# PROVINCE OF KWAZULU-NATAL
# DEPARTMENT OF EDUCATION
## PINETOWN DISTRICT

**NATIONAL SENIOR CERTIFICATE**

**GRADE 12**

**INFORMATION TECHNOLOGY P1**

**TRIAL EXAMINATIONS 2014**

**Marks: 150**

**Time: 3 hours**

**This question paper consists of 12 pages**

NSC – Grade 12 Trials

**INSTRUCTIONS AND INFORMATION**

1. This question paper contains THREE questions.

2. Answer ALL the questions.

3. 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.

4. This question paper is set in programming terms that are specific to the Java programming language (making use of the Netbeans IDE).

5. Make sure that you answer the questions according to the specifications that are given in each question. Marks will only be awarded based on the set requirements.

6. Only answer what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.

7. 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.

8. Make sure that you develop routines, such as search, sort and selection, from first principles and not use the built-in features of a programming language for any of these routines.

9. You, as the programmer, must define all data structures. You may not use components provided within the user interface to store and later retrieve data.

10. Save your work regularly on the disk (CD/flash disk/DVD, et cetera) that you have been given, or on the disk space allocated to you for this examination.

11. Make sure that your examination number appears as a comment in the first line of code you did to answer a question. Also include the question number as part of the comment.

12. If printouts are required make printouts of the code of all the programs/classes/units that you did and NOT of the code that is generated automatically.

13. Printing must be done after the examination within the timeframe provided for printing.

14. At the end of this examination session, you must hand in the disk/CD with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you. Ensure that all files can be read.

15. You have been supplied with either a disk or disk space containing files you need to complete this question paper.

16. Follow the instructions provided for each question to complete the question paper.

**SCENARIO:**

Fund-raising has become essential in all schools. The teachers, together with learners and parents at Everest High School in KZN, have planned and organized several fund-raising events for the year, to ensure they have sufficient funds.

## SECTION A: JAVA PROGRAMMING

### QUESTION 1:

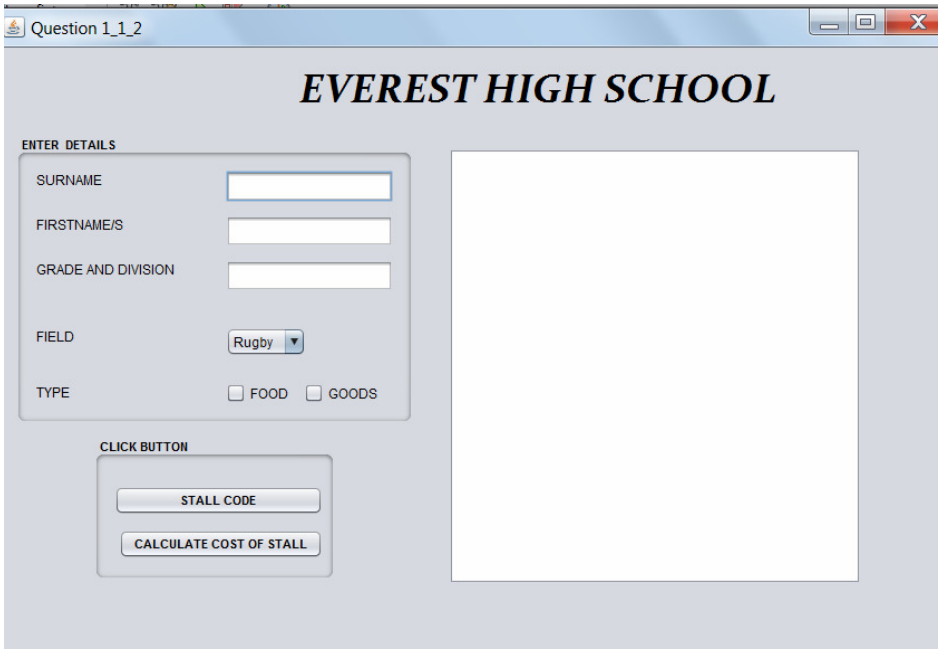
The school had decided to hold a market day in order to raise funds.Each class is responsible for a stall that will be set up, either on the rugby field, the soccer field or netball field. Each class will receive a unique stall code. Each class will pay a fee for the use of the stall, depending on where the stall is set up and the type of product they sell. The class has an option of paying for the stall before the market day(cash) or after it has taken place(credit).

Open the project **Question1** in Netbeans.

You are provided with the GUI called Question1_1_2

Insert your Full Name and Grade as comments on the first two lines of this class

Use this GUI to answer Question 1.1 and Question 1.2




All Output must be made in the Text Area.

Insert the appropriate code under the respective buttons.

## 1.1 Button: STALL CODE

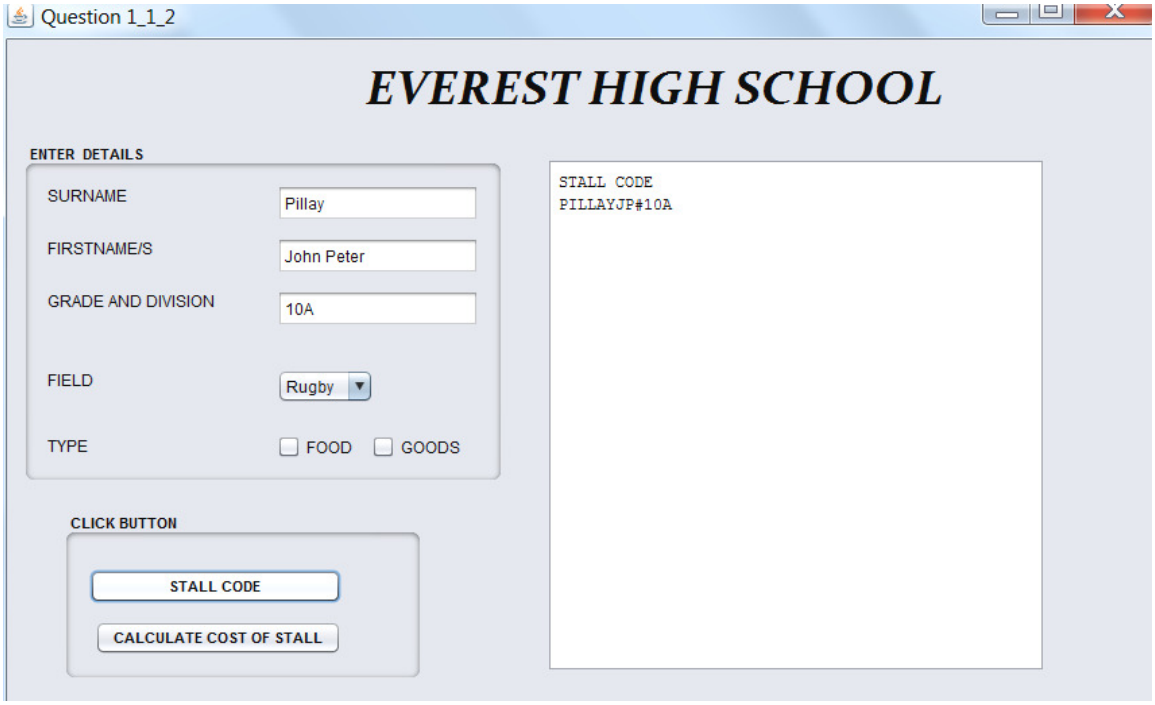
It is compulsory that a stall is given a unique code for administrative purposes.

Enter the name and surname of the class teacher, the grade and division of the class in the text fields provided on the user interface.

Create a unique code for the stall as follows:

- Surname of the class teacher.
- Initials of the class teacher.
- Grade and division of the class
- Convert the entire code to capital letters.
- Format and output the code as follows:
  <SURNAME><INITIALS><#><GRADEDIVISION>

**Sample Output:**




(10)

## 1.2 Button: CALCULATE COST OF STALL

The class has to pay a fee for the stall.

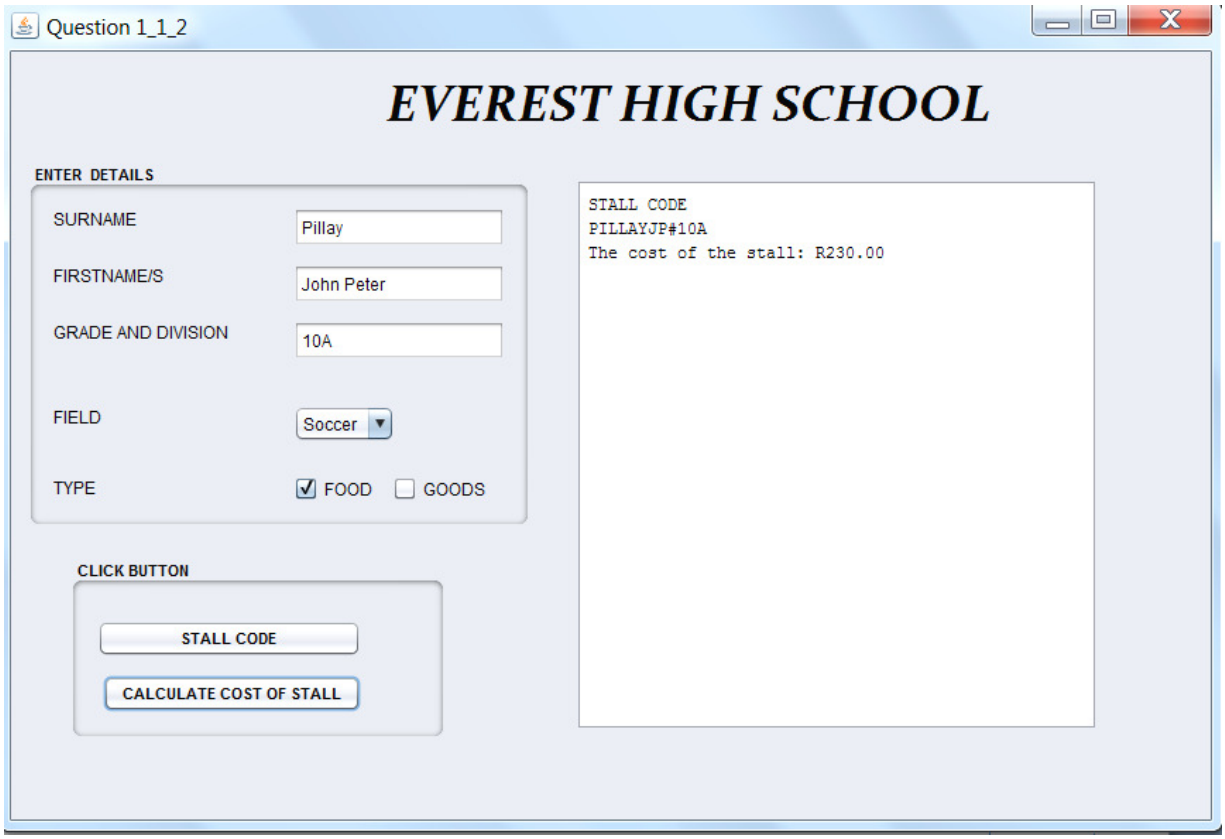
- Extract the type of product being sold (Food or Goods or both) and the preferred field (rugby, soccer or netball) from the respective GUI components.

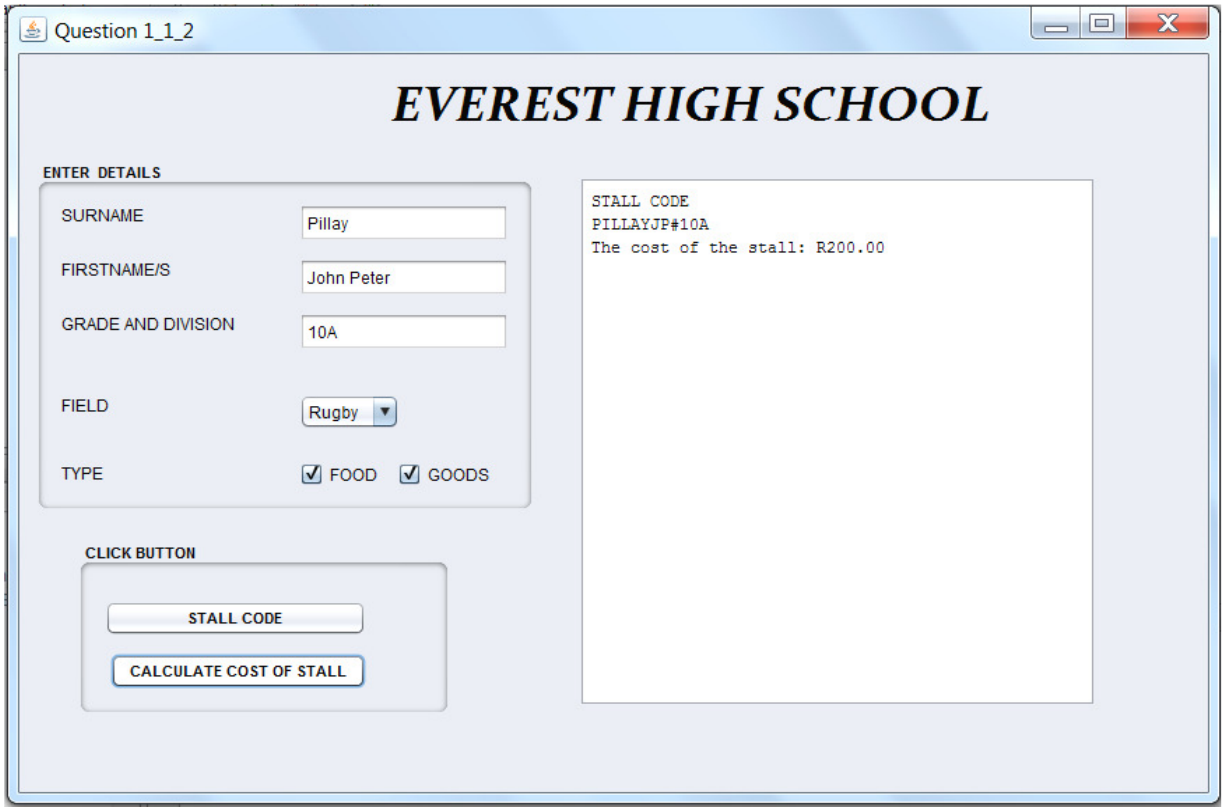
- Calculate the fee for the stall using the criteria below:

  - A food stall cost R80, a goods stall cost R50. A stall, selling both food and goods costs R100.00.

  - An additional charge is added to the cost depending on the choice of field where the stall is set up:
    - A stall on the Rugby or Netball field costs R100
    - A stall on the Soccer field costs R150

**Sample Outputs:**



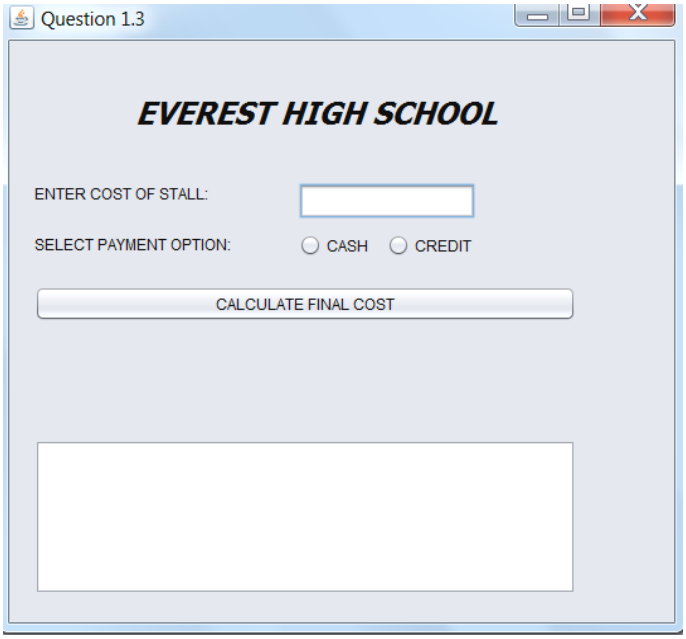




(11)

You are provided with the GUI called Question1.3

Insert your Full Name and Grade as comments on the first two lines of this class

Use this GUI to answer Question 1.3




## 1.3 Button: CALCULATE FINAL COST

A discount/interest will be given to the classes for the cost of the stall based on their payment options.

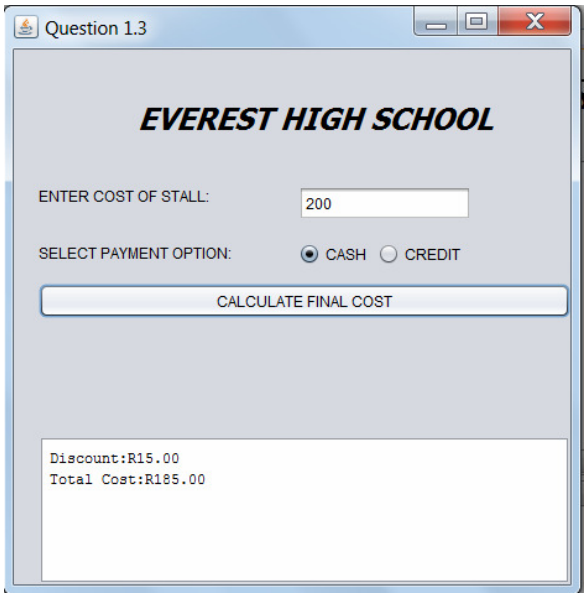
- Extract the cost of the stall and the classes payment option (Cash or Credit) from the respective GUI components.
- Ensure that a number is entered for cost.
- If the user selects the **Cash** option:
  - A discount of 7.5% of the cost of the stall is given
  - Calculate and display the discount and final amount, formatted to two decimal places.
- If the user selects the **Credit** option:
  - Prompt the user to enter a 6 character stall code via an input message dialog.
  - The first character of the stall code must be an uppercase letter.
  - The last two characters of the stall code must be digits.
  - The stall code must be validated. If the stall code is not valid, prompt the user to re-enter a code until a valid code is entered.

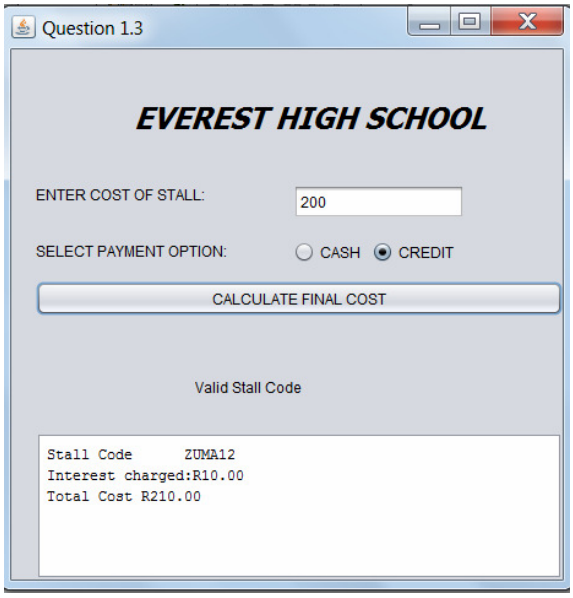
- When a valid stall code is entered write code to do the following:
  - dynamically instantiate a label with the following specifications and functionality:

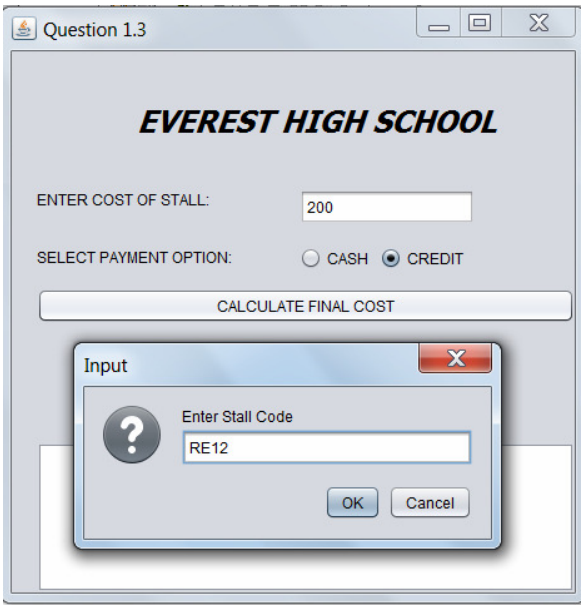
| | |
|---|---|
| Name of Panel: | jPanel1 |
| Name of the label: | newlabel |
| Text on the label: | Valid Stall Code |
| Bounds: | 150,250,100,50 |

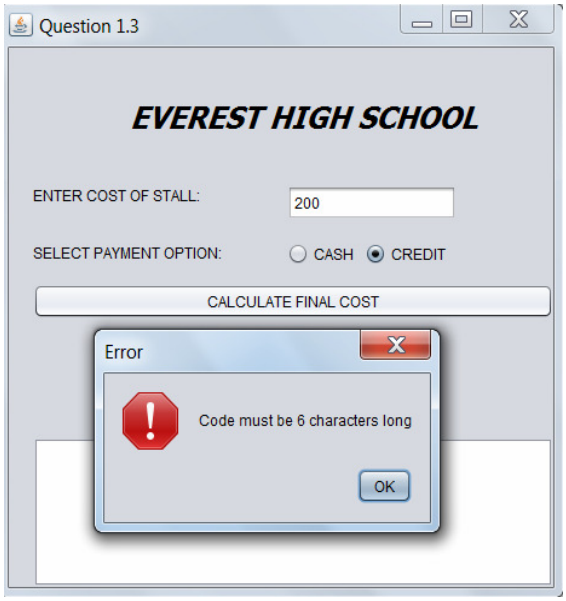
  - Calculate interest at a rate of 5% of the cost of the stall.
  - Calculate the final cost of the stall.
  - Display the stall code, the interest charged and the final cost formatted to 2 decimal places.

**Sample Output:**




(24)

**SECTION TOTAL [45]**

## SECTION B

## QUESTION 2:

One of the Fundraising events at Everest High School is a Community Fun Run. There are TWO types of races on the day. A 3Km race and a 5Km race. At a SGB meeting it was decided to put entry restrictions to the races to ensure the well being of the participants. The 3Km would be ideally suited for participants 10 years and younger, however if older participants want to take part in this race then they would allowed too, especially parents / guardians / grandparents. For the 5Km race, however, only individuals older than 10 years would be allowed to participate.

The school requires software to collate the entries for the participants as well as be able to generate much required statistics.

Open the project **Question2** in Netbeans.

2.1 Create a class called **funRun** and do the following:

Insert your Full Name and Grade as comments on the first two lines of this class

2.1.1 Create private the following class variables:

- numFemales (total number of female participants)
- numMales (total number of female participants)
- num5Km (total number of 5Km participants)
- num3Km (total number of 3Km participants)
- invalid (total number of invalid participants) (3)

2.1.2 Create a **default** constructor to initialise the class variables to zero. (2)

2.1.3 Create **accessor** methods for the class variables. (2)

2.1.3 Create the following methods to increment the value of each class variable:

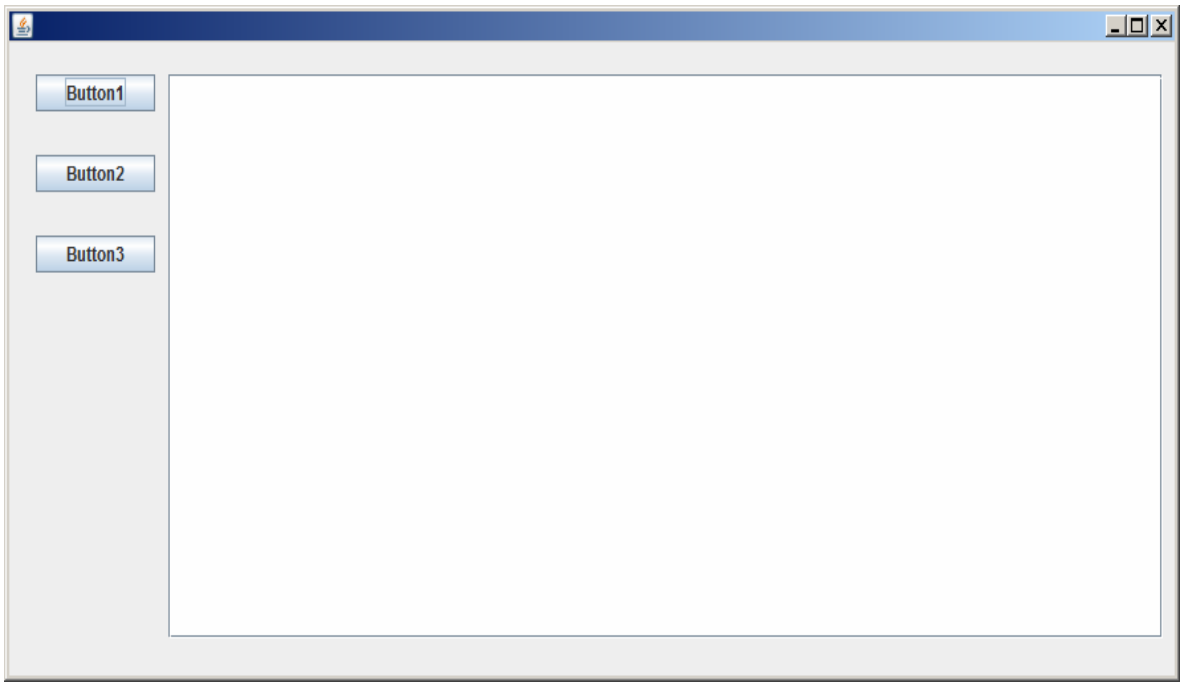
- incNumFemales
- incNumMales
- incNum5Km
- incNum3Km
- incInvalid (3)

2.1.4 Create a method called **determineTotal** that will determine and return the total number of valid participants. (2)

2.1.5 Create a method called **determinePercDiff** that will determine the percentage of male participants' as well as percentage or female participants' (using the method created in 2.1.4) and then return the difference between these percentages (a rounded value). (6)

2.1.6 Create a method called **determinePercInvalid** that will determine and return the percentage of invalid entries (a rounded value) from all the entries received. (4)

2.2 Open the java frame **testFunRun** which has Buttons and a Text Area.




Insert your Full Name and Grade as comments on the first two lines of this class

A text file called **Data.txt**, containing all the participants, has also been supplied for the project. Participants details are stored in the text file in the following format:

Firstname;Surname#Gender;Date of Birth#Race being Run

The **first three lines** of the text file are displayed below:

Joe;Ellis#M;03/08/1997#3Km
Sue;Smith#M;20/06/1998#3Km
Mike;Naidoo#M;17/11/1980#5Km

All Output must be made in the Text Area.

Insert the appropriate code under the respective buttons.

2.2.1 Create a class variable called **obj** of the type **funRun** (2)

2.2.2 Write code under **Button1** to:

- Read information from the text file **Data.txt:**
  - Read each line from the text file.
  - Separate the line into the participant's details.
  - Determine the age of the participant (in years) using the year of the participant's date of birth and the current year.
  - Using the participant's age, determine whether or not the participant has entered the correct race.

- Write the participant's details to the text file **InvalidEntries.txt** if the participant has entered the incorrect race. Write the details of the participant to the text file in the following tabular format:

  Firstname Surname Gender Age

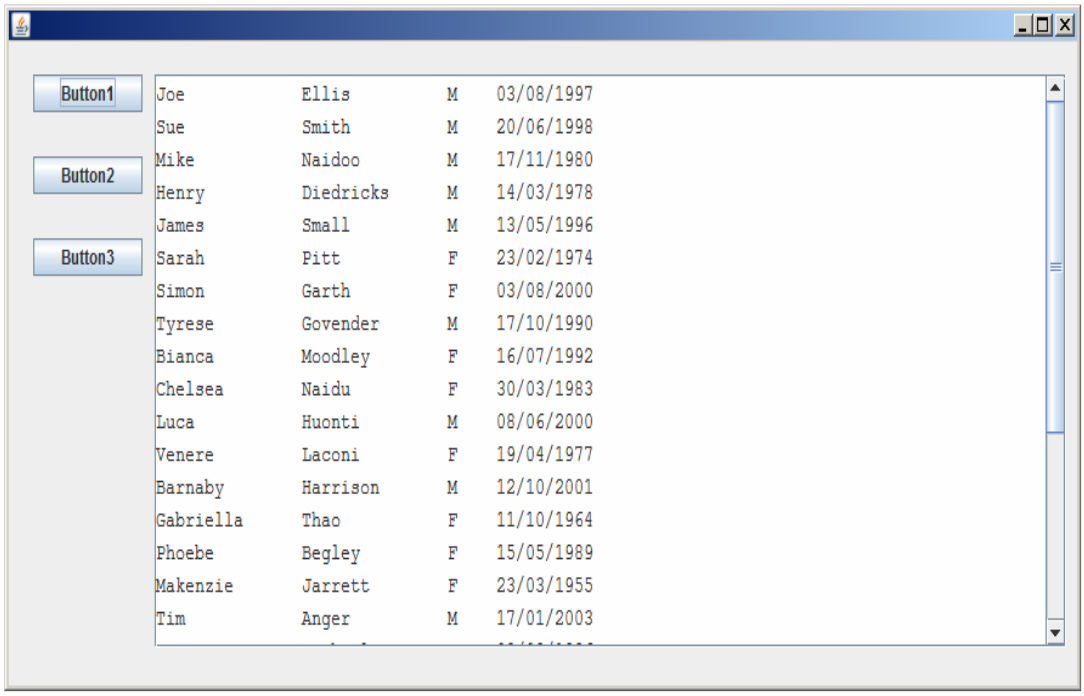
- Call the appropriate methods to increase the relevant class variables from the funRun class based on the participant's details.

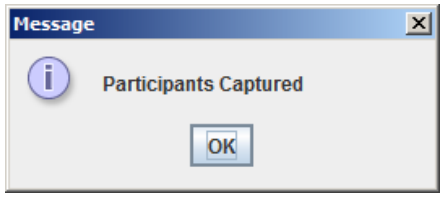
- Display the valid participant's details in the text are in the following tabular format:

  Firstname Surname Gender Date of Birth

- Display a GUI message when the capture is complete.

**Sample Output:**



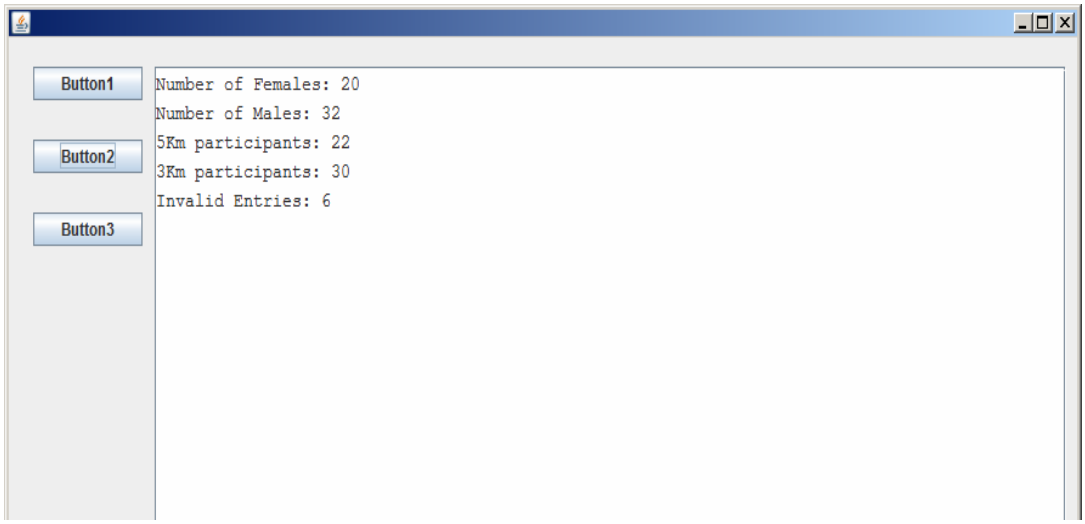

(40)

2.2.3 Write code under **Button 2** to:

- Display the following statistics (in the format below) of fun run in text area provided:

  Number of Females:
  Number of Males:
  5Km participants:
  3Km participants:
  Invalid Entries:

**Sample Output:**

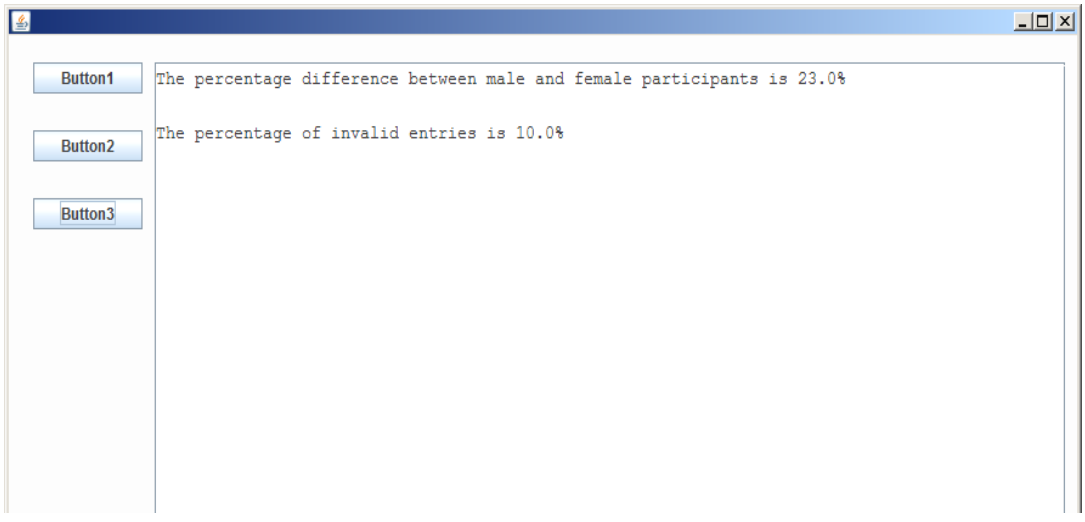



(3)

2.2.4 Write code under **Button 3** to:

- Display in the text area, with suitable text:
  - the percentage difference between the male and female participants
  - the percentage of invalid entries

**Sample Output:**




(3)

**SECTION TOTAL [70]**

**SECTION C**

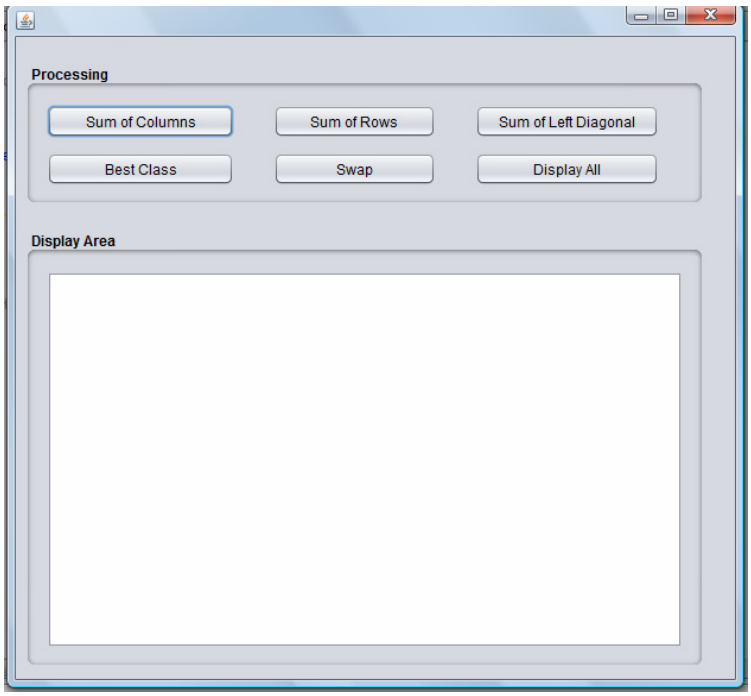
**QUESTION 3:**

**INSTRUCTIONS:**

In the spirit of fundraising, Everest High School has planned a little competition amongst the 4 grade 12 classes. The school will host 4 different fundraising events. The totals raised by each of the grade 12 classes, for every event is recorded. The class that raised the most for all events will be declared the winners of the competition.

Open the project **Question3**.

The following GUI has been created for you:




The program currently has no functionality.

The following declarations have been provided for you:

- A one-dimensional array named **events** has been supplied consisting of the four events that took place.
- A one-dimensional array named **classNames** has been supplied consisting of the names of the four classes that participated.
- A two-dimensional array named **amounts** has been supplied consisting of the total amounts raised by the classes for the four events.

**Complete the code for each of the questions below :**

3.1 **Button [Sum of Columns]:**

Write code to calculate the sum of each column and store in **colTotals**. The one-dimensional array **colTotals** has already been declared for you. (6)

3.2 **Button [Sum of Rows]:**

Write code to calculate the sum of each row and store in **rowTotals**. The one-dimensional array **rowTotals** has already been declared for you. (6)

3.3 **Button [Sum of Left Diagonal]:**

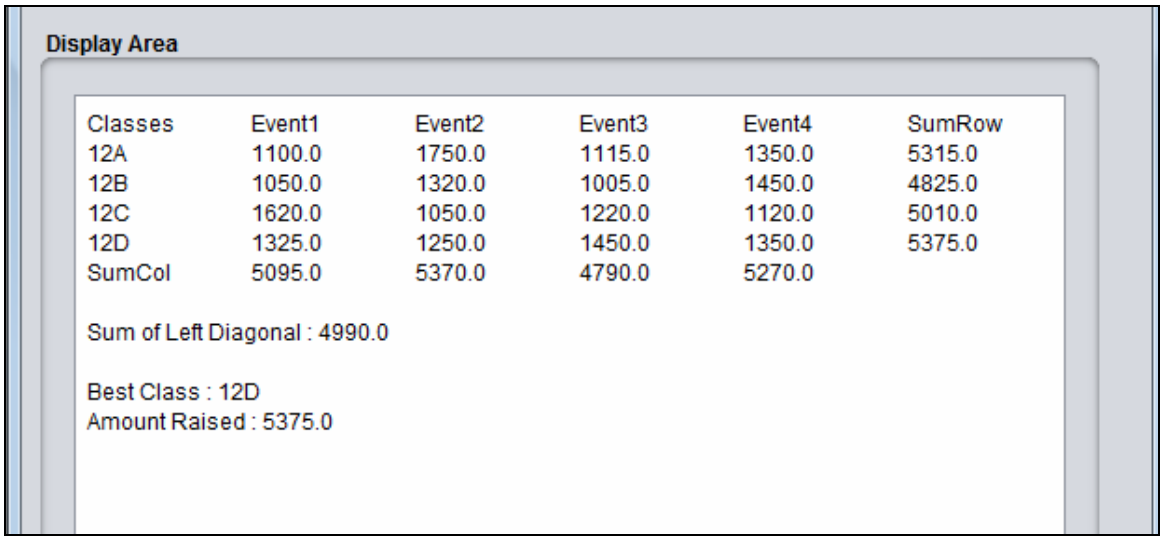
Write code to calculate the sum of the left diagonal. A variable **leftd** has already been declared for you. (4)

3.4 **Button [Best Class]:**

Write code to find the class that has raised the highest total amount. The name of the class as well as their total amount raised must be found. The variables **bestClass** and **bestClassAmount** has already been declared for you. (5)

3.5 **Button [Display All]:**

Write code to display the events, class names, amounts, rowTotals and colTotals in a grid form, as well as the sum of left diagonal, best class name and best class amount. Below is a screenshot showing the expected output.




(11)

3.6 **Button [Swap]:**

The amounts raised for the first two classes have been captured incorrectly. Write code to swap the amounts for the first two classes '12A'and '12B'. (3)

**SECTION TOTAL [30]**

**TOTAL [150]**