YOUR SCHOOL BADGE HERE

YOUR SCHOOL NAME HERE

SEPTEMBER EXAMINATION – 2013

INFORMATION TECHNOLOGY

PAPER 1 – PRACTICAL

TIME: 3 HOURS MARKS: 120

This paper consists of 11 pages.

INSTRUCTIONS AND INFORMATION

1. 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.

2. You require the files listed below in order to answer the questions. They are EITHER on a stiffy disk OR CD issued to you OR the invigilator/teacher will tell you where to find them on the hard drive of the workstation you are using OR in a network folder.

Question1_java:	**Question1_NetBeans**
tblClients.txt	tblClients.txt
tblPolicies.txt	tblPolicies.txt
Insurance.mdb	Question1 NetBeans Project:
DisplayClass.java	Insurance.mdb
TestQuestionOne.java	DisplayClass.java
	TestQuestionOne.java
Question2_java	**Question2_NetBeans**
Annuity.java	Question2NetBeans Project
AnnuityCalculator.java	Annuity.java
	AnnuityCalculator.java
Question3_java	**Question3_NetBeans**
Competition.txt	Competition.txt

   If you received your files above on a disk (CD or stiffy), write your name and surname on the label.

3. Type your name and surname as a comment in the first line of each program file that contains your programming code.
4. Your program should always be coded to answer the question in the way it has been formulated. You are not allowed to only copy the given output supplied in the question paper.
5. Read ALL the questions carefully. Do not do more than what the questions require.
6. To help you understand each question better, you have to read the entire question before answering any subquestions.
7. Save your work at regular intervals as a precaution against power failures.
8. There might be a technical interruption that prevents you from writing the examination, such as power failure. When you resume writing the examination, you will be given the time remaining when the interruption began, and an additional 10 minutes.
9. During the examination, you may use the manuals originally supplied with the hardware and software. You may also use the HELP functions of the software and the Java API files. You may NOT use any other resource material.
10. At the end of this examination session, you must hand in the disk or CD with all your work saved on it OR you must make sure that all your work has been saved on the hard drive/network as explained to you by the invigilator/teacher. Ensure that all files can be read.

SCENARIO

Rainy-Day Insurance Brokers, an insurance company, has branches all over the Western Cape. The company was founded by Mr C Cash who runs his business from the Head Office based in Bellville.

The company requires new software to manage clients and policies as well as manage incentive schemes and competitions.

You are required to complete the following THREE questions using the programming language you have studied.

QUESTION 1: PROGRAMMING AND DATABASE

A Microsoft Office Access database named **Insurance.mdb,** two text files (**tblClients.txt** and **tblPolicies.txt**) and an incomplete program are provided in the folder named **Question1_Java** or **Question1_NetBeans.**

The design of the tables in the **Insurance** database and sample data for each table can be found in **ANNEXURE A**.

Do the following:

- Make a backup copy of the **Insurance** database **BEFORE** you start answering QUESTION 1. You will need a copy of the original database to be able to test your program thoroughly.
- Rename the given folder for Question 1 by replacing Java or NetBeans with the first two letters of your name followed by the first two letters of your surname. John Doe's folder would be renamed to Question1_JoDo
- Open the incomplete program for QUESTION 1 called **TestQuestion1.java**.
- Add your name and surname as a comment in the first line of the program file.
- Compile and execute the program. The interface displays seven menu options as indicated in the section labelled **QUESTION 1** in **ANNEXURE B**.

  NOTE:
  - An error message will be displayed if any of the options A – H are selected, due to the incomplete SQL statements.
  - If you experience any problems using the database or connecting to the database, refer to **ANNEXURE C**for troubleshooting hints.
  - If you still experience database problems, you must nevertheless do the SQL code and submit it for marking.
    Marks will only be awarded for the programming code that contains the SQL statements.
- Complete the code for each menu option by formulating an appropriate SQL statement to display the results for each query as described in QUESTIONS 1.1 to 1.8 below.

  NOTE: The code to some input statements and the code to execute the SQL statements and display the results of the queries have already been written as part of the given code.

1.1 **Menu Option A: Display Clients**

Display all the details of the clients stored in the **tblClients** table, sorted firstly by the **Surname** field and secondly by **FirstName** field in **alphabetical order**.

Example of the output of the first five records: (4)

ClientNo	Surname	Firstname	Branch	DateOfBirth	Gender	Race
E599	Abrahams	Zaitoen	Bellville	1975-02-28	F	C
Q809	Black	John	Durbanville	1974-05-19	M	W
A198	Boschoff	Pragtig	George	1971-06-04	F	C
D497	Brady	Bo	Durbanville	1968-03-25	M	W
U812	Buthelezi	Vusi	Milnerton	1980-11-23	M	B
T193	Evans	Marlena	Bellville	1969-04-08	F	W

1.2 **Menu Option B: Display Branches**

Formulate an SQL statement to display the names of all the branches of Rainy-Day Insurance Brokers without duplication.

Example of output of the first five lines: (3)

Example of output:

Branch
Bellville
Cape Town
Claremont
Durbanville
GatesVille

1.3 **Menu Option C: Policies with Premium Update Facility**

Complete the code by formulating an SQL statement to calculate and output the number of policies that have a premium update facility.

The name of the calculated field is **Total**. (5)

Example of output:

Total
20

1.4 **Menu Option D: First Quarterly Policies**

Formulate an SQL statement to display the fields **Surname**, **Firstname** and **PolicyNo** of all clients who have taken out policies in the first quarter of the year 2008 (i.e. January to March). (7)

Firstname	**Surname**	**PolicyNo**
Aron	**Mtshali**	**905500**
Nene	**Mnomiya**	**236985**

1.5 **Menu Option E: Total Current Policy Value**

Complete the code for this option by formulating an SQL statement to calculate and display the total current value of all policies for each Race.

Total Current Policy Value is the name of the calculated field.

NB: All amounts must be neatly displayed, including the currency. (7)

Example of output:

```
Race   Total Current Policy Value
================================
B      R1446956.30
C      R170000.00
I      R457000.00
W      R227000.00
```

1.6 **Menu Option F: Insert New Policy**

Formulate an SQL statement to insert the following record in the **tblPolicies** table.

Policy Number	651017
Policy Type	Life Cover
Client Number	C607
Start Date	2013/08/01
End Date	2023/08/01
Premium Update Facility	YES
Percentage Update	10%
Premium	R 200.00

Output: (5)

```
Records Processed Successfully
```

1.7 **Menu Option G: Remove Matured Policies**

Complete the code for this option by formulating an SQL statement to delete all policies that have matured **before the Year 2013**.

Output: (3)

```
Records Processed Successfully
```

1.8 **Menu Option H: Update Premium**

Formulate an SQL statement to update all monthly premiums by their respective Percentage Update for the current year.

Output**:** (4)

```
Records Processed Successfully
```

- Enter your name and surname as a comment in the first line of the file containing your SQL statements.
- Save your program.

[38]

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

Mr CCash decides to improve the service the sales person gives to clients. They currently use a standard calculator to calculate the various what-if scenarios clients normally request. He needs a future value annuity calculator program for his sales people. The two most common requests from clients are the calculation of future value and monthly premium.

The formula to calculate the future value of an investment if regular premiums are invested at a fixed interest rate for a fixed period is:

$$F = \frac{x[(1+i)^n - 1]}{i}$$

The formula to calculate the regular premiums to be invested at a fixed interest rate for a fixed period to obtain the desired future value is:

$$x = \frac{Fi}{(1+i)^n - 1}$$

Where: F = future value, x= premium, n = period and i= interest rate divided by 100

For the purpose of this examination:
The annuity calculator will only be required to calculate future value and monthly premiums.
Inputs for interest will be per annum and period of investment will be in years.

The files required for this question can be found in the folder named **Question2_Java** or **Question2_NetBeans**. Rename the given folder for Question 2 by replacing Java or NetBeans with the first two letters of your name followed by the first two letters of your surname.

2.1 Open **Annuity.java** and provide code for the following:

2.1.1 Declare the following fields and set the default values for each.

Field	Variable name	Default value
Future value	futureValue	R60000.00
Interest rate	rate	12.0%
Investment period	period	10 years
Monthly premium	premium	R260.83

Choose appropriate data types. 3

2.1.2 Provide a method called **getMonths()** that will return the investment period in months. 3

2.1.3 Provide a method called **getMonthlyRate()** that will return the interest rate per month. 3

For questions 2.1.4 and 2.1.5 code the appropriate formula given above.
Use the methods developed in 2.1.2 and 2.1.3.

2.1.4 Provide code for the **calculateFutureValue()** method that will calculate the future value and set the **futureValue** field. 3

2.1.5 Provide code for the **calculatePremium()** method that will calculate the monthly premium and set the **premium** field. 3

2.1.6 Code a **toString()** method that will format the investment information as indicated in the example below and return the formatted string :

```
Premium:      R382.64 per month
Period:       7 years
Interest:     12.0% per annum
Future Value: R50000.00
```

6

2.1.7 Provide a parameterised constructor that will accept **future value**, **rate** and **period** and set the corresponding field values. 3

2.1.8 Provide a parameterised constructor that will accept **monthly premium**, **rate** and **period** and set the corresponding field values. 3

2.1.9 To ensure that the investment figures fit the given formula, the method in 2.1.4 or 2.1.5 must be called from the appropriate constructor. 3

The purpose of the **AnnuityCalculator** class is to run various client queries and display the investment figures in the format indicated in 2.1.6 as well as sending it to a file called **WhatIf.txt.** All input will be from the keyboard; therefore set up the input code you are most skilled at, be it **BufferedReader**, **Scanner** or **JOptionPane**.

2.2 Open **AnnuityCalculator.java** and provide code for the following:

2.2.1 The **toFile** field of type **PrintWriter** has been declared. In the **AnnuityCalculator ()** constructor

- create an instance of the PrintWriterlinked to **WhatIf.txt**
- call the **runWhatIf ()** method
- provide code to display any exceptions

5

2.2.2 Complete the **runWhatIf()** method by inserting code to

- determine whether future value or monthly premium needs to be calculated
- capture input for the fields required for the calculation.
  e.g. if monthly premium must be calculated then the user must give the future value, interest rate and period.
- create an **Annuity** instance for the calculation
- display the investment figures on the screen as well as send it to the **WhatIf.txt** file. Use the format indicated in 2.1.6
- determine if the client wants another calculation done
- close the print writer

14

- Enter your name and surname as a comment in the first line of both **Annuity.java** and **AnnuityCalculator.java**.
- Save your program.

[49]

QUESTION 3: PROBLEM-SOLVING PROGRAMMING

Rainy-Day Insurance Brokers hosted a Cell Phone SMS (short message service) competition for all its clients. Clients had to SMS in their full names and policy number. The details of all the SMSs received by Rainy-Day Insurance Brokers have been captured and stored in a text file called **Competition.txt** in the following format:

	Full name#Policy number
The first six entries:	Vuyo Madhanga#810694
	Roland Singh#625414
	Sabelo Sibambo#769254
	Vilomoney Naidoo#148563
	Vilomoney Naidoo#920362
	Roland Singh#625414

Some clients have more than one policy. They are therefore allowed to enter more than once (see VilomoneyNaidoo in the example above). But only one entry is allowed per policy. The folders **Question3_Java** and **Question3_NetBeans** contains the text **Competition.txt**. The file also has duplicate entries as some clients did not read the competition rules and thought that the more entries the better their chance of winning.

Do the following:

- Rename the given folder for Question 3 by replacing Java or NetBeans with the first two letters of your name followed by the first two letters of your surname.
- Create a new program/project/application
- Add your name and surname as a comment in the first line of the program file(s) you have created that will contain code. If your solution consists of more than one class file, make sure to add your name and surname as a comment in all the class files.
- Save the program file(s) by using the question number as part of the filename/project name in the renamed folder for QUESTION 3.

The program must do the following:

3.1 Use any suitable data structure to hold the data required to solve this problem.
e.g. 2D arrays, parallel arrays, Lists, classes etc. (1)

3.2 Test if the file **Competition.txt** exists, if not the program must terminate.
Read the file and store the data in the data structure that you declared in 3.1. (5)

3.3 Remove all duplicate entries (see Roland Singh in the example above). (6)

3.4 Calculate points for each entry as follows:

- 5 points are awarded for every vowel in the full name and 2 points are awarded for every consonant in the full name. No points for spaces. Calculate the points for the full name.
- The value of each digit in the policy number is added to the points calculated for the full name.

Example: VuyoMadhanga 810694
Name points: 2+5+2+5+0+2+5+2+2+2+5+2+2+5 = 39
Number points: 8+1+0+6+9+4 = 28
Total points: 39 + 28 = 67 (7)

3.5 Determine the clients with the highest points and award a Mr C Cash prize as follows:

Position	Prize Money
1st	R5000.00
2nd	R4000.00
3rd	R3000.00
4th	R2000.00
5th	R1000.00

(5)

3.6 Resolve tied positions by adding the prize money for those positions and sharing it equally.
Display only the prize winners as follows:

Name	Points	Prize Money
VilomoneyNaidoo	83	R5000.00
VilomoneyNaidoo	81	R4000.00
Patience Mthembu	78	R3000.00
SabeloSibambo	77	R1000.00
Vusi Buthelezi	77	R1000.00
Zaitoen Abrahams	77	R1000.00

(9)

- Enter your name and surname as a comment in the first line of all the files containing code to solve this problem.
- Save your program.

[33]

ANNEXURE A:

tblClients: Table Structure

Field Name	Data Type
ClientNo	Text
Surname	Text
Firstname	Text
Branch	Text
DateOfBirth	Date/Time
Gender	Text
Race	Text

tblPolicies: Table Structure

Field Name	Data Type
PolicyNo	Text
PolicyType	Text
ClientNo	Text
StartDate	Date/Time
EndDate	Date/Time
GuaranteedAmt	Currency
PremiumUpdateFacility	Yes/No
PercentageUpdate	Currency
CurrentPolicyValue	Currency
Premium	Currency

tblClients: Sample Data

ClientNo	Surname	Firstname	Branch	DateOfBirth	Gender	Race
A198	Boschoff	Pragtig	George	1971/06/04	F	C
A301	Swardt	Annerly Peters	Bellville	1979/12/25	F	W
A324	Sibambo	Sne	Langa	1967/04/26	F	B
A502	Loubser	Shirlene	Cape Town	1979/09/09	F	W
A619	Singh	Nitesh	GatesVille	1965/12/22	M	I

tblPolicies: Sample Data

PolicyNo	PolicyType	ClientNo	StartDate	EndDate	GuaranteedAn	PremiumU	Percen	CurrentPolicyV	Premium
10245	Endowment	E896	2005/07/01	2015/07/01	R 60 000.00	☐	0%	R 25 000.00	R 250.00
148563	Life Cover	A882	1995/01/01	2014/01/01	R 60 000.00	☐	10%	R 70 000.00	R 350.00
185698	Endowment	A911	2002/10/01	2012/10/01	R 5 000.00	☑	10%	R 20 000.00	R 200.00
196524	Life Cover	T193	2006/08/01	2021/08/01	R 36 000.00	☑	10%	R 42 000.00	R 300.00
200035	Life Cover	A502	2004/03/12	2014/03/12	R 40 000.00	☑	20%	R 140 000.00	R 210.00

ANNEXURE B: USER INTERFACE FOR QUESTION 1

When you execute the program, the inteface below will be displayed.

MENU

A - Display Clients
B - Display Branches
C - Policies with Premium Update Facility
D - First Quarterly Policies
E - Total Current Policy Value
F - Insert New Policy
G - Remove Matured Policies
H - Update Premiums
Q - QUIT

Your Choice -

ANNEXURE C: TROUBLSHOOTING DATABASE PROBLEMS

C1 If you cannot use the given database:

- Create your own database with the name **Insurance** that includes a table named **tblClients** and another table named **tblPolicies** in the same folder as your program for QUESTION 1.
- Inport the two text files (**tblClients.txt** and **tblPolicies.txt**) to use as data for the different tables.

C2 If your program cannot establish connectivity with the database:

- Make sure that the database file **Insurance** is in the same foldere as your program for QUESTION 1. If this is not the case, copy the database file into the same folder as your program.

C3 If you cannot establish connectivity with the given database with the given program files, use the following source code to ensure database connectivity:

```
try {   Class.forName (”sun.jdbc.odbc.JdbcOdbcDriver”);
        String filename = “Insurance.mdb”;
        String database = “jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)}; DBQ=”;
              Database += filename.trim() + “;DriverID=22;READONLY=true}”;
        Conn = DriverManager.getConnection(database, “”, “”);
}catch(Exception e){
        System.out.println(“Unable to connect to database”);
}
```