HERE

YOUR SCHOOL NAME HERE

JUNE EXAMINATION – 2015

INFORMATION TECHNOLOGY

PAPER 1 – PRACTICAL

TIME: 3 HOURS MARKS: 150

This paper consists of 11 pages, and there are 3 NetBeans projects in your exam folder.

Make sure that you have a complete set.

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into THREE sections. Candidates must answer ALL THREE sections.
2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.
3. Make sure that you answer the questions according to the specifications that are given in each question. Marks will only be awarded according to the set of requirements.
4. Answer only what is asked for in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.
5. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.
6. **Routines such as search, sort and selection** must be developed from first principles. You may not use the built-in features of the programming language for any of these routines,
7. You must **save your work regularly** on the disk you have been given, or the disk space allocated to you for the examination.
8. Make sure that your **name and surname** appears as a comment in every program that you code.
9. At the end of the examination session, you must hand in all your work saved. Ensure that all the **files can be read**.
10. The files you need to complete this question paper have been given to you on the disk space allocated to you in the form of a password-protected executable file: **JavaDataENG.exe**

    Do the following:
    - Double click on the file.
    - Click on the extract button.
    - Enter the following password: **Go@180Km%hr**

    Once the file has been extracted, the following list of files will be available in the folder JavaDataENG:

    Java (NetBeans) files

    Question1
        Question1.java
        Question1.form
        Lightning-McQueen.png
        PlatePic.png

    Question2
        Question2.java
        Question2.form
        Salesperson.java
        CarSales.txt
        Lightning-McQueen.png

    Question3
        Question3.java
        Question3.form
        Lightning-McQueen.png

SCENARIO

Lightning McQueen Inc.

A multi-franchise car dealership encourages their clients to have personalised number plates. In Question 1 you are required to write a program to generate these number plates.

The company has been operating for two months and all 10 sales people could sell both new and used cars. The company now wants to divide the group into a New Car Sales Team and a Used Car Sales Team. In Question 2 you will assist them by writing a program that will output the new car sales team.

Buying used cars can be risky so the company takes special care that all vehicle identification numbers, VIN, are valid. Part of the solution to Question 3 will assist them with validating VINs.

SECTION A

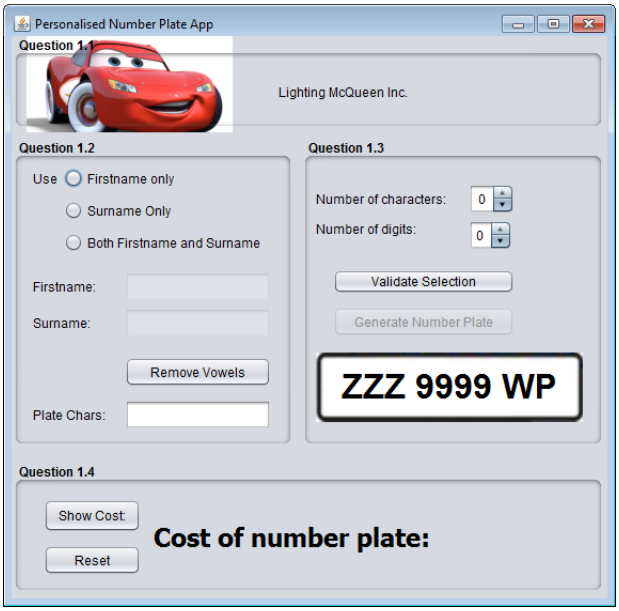
QUESTION 1: GENERAL PROGRAMMING SKILLS

INSTRUCTIONS

- The project Question 1 is provided in the JavaDataEng folder.
- Open the incomplete class called Question1.java in the package question1package.
- Add your name and surname as a comment in the first line of the class Question1.java

Do the following:

- Compile and execute the program. The interface displays four sections labelled Question 1.1 to Question 1.4.



- Complete the code for each section as described in QUESTION 1.1 to QUESTION 1.4 below.

1.1 When the program runs the label titled Question 1.1 should have the following appearance:

- No Border
- Background colour: black
- Font colour: white
- Font size: 24

5

1.2 In this section a string is created based on the name and/or surname entered.The string has no vowels and is presented in uppercase letters.

1.2.1 Write code in the action event handler of each radio button to enable the text fields indicated in the radio button's label.

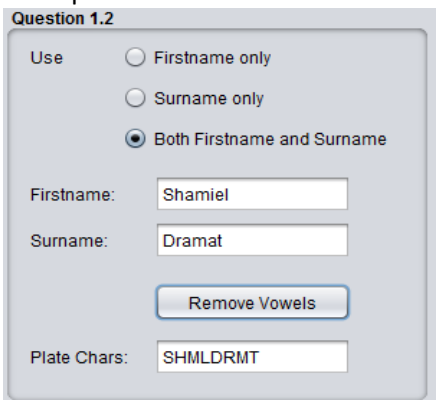
Radio Button Selected	Firstname text field	Surname text field
Firstname only	Enabled	Disabled
Surname only	Disabled	Enabled
Both Firstname and Surname	Enabled	Enabled

4

1.2.2 Code the Remove Vowels button to:

- Read the text entered in the Firstname and/or surname text fields If both firstname and surname are provided then the surname must be concatenated to the firstname
- Remove the vowels
- Convert all to uppercase letters
- In 1.3 below this string will be called the **no-vowel string**

Example:



7

1.3 The user can now choose how many letters of **no-vowel String** should appear in the number plate. Counting starts from the first letter. The user can also choose how many digits should be in the number plate. The sum of letters and digits must **not exceed 7**. A space and the WP is not counted.

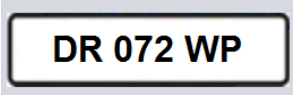
1.3.1 Write code for the Validate Selection button that will:

- Check if the number of letters is less than or equal to the number of letters in the **no-vowel string**. Provide an error message in a message box if the above test fails.
- Now check if the sum of the selected letters and digits is less than or equal to seven. Provide a suitable error message if the test fails.
- The method must exit on error, i.e. if the first test fail the second test should not be performed.
- Only if both tests pass should the Generate Number Plate button be enabled

10

1.3.2 Code the Generate Number Plate button as follows:

- Obtain the required number of letters from the start of the **no-vowel string**
- Randomly generate each digit of the number section. Digits 0 to 9 are allowed in all places in the number.
- Concatenate the letters, digits and WP with spaces inbetween
- Display the number plate in the number plate text field provided



8

1.4 Personalised number plates are charged as follows:

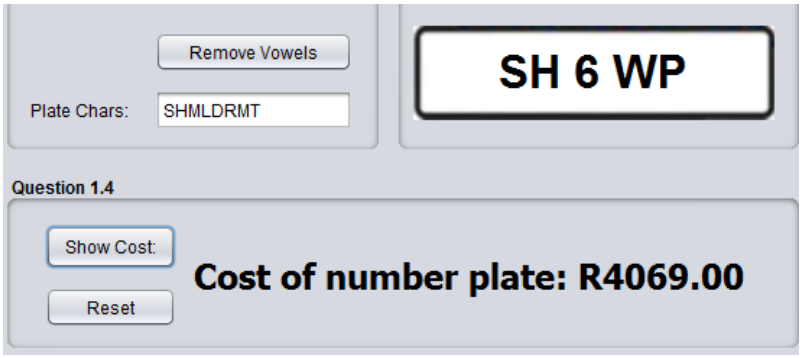
- Basic registration fee R 69.00
- 1 letter or digit number plate R10000.00

  Examples: A WP 7 WP
- 2 letters, 2 digits or 1 of each R 6000.00

  Examples: AB WP 66 WPA 6 WP
- 3 letters, 3 digits or a combination R 4000.00

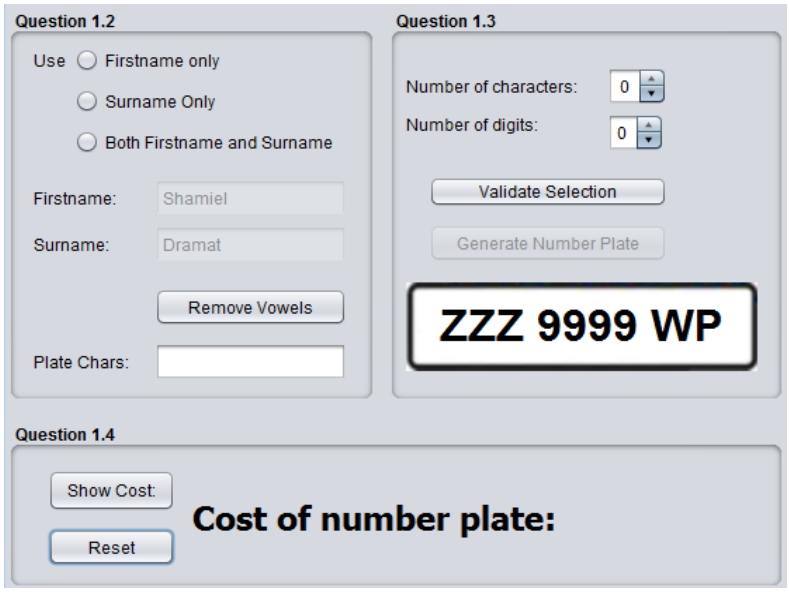
  Examples: ABC WP 666 WPAB 6 WP
- 4 to 7 any combination of letters and/or digits R 1750.00

1.4.1 Code the button Show Cost to determine the cost of the generated number plate in 1.3. Note WP and spaces do not count and therefore do not contribute to the cost. Append the cost to the text "Cost of Number Plate:"



Example: Cost of a 2-letter1-digit number plate 7

1.4.2 Provide code for the Reset button that will reset the GUI as indicated in the picture.



7

Save your program **SECTION A TOTAL** **48**

SECTION B

QUESTION 2: OBJECT-ORIENTATED PROGRAMMING

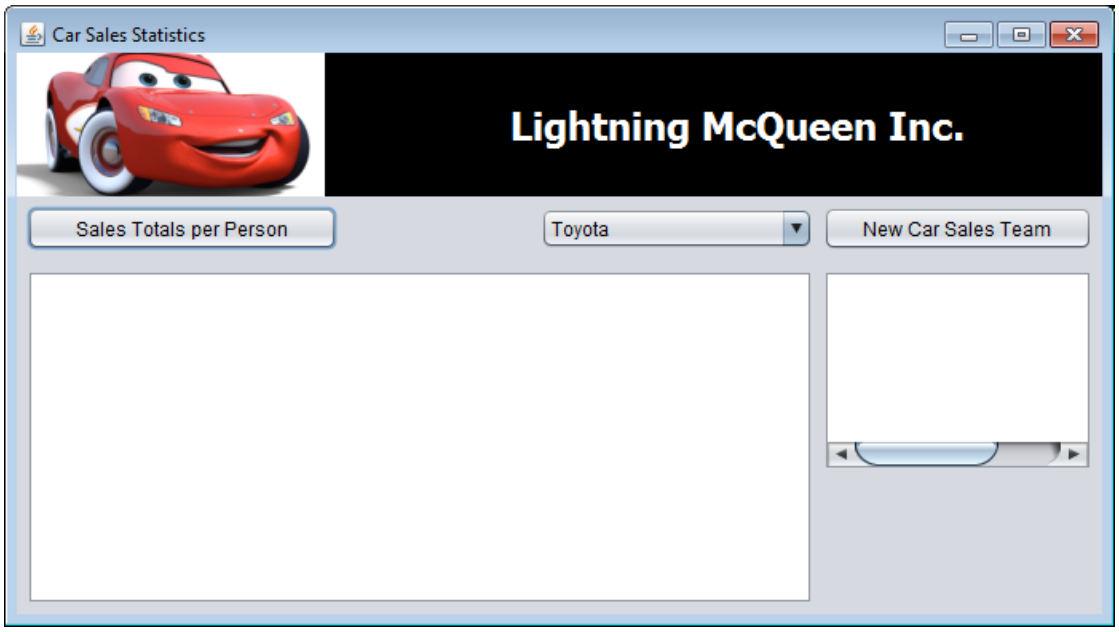
In order to identify the new car sales team the company keeps sales records in a comma-delimited text file for this purpose. Each record consists of the salesperson's name, make of car and the year manufactured. Your program will analyse this data and output the four sales people that form the New Car Sales Team. The rest will be on the Used Car Sales Team.

INSTRUCTIONS:

- The project Question 2 is provided in the NetBeans folder.
- Open the incomplete class files called Question2.java and Salesperson.java in the package question2package.
- Add your name and surname as a comment in the first line of the classes Question2.java and Salesperson.java.

Do the following:

- Compile and execute the program. Currently the program has no functionality.



- The GUI above provides an output to list the sales per person for the 2 months as well as the number of cars of the selected make sold.
- In the smaller output area the new car sales team will be revealed.
- Complete the code as described in QUESTION 2.1 and QUESTION 2.2 to add functionality to the program.

2.1 Complete the code in the Salesperson class as described

2.1.1 Write code to add the following attributes (fields) to the class: 2

Description	**Names of attributes**
Name of salesperson	name
Number of used cars sold (all makes)	usedCarsSold
Number of new cars sold (all makes)	newCarsSold
Make of all the cars sold (#delimited text)	makeLog
Make to report on as selected in the combo box	make

2.1.2 Write code to create a constructor which will receive the following parameters:

Name of the salesperson
Make to report on

Initialise the relevant attributes using the parameter values.
Initialise the makeLog attribute to an empty string. 3

2.1.3 Write an accessor method for attributes name and newCarsSold. 2

2.1.4 The methods called incrementUsedCarTotal and incrementNewCarTotal have been provided. Uncomment the methods in order to make them functional. 1

2.1.5 The method updateLog has been provided. It has a string parameter that holds the make of the car to be added to the log. Write code to update the makeLog string by concatenating the parameter value to it. The make values must be kept separate using the #-character.

Example: Toyota#Kia#Toyota#Volkswagen 2

2.1.6 The method called carsOfMake has been provided. It should return the number of cars of the type set in the make attribute found in the makeLog string.

If the make attribute is set to Toyota then it must return the number of Toyotas sold by the salesperson. This information is stored in the makeLog string.

Example: For the makeLog string value: Toyota#Kia#Toyota#Volkswagen
and Make value: Toyota
the method returns 2.
Indicating that the sales person sold 2 Toyotas.

Write code to implement this functionality. 6

2.1.7 Write code to create a toString method to return a string formatted as follows

<salesperson><total cars ><total new ><total used ><total cars of make>
Carl 14 5 9 3
in neat columns. 6

2.2 You have been provided with a text file called CarSales.txt that contains the name of salesperson, make of car and year manufactured. The first two lines are given below:

Carl,Volkswagen,2011
Lindiwe,Hyundai,2012
…

You have also been provided with a string array containing the names of the sales people. This array is called salesperson.

2.2.1 Create an array to hold the values of the new cars sold by each salesperson. This array should run parallel to the given salesperson array and should be accessible to both the Sales Totals per Person button and the New Car Sales Team button. 3

2.2.2 Provide code for the Sales Totals per Person button that will:

- Insert the headings in the text area below the Sales Totals per Person button
  - In neat columns, see headings in the sample output below
  - This process must also clear any text in the text area
- For each person in the given salesperson array
  - Open the file CarSales.txt for reading
  - Report if the file does not exist and if so exit the program
  - Create a Salesperson object
  - If the line read from the file is for the current salesperson object, call the appropriate object methods to
    - increment the person's used or new car sales total: 2015 = new car
    - update the make log
  - Display the salesperson's totals in the text area
  - Insert the salesperson's new car sales total in the array created in 2.2.1.
  - close the file

25

2.2.3 The four sales people who sold the most new cars for the two-month period will form the New Car Sales Team.

If you did not manage to populate the new car sales array in 2.2.2 above then initialise the array with the following totals:

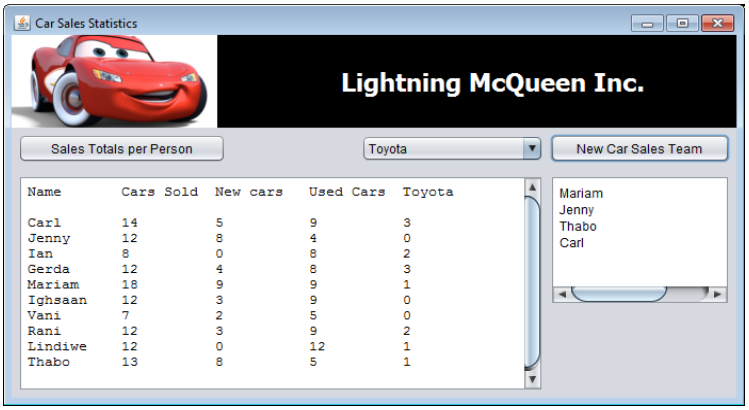
5 ; 8 ; 0 ; 4 ; 9 ; 3 ; 2 ; 3 ; 0 ; 8

Code the New Car Sales Team button to

- Sort the parallel arrays
- List the names of the 4 members of the New Car Sales Team in the text area below the button. See sample output below.

Remember when a swap is required in the one array the other one must also be swapped.

You may NOT use any built-in sorting methods.



8

2.2.4 Provide a suitable method call in the combo box event handler that will update the information in the Sales Totals per Person text area when a new make is selected. 1

Save your program **SECTION B TOTAL** **59**

SECTION C

QUESTION 3: PROBLEM-SOLVING PROGRAMMING

A Vehicle Identification Number (VIN) is a unique 17 character code assign to a vehicle by its manufacturer. The first three characters is the world manufacturer identifier (WMI), the next 6 characters is the vehicle description section (VDS) and the last 8 characters is the vehicle indicator section (VIS).

INSTRUCTIONS:

- The project Question3 is provided in the NetBeans folder.
- Open the incomplete file called Question3.java in the package question3package.
- Add your name and surname as a comment in the first line of the class Question3.java.
- You may design more classes if your solution requires it.

VIN number specification

VIN characters

- Alphabet plus digits
- Letters I, O and Q are not used at all
- For the year character letters U and Z as well as digit 0 are also excluded
- VIN Example

WDBRF56H47F506498.

The Year Character

- The 10th character in the VIN number represents the model year. In the exampleabove that would be "7". I, O, Q, U, Z and zero are not allowed. Year characters repeat every 30 years. See **Table A** below.

The Check Digit Calculation

- The 9th character in the VIN number represents the check digit(in the example that would be "4") and it is calculated as follows:
  - Convert all letters in the VIN to a numerical value according to Table B below. Numbers stay as they are.
  - Look up a weighting factor for each position in Table C below
  - Multiply the numbers and the numerical values of the letters by their assigned weight factor
  - Add the products
  - Divide the sum of products by 11. The remainder is the check digit. If the remainder is 10, the check digit is X.

In the example above: Year Character = 7 Year Manufactured = 2007(Model Year, Table A)

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
VIN	W	D	B	R	F	5	6	H	**4**	7	F	5	0	6	4	9	8
Numerical value	6	4	2	9	6	5	6	8	4	7	6	5	0	6	4	9	8
Weighting	8	7	6	5	4	3	2	10	0	9	8	7	6	5	4	3	2
Product	48	28	12	45	24	15	12	80	0	63	48	35	0	30	16	27	16
Sum of products	499		Remainder of sum of products/11								**4**	= check digit in position 9					

Table A: Model Year Character for the 30 years 1986 to 2015

1986→	G	H	J	K	L	M	N	P	R	S	T	V	W	X	Y	**←2000**
2001 →	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	**←2015**

Table B: Numerical values of Letter characters in the VIN number

Letters	Numerical value	Letters	Numerical value	Letters	Numerical value
A; J	1	D; M; U	4	G; P; X	7
B; K; S	2	E; N; V	5	H; Y	8
C; L; T	3	F; W	6	R; Z	9

Table C: VIN number position weighting factor

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Weighting	8	7	6	5	4	3	2	10	0	9	8	7	6	5	4	3	2

Do the following:

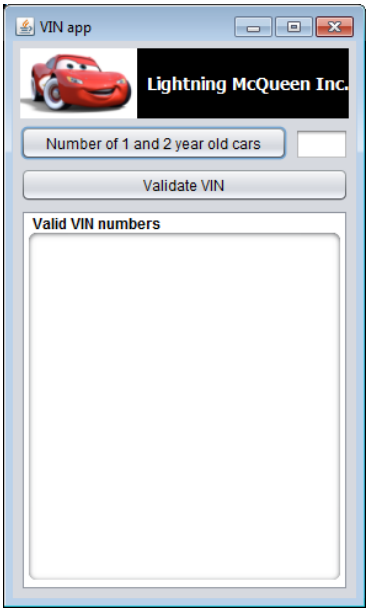
- Compile and execute the program. Currently the program has no functionality.
- Complete the code as described in QUESTION 3.1 to QUESTION 3.3 to add functionality to the program.

3.1 Provide code for the **Number of 1 and 2 year old cars**button to report on how many one-year-old and two-year-old cars are in the array i.e. cars manufactured in 2014 and 2013. Place the answer in the non-editable text field to the right of the button. 7

3.2 Write a method called getNumericalValue which will receive a VIN character as parameter and return the character's numerical value as in Table B. 7

3.3 Code the **Validate VIN** button to do the following: 29

- Validate the VIN number by testing if the check digit is correct. Make use of the method written in 3.2
  - If it is correct then display the VIN number in the text area labelled Valid VIN numbers.
  - The incorrect VIN numbers must be written to a text file called **HotCars.txt**.



Save your program **SECTION C TOTAL** **43**