NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2014

INFORMATION TECHNOLOGY: PAPER II

MARKING GUIDELINES

Time: 3 hours 120 marks

These marking guidelines are prepared for use by examiners and sub-examiners, all of whom are required to attend a standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' scripts.

The IEB will not enter into any discussions or correspondence about any marking guidelines. It is acknowledged that there may be different views about some matters of emphasis or detail in the guidelines. It is also recognised that, without the benefit of attendance at a standardisation meeting, there may be different interpretations of the application of the marking guidelines.

SECTION A SQL ANSWER SHEET

1.1:

SELECT *
FROM Passengers
WHERE Destination = 'Cape Town'

(Accept LIKE for all questions where = is used)

1.2:

SELECT DISTINCT BaggageType
FROM Baggage

Alternatively: SELECT BaggageType FROM Baggage GROUP BY BaggageType

1.3:

INSERT INTO Claims (ClaimID, BaggageID, Description)
VALUES (16, 240402, 'Theft')

(1 mark for correct value and 1 mark for all values correct types; Field list can be omitted if fourth value is included as "" or null; Do not accept "Stolen" or any other value only "Theft")

1.4:

DELETE * FROM Baggage WHERE PassengerID = 14

1.5:

UPDATE Claims
SET Reference = LEFT(Description, 2) + RIGHT(BaggageID, 4)

Alternatively for mysql: CONCAT (LEFT(Description,2) , RIGHT(BaggageID,4))

(Accept & instead of + and any working string manipulation)

1.6:

SELECT PassengerID (or Fullname if Join was used) , SUM(Weight) AS TotalWeight
FROM Baggage
GROUP BY PassengerID
ORDER BY SUM(Weight) DESC

Alternatively (mysql only): ORDER BY TotalWeight DESC

(Do not reward OR penalise if a join was used and the passenger name was listed instead of ID)

1.7:

SELECT Fullname, InsuredValue, Weight
FROM Passengers, Baggage
WHERE Baggage.PassengerID = Passengers.PassengerID
AND (BaggageType = 'Sporting Equipment' OR BaggageType = ‘Other’)

(Accept INNER JOIN instead of WHERE Baggage.PassengerID = Passengers.PassengerID)
(-1 for omitting brackets in WHERE clause unless INNER JOIN used)
(Accept BaggateType IN ('Sporting Equipment', 'Other') for 2 marks)

1.8:

```
SELECT Fullname
FROM Passengers
WHERE PassengerID NOT IN (SELECT PassengerID FROM Baggage)
```

Alternatively:

```
SELECT Fullname FROM Passengers
LEFT JOIN Baggage ON Passengers.PassengerID = Baggage.PassengerID
WHERE Baggage.PassengerID is null
```

1.9:

```
SELECT *
FROM Baggage
WHERE CheckInCounter = 6 AND Fragile = true
AND InsuredValue > (SELECT AVG(InsuredValue) FROM Baggage)
```

(If InsuredValue > AVG(InsuredValue) used without nested query then -1 mark)
(Accept Fragile = Yes or Fragile = 1 in MS Access)

SECTION B OBJECT ORIENTED PROGRAMMING

JAVA SOLUTION:

QUESTION 2 AND 6.1

```
package question2;

// Question 2.1
public class Flight
```
(If named incorrectly penalise -1 but only here and not again in Q3.1)

```
{
    // Question 2.2
    private String code;
    private String origin;
    private String destination;
    private String departureTime;
    private String arrivalTime;
    private double cost;
```

Private
Correct Types
Appropriate names
(If named incorrectly penalise -1 but only here and not again in Q3.2)

```
    // Question 2.3
    public Flight  (String cde, String orig, String dest, String dtime, String
                      atime, double cst)
    {
        code = cde;
        origin = orig;
        destination = dest;
        departureTime = dtime;
        arrivalTime = atime;
        cost = cst;
    }
```

Assignment of parameters to attributes

```
    // Question 2.4
    public String getCode()
    {
        return code;
    }

    public String getDepartureTime()
    {
        return departureTime;
    }

    public double getCost()
    {
        return cost;
    }
```

Three accessors correctly named
Correct return types
Return the correct attributes' values
(If named incorrectly penalise -1 but only here and not again later)

```
    // Question 2.5
    public String toString()
    {
        return  code + "\t" + origin + " " + departureTime + "\t" +
                destination + " " + arrivalTime;
    }
```

[14]

```
    // Question 6.1
    public int getDuration()     (2 marks for putting method in correct class)
    {
        int endTime = Integer.parseInt(arrivalTime.substring(11,13)) * 60 +
                        Integer.parseInt(arrivalTime.substring(14, 16));
        int startTime = Integer.parseInt(departureTime.substring(11,13)) * 60 +
                        Integer.parseInt(departureTime.substring(14, 16));

        return endTime - startTime;
    }
}
```

[4]

QUESTION 3

```
// Question 3.1
public class Ticket
{
    // Question 3.2
    private String ticketID;
    private String name;
    private Flight departingFlight;
    private Flight returningFlight;
```

ticketID and name declared with correct name, type and private access modifier
departing and returningFlight Declared as objects

```
    // Question 3.3
    public Ticket (String tID, String nme , Flight dflight, Flight rflight )
    {
        ticketID = tID;
        name = nme;
        departingFlight= dflight;
        returningFlight = rflight;
    }
```

Assignment of parameters to attributes

```
    // Question 3.4
    public String getName()
    {
        return name;
    }

    // Question 3.5
    public Flight getDepartingFlight()
    {
        return departingFlight;
    }

    public Flight getReturningFlight()
    {
        return returningFlight;
    }
```

Accessor methods for flights
Return object attributes

```
    // Question 3.6
    public double getTotalCost()
    {
        return departingFlight.getCost() + returningFlight.getCost();
    }

    // Question 3.7
    public String toString()
    {
        return ticketID + "\t" + name + "\tR " + getTotalCost() + "\n"  +
                        departingFlight.toString()  +
                    "\n" + returningFlight.toString() ;
    }
}
```

[15]

QUESTIONS 4 AND 6.2

```
import java.io.*;

// Question 4.1
public class FlightManager
{
    // Question 4.2
    private Ticket[] tickets = new Ticket[500]   ;
    private int counter = 0 ;

    // Question 4.3
    public FlightManager()
    {
        try
        {
            BufferedReader fIn = new BufferedReader(new
                        FileReader("tickets.txt"));

            String line = fIn.readLine();

            while (line != null)
            {
                String[] ticketTokens = line.split("#");

                line = fIn.readLine();
                String[] thereTokens = line.split("#");

                line = fIn.readLine();
                String[] backTokens = line.split("#");
```

Reading in a line
Repeating the read for two subsequent lines
Tokenizing a line
Repeating the process for the two other lines

```
                Flight thereFlight = new Flight(thereTokens[0], thereTokens[1],
                        thereTokens[2], thereTokens[3], thereTokens[4],
                        Double.parseDouble(thereTokens[5]));

                Flight backFlight  = new Flight(backTokens[0], backTokens[1],
                        backTokens[2], backTokens[3], backTokens[4],
                        Double.parseDouble(backTokens[5]));

                tickets[counter]   = new Ticket(ticketTokens[0],
                       ticketTokens[1], thereFlight, backFlight);

                counter++;
                line = fIn.readLine();
            }

            fIn.close();
        }
        catch (IOException e)
        {
            System.out.println("File not found!");
        }
    }
```

Reading in a line
Repeating the read for two subsequent lines
Tokenizing a line
Repeating the process for the two other lines

```
// Question 4.3 – Alternative Solution using Scanner Class
public FlightManager()
{
    try
    {
        Scanner fIn = new Scanner(new File("tickets.txt"));

        while (fIn.hasNextLine())
        {
            String line = fIn.nextLine();
            Scanner ticketTokens = new Scanner(line).useDelimiter("#");

            line = fIn.nextLine();
            Scanner thereTokens = new Scanner(line).useDelimiter("#");

            line = fIn.nextLine();
            Scanner backTokens = new Scanner(line).useDelimiter("#");

            Flight thereFlight  = new Flight(thereTokens.next(),
                    thereTokens.next(), thereTokens.next(),
                    thereTokens.next(), thereTokens.next(),
                    thereTokens.nextDouble());

            Flight backFlight  = new Flight(backTokens.next(),
                    backTokens.next(), backTokens.next(), backTokens.next(),
                    backTokens.next(), backTokens.nextDouble());

            tickets[counter]   = new Ticket(ticketTokens.next(),
                    ticketTokens.next(), thereFlight, backFlight);

            counter++;
        }

        fIn.close();
    }
    catch (IOException e)
    {
        System.out.println("File not found!");
    }
}

// Question 4.4
public String allTickets()
{
    String output = "";

    for(int i = 0; i < counter; i++)
    {
        output = output + tickets[i].toString()  + "\n\n";
    }

    return output;
}
```

```
// Question 4.5
public void sort()
{
    for (int i = 0; i < counter - 1; i++)
    {
        for(int j = 0; j < counter - i - 1; j++)
        {
            if (tickets[j].getDepartingFlight().getDepartureTime()
                    .compareTo(tickets[j+1].getDepartingFlight()
                    .getDepartureTime()) > 0)
            {
                Ticket tempTicket = tickets[j];
                    tickets[j] = tickets[j+1];
                tickets[j+1] = tempTicket;
            }
        }
    }
}
```

(Temp variable must be same type as array)

(Do not deduct any marks if candidate sorts only on the time portion of the DepartureTime)

[32]

ALLOCATE MARKS TO ANY CORRECT ALTERNATIVE SOLUTION AS PER RUBRIC

```
// Question 6.2
```

Marked with Rubric

```
public String frequentFlyer()
{
    int maxTime = 0;
    String maxName = "";
    String checkedNames = "";

    for(int i = 0; i < counter; i++)
    {
        int time = 0;
        String currentName = tickets[i].getName();
        if (checkedNames.indexOf(currentName) < 0)
        {
            for(int j = 0; j < counter; j++)
            {
                if (tickets[j].getName().equals(currentName))
                {
                    time = time +
                    tickets[j].getDepartingFlight().getDuration()  +
                            tickets[j].getReturningFlight().getDuration();
                }
            }
        }
        checkedNames = checkedNames + currentName;

        if (time > maxTime)
        {
            maxTime = time;
            maxName = currentName;
        }
    }

    return maxName + " " + maxTime;
}
}
```

Solution is efficient AND uses minimal code
(Award this mark if the algorithm does not check a passenger's total duration more than once AND uses minimal code)

[10]

ALLOCATE MARKS TO ANY CORRECT ALTERNATIVE SOLUTION AS PER RUBRIC

QUESTIONS 5 AND 6.3

```
// Question 5.1
public class FlightUI
{
    public static void main(String[] args)
    {
        // Question 5.2
        FlightManager fm = new FlightManager();

        // Question 5.3
        fm.sort();
        System.out.println(fm.allTickets());
```

[4]

```
        // Question 6.3
        System.out.println();
        System.out.println("Most Frequent Flyer");
        System.out.println("-------------------");
        System.out.println(fm.frequentFlyer());
    }
}
```

[1]

ALTERNATIVE JAVA SOLUTION FOR QUESTION 6.2

Additions to the Ticket Class highlighted in bold

```
public class Ticket
{   …
    private int miles;

    public Ticket(String tID, String nme, Flight dflight, Flight rflight)
    {
        …
        miles = dflight.getDuration() + rflight.getDuration();
    }
```

Methods are in the correct class

```
    public int getMiles()
    {
        return miles;
    }

    public void increaseMiles(int m)
    {
        miles += m;
    }
}
```

frequentFlyer method in FlightManager.java

```
    public String frequentFlyer()
    {
        Ticket[] frequentFlyers = new Ticket[counter];
        System.arraycopy(tickets, 0, frequentFlyers, 0, counter);

        for (int i = 0; i < counter; i++)
        {
            if (frequentFlyers[i] != null)
            {
                for (int j = i + 1; j < counter; j++)
                {
                    if (frequentFlyers[j] != null &&
                        frequentFlyers[j].getName().equals(frequentFlyers[i].get
                        Name()))
                    {

                        frequentFlyers[i].increaseMiles(frequentFlyers[j].getMil
                        es());
                         frequentFlyers[j] = null;
                    }
                }
            }
        }
        int maxPos = 0;
        for(int i = 0; i < counter; i++ )
        {
            if (frequentFlyers[i] != null && frequentFlyers[i].getMiles() >
                        frequentFlyers[maxPos].getMiles())
            {
                maxPos = i;
            }
        }
        return frequentFlyers[maxPos].getName() + " " +
                        frequentFlyers[maxPos].getMiles();
}
```

Efficiency as stated above

ALLOCATE MARKS TO ANY CORRECT ALTERNATIVE SOLUTION AS PER RUBRIC

DELPHI SOLUTION:

QUESTION 2 AND 6.1

```
unit uFlight;

interface

uses SysUtils;

// Question 2.1
type TFlight = class
  private
    // Question 2.2
    code : String;
    origin : String;
    destination : String;
    departureTime : String;
    arrivalTime : String;
    cost : real;
  public
    constructor Create(cde, orig, dest, dtime, atime : String; cst : real);
    function getDuration : integer;
    function getCode : String;
    function toString: String;
    function getCost : real;
    function getDepartureTime : String;
end;
```

(If named incorrectly penalise -1 but only here and not again in Q3.1)

Private
Correct Types
Appropriate names
(If named incorrectly penalise -1 but only here and not again in Q3.2)

```
implementation

{ TFlight }

// Question 2.3
Constructor TFlight.Create (cde, orig, dest, dtime, atime : String; cst : real);
begin
  code := cde;
  origin := orig;
  destination := dest;
  departureTime := dtime;
  arrivalTime := atime;
  cost := cst;
end;
```

Assignment of parameters to attributes

```
// Question 2.4
function TFlight.getCode: String;
begin
  Result := code;
end;

function TFlight.getCost: real;
begin
  Result := cost;
end;

function TFlight.getDepartureTime: String;
begin
  Result := departureTime;
end;
```

Three accessors correctly named
Correct return types
Return the correct attributes' values
(If named incorrectly penalise -1 but only here and not again later)

```
// Question 2.5
function TFlight.toString: String;
begin
  Result  := code + #9 + origin + ' ' + departureTime + #9 + destination + ' ' +
                      arrivalTime;
end;
```

[14]

```
// Question 6.1
function TFlight.getDuration: integer;
var
  startTime, endTime : Integer;
begin
  endTime := StrToInt(Copy(arrivalTime, 12, 2)) * 60 +
                      StrToInt(Copy(arrivalTime, 15, 2));
  startTime := StrToInt(Copy(departureTime, 12, 2)) * 60 +
                      StrToInt(Copy(departureTime, 15, 2));

  Result := endTime - startTime;
end;

end.
```

(2 marks for putting method in correct class)

[4]

QUESTION 3

```
unit uTicket;

interface

uses SysUtils, uFlight;

// Question 3.1
type TTicket = class
  private
    // Question 3.2
    ticketID : String;
    name : String;
    departingFlight : TFlight;
    returningFlight : TFlight;
  public
    constructor Create(tID : String; nme : String; dflight, rflight : TFlight);
    function getTotalCost : real;
    function toString : String;
    function getDepartingFlight : TFlight;
    function getReturningFlight : TFlight;
    function getName : String;
end;
```

ticketID and name declared with correct name, type and private access modifier
departing and returningFlight Declared as objects

```
implementation

{ TTicket }

// Question 3.3
constructor TTicket.Create (tID, nme: String; dflight, rflight: TFlight) ;
begin
  ticketID := tID;
  name := nme;
  departingFlight := dflight;
  returningFlight := rflight;
end;
```

Assignment of parameters to attributes

```
// Question 3.4
function TTicket.getName: String;
begin
  Result := name;
end;

// Question 3.5
function TTicket.getDepartingFlight: TFlight;
begin
  Result := departingFlight;
end;

function TTicket.getReturningFlight: TFlight;
begin
  Result := returningFlight;
end;
```

Accessor methods for flights
Return object attributes

```
// Question 3.6
function TTicket.getTotalCost: real;
begin
  Result := returningFlight.getCost + departingFlight.getCost;
end;

// Question 3.7
function TTicket.toString: String;
begin
  Result := ticketID + #9 + name + #9 + 'R ' + FloatToStr(getTotalCost) + #13  +
                      departingFlight.toString  + #13 +
                      returningFlight.toString ;
end;

end.
```

[15]

QUESTIONS 4 AND 6.2

```
unit uFlightManager;

interface

uses uTicket, uFlight, SysUtils, Dialogs;

// Question 4.1
type TFlightManager = class
  private
    // Question 4.2
    tickets : array[1..500] of TTicket;
    counter : integer;
  public
    constructor Create;
    function allTickets : String;
    procedure sort;
    function frequentFlyer : String;
end;

implementation

{ TFlightManager }
```

```
// Question 4.3
constructor TFlightManager.Create;
var
  infile : textfile;
  line : String;
  name, ticketID : String;
  // Details for the flight there
  thereCode, thereOrig, thereDest, thereOrigTime, thereDestTime : String;
  thereCost : real;
  // Details for the flight back
  backCode, backOrig, backDest, backOrigTime, backDestTime : String;
  backCost : real;
  toFlight, fromFlight : TFlight;
begin
  If FileExists('tickets.txt') <> true  then
    begin
      ShowMessage('File Not Found');
    end
  else
    begin
      AssignFile(infile, 'tickets.txt');
      Reset(infile);

      counter := 0;

      while NOT EOF(infile) do
        begin
          ReadLn(infile, line);
          Inc(counter);

          ticketID := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          name := line;

          ReadLn(infile, line);
          thereCode := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          thereOrig := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          thereDest := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          thereOrigTime := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          thereDestTime := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          thereCost := StrToFloat(line);

          ReadLn(infile, line);
          backCode := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          backOrig := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          backDest := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          backOrigTime := Copy(line, 1, Pos('#', line) - 1);
```

Reading in a line
Repeating the read for two
subsequent lines
Tokenizing a line
Repeating the process for the two other lines

```
          Delete(line, 1, Pos('#', line));

          backDestTime := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          backCost := StrToFloat(line);

          toFlight  := TFlight.Create(thereCode, thereOrig, thereDest,
                       thereOrigTime, thereDestTime, thereCost);
          fromFlight  := TFlight.Create(backCode, backOrig, backDest,
                       backOrigTime,backDestTime, backCost);

          tickets[counter]   := TTicket.Create(ticketID, name, toFlight,
                       fromFlight);
        end;
    end;
end;

// Question 4.4
function TFlightManager.allTickets: String;
var
  i : integer;
  output : String;
begin
  output := '';

  for i := 1 to counter do
    begin
      output := output + tickets[i].toString  + #13 + #13;
    end;

  Result := output;
end;

// Question 4.5
procedure TFlightManager.sort;
var
  i, j : integer;
  tempTicket : TTicket;
begin
  for i := 1 to counter - 1 do
    for j := 1 to counter - i do
      begin
        if (tickets[j].getDepartingFlight.getDepartureTime >
                       tickets[j+1].getDepartingFlight.getDepartureTime) then
          begin
            tempTicket := tickets[j];
            tickets[j] := tickets[j+1];
            tickets[j+1] := tempTicket;
          end;
      end;
end;
```

(Do not deduct any marks if candidate sorts only on the time portion of the DepartureTime)

(Temp variable must be same type as array)

[32]

ALLOCATE MARKS TO ANY CORRECT ALTERNATIVE SOLUTION AS PER RUBRIC

```
// Question 6.2
```

Marked with rubric

```
function TFlightManager.frequentFlyer: String;
var
  i, j, time, maxTime : integer;
  currentName, maxName, checkedNames : String;
begin
  maxTime := 0;
  maxName := '';
  checkedNames := '';

  for i := 1 to counter do
    begin
      time := 0;
      currentName := tickets[i].getName;
      if (Pos(currentName, checkedNames) <= 0) then
        begin
          for j := 1 to counter do
            begin
              if (tickets[j].getName = currentName)  then
                begin
                  time := time  + tickets[j].getDepartingFlight.getDuration  +
                         tickets[j].getReturningFlight.getDuration;
                end;
            end;
          checkedNames := checkedNames + currentName;
          if (time > maxTime) then
            begin
              maxTime := time;
              maxName := currentName;
            end;
        end;
    end;

    Result := maxName + ' ' + IntToStr(maxTime);
end;

end.
```

Solution is efficient AND uses minimal code (Award this mark if the algorithm does not check a passenger's total duration more than once AND uses minimal code)

[10]

ALLOCATE MARKS TO ANY CORRECT ALTERNATIVE SOLUTION AS PER RUBRIC

QUESTIONS 5 AND 6.3

```
unit uFrmFlightUI;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uFlightManager, StdCtrls, ComCtrls;

type
  // Question 5.1
  TfrmTicketGUI = class(TForm)
    rchOutput: TRichEdit;
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmTicketGUI: TfrmTicketGUI;
  fm : TFlightManager;

implementation

{$R *.dfm}

procedure TfrmTicketGUI.FormActivate(Sender: TObject);
begin
  // Question 5.2
  fm := TFlightManager.Create;

  // Question 5.3
  fm.sort;
  rchOutput.Lines.Add(fm.allTickets);
```
[4]

```
  // Question 6.3
  rchOutput.Lines.Add('');
  rchOutput.Lines.Add('Most Frequent Flyer');
  rchOutput.Lines.Add('-------------------');
  rchOutput.Lines.Add(fm.frequentFlyer);

end;

end.
```
[1]

ALTERNATIVE DELPHI SOLUTION FOR QUESTION 6.2

Additions to the TTicket Class highlighted in bold

```
unit uTicket;

interface

uses SysUtils, uFlight;

type TTicket = class
  private
    ...
    miles : integer;
  public
    ...
      Methods are in the correct class
    function getMiles : Integer;
    procedure increaseMiles(m : Integer);
end;

implementation

{ TTicket }
constructor TTicket.Create(tID, nme: String; dflight, rflight: TFlight);
begin
  ...
  miles := departingFlight.getDuration + returningFlight.getDuration;
end;

...

function TTicket.getMiles: Integer;
begin
  Result := miles;
end;

procedure TTicket.increaseMiles(m: Integer);
begin
  miles := miles + m;
end;

end.
```

frequentFlyer method in TFlightManager class

Efficiency as stated above

```
function TFlightManager.frequentFlyer: String;
var
  i, j, maxPos, fcounter : Integer;
  frequentFlyers : array[1..500] of TTicket;
  found : boolean;
begin
  fcounter := 0;
  for i := 1 to counter do
    begin
      found := false;
      for j := 1 to fcounter do
        begin
          if (tickets[i].getName = frequentFlyers[j].getName) then
            begin
              found := true;
              frequentFlyers[j].increaseMiles(tickets[i].getMiles);
            end;
        end;
```

```
    if (found = false)   then
        begin
          Inc(fcounter);
          frequentFlyers[fcounter] := tickets[i];
        end;
    end;

    maxPos := 1;

    for i := 1 to fcounter do
      begin
        if frequentFlyers[i].getMiles > frequentFlyers[maxPos].getMiles  then
          begin
            maxPos := i;
          end;
      end;

    Result := frequentFlyers[maxPos].getName + ' ' +
                        IntToStr(frequentFlyers[maxPos].getMiles);
end;
```

ALLOCATE MARKS TO ANY CORRECT ALTERNATIVE SOLUTION AS PER RUBRIC

OUTPUT

SECTION A STRUCTURED QUERY LANGUAGE

QUESTION 1.1

Query1			
PassengerID	Fullname	Flight	Destination
1	Emma Glenn	FL101	Cape Town
4	Zoe Rose	FL101	Cape Town
10	Kimberly Lawrence	FL101	Cape Town
14	Shana Woodward	FL101	Cape Town
15	Hanna Martinez	FL101	Cape Town
16	Clio Chan	FL101	Cape Town
21	Georgia Macias	FL101	Cape Town
25	Lee Cash	FL101	Cape Town
26	Ursula Reid	FL101	Cape Town
27	Fulton Foster	FL101	Cape Town
30	Miriam Delaney	FL101	Cape Town
32	Hop Hull	FL101	Cape Town
34	Clarke May	FL101	Cape Town
36	Elvis Stewart	FL101	Cape Town
37	Palmer Parker	FL101	Cape Town
39	Moses Franks	FL101	Cape Town
42	Yen Stark	FL101	Cape Town
43	Barry Chaney	FL101	Cape Town
44	Kuame Rich	FL101	Cape Town
46	Colleen Mann	FL101	Cape Town
54	Aspen Kelley	FL101	Cape Town
65	Suki Baldwin	FL101	Cape Town
68	Wing Mathews	FL101	Cape Town
72	Aimee Osborne	FL101	Cape Town

QUESTION 1.2

Query2
BaggageType
Other
Parcel
Sporting Equipment
Suitcase

QUESTION 1.3
(No output)

QUESTION 1.4
(No output)

QUESTION 1.5

(Table contents after query has been executed)

Claims			
ClaimID	BaggageID	Description	Reference
1	978472	Other	Ot8472
2	418006	Other	Ot8006
3	119910	Breakage	Br9910
4	879591	Lost in Transit	Lo9591
5	900899	Breakage	Br0899
6	117502	Theft	Th7502
7	555325	Theft	Th5325
8	130349	Other	Ot0349
9	895701	Breakage	Br5701
10	596430	Other	Ot6430
11	108210	Breakage	Br8210
12	883064	Breakage	Br3064
13	180426	Breakage	Br0426
14	585953	Theft	Th5953
15	892992	Breakage	Br2992

QUESTION 1.6

Query6	
PassengerID	TotalWeight
12	99.11
59	90.66
57	82.1
13	74.85
43	74.71
30	68.79
33	68.42
29	66.53
26	64.11
1	63.29
60	61.54
73	60.82
49	60.33
20	60.29
61	57.8
53	57.3
55	53.33
42	50.81
4	45.49
10	45.43
47	44.3
27	43.95

Query6	
PassengerID	TotalWeight
69	43.28
23	40.79
25	39.74
6	38.34
75	37.44
2	35.58
15	35.16
11	31.64
18	30.91
38	30.3
35	30.14
44	29.78
56	29.72
48	29.43
19	28.08
7	24.8
71	22.83
22	22.78
65	21.45
50	19.96
24	19.54
39	19.21
9	18.62
63	18.61
70	18.37
8	18
31	17.44
66	17.41
68	17.18
17	13.77
51	10.71
37	10.35
41	8.82
16	7.44
3	6.68
45	5.23
28	4.83
64	4.28
58	2.99
54	2.13
67	1.25

QUESTION 1.7

Query7		
Fullname	**InsuredValue**	**Weight**
Cheryl Booth	1648.1	28.93
Amery Harvey	2054.1	1.57
Tyrone Holt	710.8	15.74
Suki Baldwin	693.3	21.45
Barry Chaney	2420.7	23.66
Miriam Delaney	563.5	20.24
Hanna Martinez	599.1	6.54
Veda Bennett	1784.2	12.11
Nero Mercado	2374.9	4.58
Zoe Rose	945.7	25.33
Amery Harvey	1604.9	18.39
Kimberly Lawrence	507.6	1.98
Haley Bush	227.2	21.17
Tyrone Holt	1395.4	27.91
Keiko Bruce	908.4	4.94
Veda Bennett	230.7	3.51
Kimberly Holcomb	371.2	11.93
Tyrone Holt	786.8	13.81
Knox Schwartz	1244.4	9.01
Emma Glenn	883.3	24.46
McKenzie Stephens	1826	20.74
Burton Wiley	652.2	6.68
Haley Bush	933.9	8.26
Keiko Bruce	1810.4	27.65
Demetrius Kirk	2182.4	10.55
Dacey Tran	2011.7	18.61
Calvin Rodgers	1746.8	8.72
Kieran Johnston	290.2	3.88
Dean Reed	2207.8	23.01
Kimberly Holcomb	985.4	1.84
Miriam Delaney	233.6	18.68
Kuame Rich	202.6	29.78
Gisela Christensen	1028.5	17.44
Karyn Pacheco	1003.2	15.93
Hadassah Melton	946.8	16.67
Reese Black	1649.9	12.35
Tate Gray	973.3	14.22
Demetrius Kirk	1572.2	15.08
Knox Schwartz	538.5	16.41
Vincent Daniels	1893.9	10.56
Kieran Johnston	1234.8	13.53
Clio Chan	607.3	4.51
Barry Chaney	388.2	5.67

Query7		
Fullname	**InsuredValue**	**Weight**
Veda Bennett	751.3	19.96
Knox Schwartz	1866.1	24.95
Chastity Turner	184.6	29.35
Chastity Turner	139.3	12.74

QUESTION 1.8

Query8
Fullname
Alma Berg
Shana Woodward
Georgia Macias
Hop Hull
Clarke May
Elvis Stewart
Benjamin Shannon
Colleen Mann
Nayda Rose
Davis William
Aimee Osborne
Asher Graves

QUESTION 1.9

Query9						
BaggageID	**PassengerID**	**Weight**	**CheckInCounter**	**BaggageType**	**InsuredValue**	**Fragile**
157491	2	12.11	6	Sporting Equipment	1784.2	True
240402	11	19.57	6	Suitcase	2326.2	True
335841	38	17.95	6	Suitcase	2255.6	True
596430	6	8.72	6	Sporting Equipment	1746.8	True
615505	47	23.01	6	Sporting Equipment	2207.8	True
723694	38	12.35	6	Other	1649.9	True
989365	37	10.35	6	Suitcase	2014.6	True

SECTION B OBJECT ORIENTED PROGRAMMING

FINAL OUTPUT:

```
217085910      Barbara Knapp  R 1639.6599999999999
LK536  JNB      2014-11-02 08:25     CPT 2014-11-02 10:30
CA177  CPT      2014-11-10 18:00     JNB 2014-11-10 20:00

121068396      Jemima Knight  R 2012.12
YP794  DUR      2014-11-06 09:55     PLZ 2014-11-06 11:00
YP689  PLZ      2014-11-12 16:45     DUR 2014-11-12 17:45

273876910      Leandra Albert R 1692.79
CA916  JNB      2014-11-10 07:05     CPT 2014-11-10 09:10
CA177  CPT      2014-11-12 18:00     JNB 2014-11-12 20:00

229802502      Sierra Wade    R 2164.83
AS976  PLZ      2014-11-17 10:05     CPT 2014-11-17 12:10
DM137  CPT      2014-11-26 19:00     PLZ 2014-11-26 20:45

354575780      Wanda Hutchinson      R 1692.79
CA916  JNB      2014-11-18 07:05     CPT 2014-11-18 09:10
CA177  CPT      2014-11-19 18:00     JNB 2014-11-19 20:00

174303268      Shoshana Shepard      R 1639.6599999999999
LK536  JNB      2014-11-18 08:25     CPT 2014-11-18 10:30
CA177  CPT      2014-11-20 18:00     JNB 2014-11-20 20:00

247320475      Oliver Mcbride R 1639.6599999999999
LK536  JNB      2014-11-25 08:25     CPT 2014-11-25 10:30
CA177  CPT      2014-11-27 18:00     JNB 2014-11-27 20:00

270790486      Alisa Wilcox   R 1780.73
WJ455  JNB      2014-11-29 11:45     PLZ 2014-11-29 13:15
WJ377  PLZ      2014-12-08 20:15     JNB 2014-12-08 21:40

312577476      Germane Owens  R 1780.73
WJ455  JNB      2014-12-01 11:45     PLZ 2014-12-01 13:15
WJ377  PLZ      2014-12-02 20:15     JNB 2014-12-02 21:40

231340565      Tate Miranda   R 1692.79
CA916  JNB      2014-12-06 07:05     CPT 2014-12-06 09:10
CA177  CPT      2014-12-10 18:00     JNB 2014-12-10 20:00

175719020      Jillian Vincent       R 1457.81
ZF401  PMB      2014-12-06 10:00     PLZ 2014-12-06 10:50
ZF843  PLZ      2014-12-07 15:05     PMB 2014-12-07 16:00

105255264      Jillian Vincent       R 2012.12
YP794  DUR      2014-12-08 09:55     PLZ 2014-12-08 11:00
YP689  PLZ      2014-12-12 16:45     DUR 2014-12-12 17:45

261279769      Colette Raymond       R 1639.6599999999999
LK536  JNB      2014-12-13 08:25     CPT 2014-12-13 10:30
CA177  CPT      2014-12-22 18:00     JNB 2014-12-22 20:00

347183897      Thor Hensley   R 1692.79
CA916  JNB      2014-12-19 07:05     CPT 2014-12-19 09:10
CA177  CPT      2014-12-23 18:00     JNB 2014-12-23 20:00

214506253      Wanda Hutchinson      R 1745.33
ZA751  PMB      2014-12-20 08:45     CPT 2014-12-20 10:20
ZA117  CPT      2014-12-25 19:10     PMB 2014-12-25 20:50

331491637      Oliver Mcbride R 2012.12
YP794  DUR      2014-12-23 09:55     PLZ 2014-12-23 11:00
YP689  PLZ      2014-12-27 16:45     DUR 2014-12-27 17:45

153518381      Tamara Booth   R 1985.81
DM628  PLZ      2014-12-26 07:45     CPT 2014-12-26 09:25
DM137  CPT      2015-01-02 19:00     PLZ 2015-01-02 20:45

279681919      Wanda Hutchinson      R 2037.81
TZ781  CPT      2014-12-28 06:30     JNB 2014-12-28 08:30
TZ839  JNB      2014-12-30 14:00     CPT 2014-12-30 16:05

202455310      Shoshana Shepard      R 1639.6599999999999
LK536  JNB      2015-01-01 08:25     CPT 2015-01-01 10:30
```

```
CA177   CPT      2015-01-05 18:00      JNB 2015-01-05 20:00

302014884       Wanda Hutchinson       R 1985.81
DM628   PLZ      2015-01-03 07:45      CPT 2015-01-03 09:25
DM137   CPT      2015-01-08 19:00      PLZ 2015-01-08 20:45

311989762       Brendan Montgomery     R 2164.83
AS976   PLZ      2015-01-05 10:05      CPT 2015-01-05 12:10
DM137   CPT      2015-01-15 19:00      PLZ 2015-01-15 20:45

186830824       Leandra Albert R 1985.81
DM628   PLZ      2015-01-06 07:45      CPT 2015-01-06 09:25
DM137   CPT      2015-01-08 19:00      PLZ 2015-01-08 20:45

266914966       Jemima Knight  R 1639.6599999999999
LK536   JNB      2015-01-07 08:25      CPT 2015-01-07 10:30
CA177   CPT      2015-01-15 18:00      JNB 2015-01-15 20:00

199155758       Germane Owens  R 1639.6599999999999
LK536   JNB      2015-01-10 08:25      CPT 2015-01-10 10:30
CA177   CPT      2015-01-13 18:00      JNB 2015-01-13 20:00

230923170       Wanda Hutchinson       R 2027.81
PQ842   CPT      2015-01-10 11:00      JNB 2015-01-10 13:00
TZ839   JNB      2015-01-12 14:00      CPT 2015-01-12 16:05

298856833       Tamara Booth   R 1780.73
WJ455   JNB      2015-01-12 11:45      PLZ 2015-01-12 13:15
WJ377   PLZ      2015-01-15 20:15      JNB 2015-01-15 21:40

286124279       Leandra Albert R 1639.6599999999999
LK536   JNB      2015-01-14 08:25      CPT 2015-01-14 10:30
CA177   CPT      2015-01-15 18:00      JNB 2015-01-15 20:00

151832419       Brendan Montgomery     R 1457.81
ZF401   PMB      2015-01-16 10:00      PLZ 2015-01-16 10:50
ZF843   PLZ      2015-01-21 15:05      PMB 2015-01-21 16:00

343584357       Cleo Richardson        R 1457.81
ZF401   PMB      2015-01-16 10:00      PLZ 2015-01-16 10:50
ZF843   PLZ      2015-01-21 15:05      PMB 2015-01-21 16:00

227356806       Alisa Wilcox   R 2027.81
PQ842   CPT      2015-01-19 11:00      JNB 2015-01-19 13:00
TZ839   JNB      2015-01-21 14:00      CPT 2015-01-21 16:05

161002239       Thor Hensley   R 2027.81
PQ842   CPT      2015-01-21 11:00      JNB 2015-01-21 13:00
TZ839   JNB      2015-01-25 14:00      CPT 2015-01-25 16:05

351511045       Jillian Vincent        R 2027.81
PQ842   CPT      2015-01-21 11:00      JNB 2015-01-21 13:00
TZ839   JNB      2015-01-23 14:00      CPT 2015-01-23 16:05

111864400       Barbara Knapp  R 2164.83
AS976   PLZ      2015-01-24 10:05      CPT 2015-01-24 12:10
DM137   CPT      2015-01-25 19:00      PLZ 2015-01-25 20:45

213780740       Brendan Montgomery     R 1457.81
ZF401   PMB      2015-01-27 10:00      PLZ 2015-01-27 10:50
ZF843   PLZ      2015-02-02 15:05      PMB 2015-02-02 16:00

175216886       Wanda Hutchinson       R 1745.33
ZA751   PMB      2015-01-28 08:45      CPT 2015-01-28 10:20
ZA117   CPT      2015-02-03 19:10      PMB 2015-02-03 20:50

327995425       Cleo Richardson        R 2164.83
AS976   PLZ      2015-02-13 10:05      CPT 2015-02-13 12:10
DM137   CPT      2015-02-15 19:00      PLZ 2015-02-15 20:45

314714349       Sierra Wade    R 1457.81
ZF401   PMB      2015-02-15 10:00      PLZ 2015-02-15 10:50
ZF843   PLZ      2015-02-17 15:05      PMB 2015-02-17 16:00

118685037       Cleo Richardson        R 1745.33
ZA751   PMB      2015-02-19 08:45      CPT 2015-02-19 10:20
ZA117   CPT      2015-02-22 19:10      PMB 2015-02-22 20:50

220340491       Delilah Hebert R 2012.12
YP794   DUR      2015-02-20 09:55      PLZ 2015-02-20 11:00
```

```
YP689   PLZ      2015-02-21 16:45      DUR 2015-02-21 17:45

178245667       Oliver Mcbride R 1985.81
DM628   PLZ      2015-02-24 07:45      CPT 2015-02-24 09:25
DM137   CPT      2015-02-25 19:00      PLZ 2015-02-25 20:45

258369165       Cleo Richardson        R 1457.81
ZF401   PMB      2015-02-28 10:00      PLZ 2015-02-28 10:50
ZF843   PLZ      2015-03-04 15:05      PMB 2015-03-04 16:00

147011685       Sierra Wade    R 2164.83
AS976   PLZ      2015-03-04 10:05      CPT 2015-03-04 12:10
DM137   CPT      2015-03-07 19:00      PLZ 2015-03-07 20:45

307012187       Brendan Montgomery     R 1692.79
CA916   JNB      2015-03-06 07:05      CPT 2015-03-06 09:10
CA177   CPT      2015-03-16 18:00      JNB 2015-03-16 20:00

115907703       Teegan Lawrence        R 1692.79
CA916   JNB      2015-03-10 07:05      CPT 2015-03-10 09:10
CA177   CPT      2015-03-11 18:00      JNB 2015-03-11 20:00

192603055       Alisa Wilcox   R 1745.33
ZA751   PMB      2015-03-18 08:45      CPT 2015-03-18 10:20
ZA117   CPT      2015-03-26 19:10      PMB 2015-03-26 20:50

115194993       Brendan Montgomery     R 1692.79
CA916   JNB      2015-03-21 07:05      CPT 2015-03-21 09:10
CA177   CPT      2015-03-30 18:00      JNB 2015-03-30 20:00

286692016       Thor Hensley   R 2012.12
YP794   DUR      2015-03-22 09:55      PLZ 2015-03-22 11:00
YP689   PLZ      2015-03-23 16:45      DUR 2015-03-23 17:45

266351838       Oliver Mcbride R 2164.83
AS976   PLZ      2015-03-22 10:05      CPT 2015-03-22 12:10
DM137   CPT      2015-03-25 19:00      PLZ 2015-03-25 20:45

346089136       Alisa Wilcox   R 1985.81
DM628   PLZ      2015-03-27 07:45      CPT 2015-03-27 09:25
DM137   CPT      2015-03-30 19:00      PLZ 2015-03-30 20:45

210583460       Germane Owens  R 2012.12
YP794   DUR      2015-03-27 09:55      PLZ 2015-03-27 11:00
YP689   PLZ      2015-03-30 16:45      DUR 2015-03-30 17:45

329268658       Oliver Mcbride R 2164.83
AS976   PLZ      2015-03-28 10:05      CPT 2015-03-28 12:10
DM137   CPT      2015-04-04 19:00      PLZ 2015-04-04 20:45

229684837       Germane Owens  R 1985.81
DM628   PLZ      2015-04-01 07:45      CPT 2015-04-01 09:25
DM137   CPT      2015-04-02 19:00      PLZ 2015-04-02 20:45

133426853       Sierra Wade    R 2037.81
TZ781   CPT      2015-04-12 06:30      JNB 2015-04-12 08:30
TZ839   JNB      2015-04-18 14:00      CPT 2015-04-18 16:05

295073567       Aidan Blanchard        R 2037.81
TZ781   CPT      2015-04-14 06:30      JNB 2015-04-14 08:30
TZ839   JNB      2015-04-19 14:00      CPT 2015-04-19 16:05

159627487       Tate Miranda   R 1985.81
DM628   PLZ      2015-04-17 07:45      CPT 2015-04-17 09:25
DM137   CPT      2015-04-26 19:00      PLZ 2015-04-26 20:45

113612810       Wanda Hutchinson       R 1780.73
WJ455   JNB      2015-04-19 11:45      PLZ 2015-04-19 13:15
WJ377   PLZ      2015-04-28 20:15      JNB 2015-04-28 21:40

222967406       Alisa Wilcox   R 2164.83
AS976   PLZ      2015-04-21 10:05      CPT 2015-04-21 12:10
DM137   CPT      2015-04-23 19:00      PLZ 2015-04-23 20:45

171852007       Barbara Knapp  R 2164.83
AS976   PLZ      2015-04-30 10:05      CPT 2015-04-30 12:10
DM137   CPT      2015-05-02 19:00      PLZ 2015-05-02 20:45
```

```
335804770       Delilah Hebert R 1780.73
WJ455   JNB      2015-05-04 11:45      PLZ 2015-05-04 13:15
WJ377   PLZ      2015-05-07 20:15      JNB 2015-05-07 21:40

306450064       Thor Hensley   R 2012.12
YP794   DUR      2015-05-13 09:55      PLZ 2015-05-13 11:00
YP689   PLZ      2015-05-17 16:45      DUR 2015-05-17 17:45

330823134       Cleo Richardson        R 2037.81
TZ781   CPT      2015-05-14 06:30      JNB 2015-05-14 08:30
TZ839   JNB      2015-05-21 14:00      CPT 2015-05-21 16:05

350071139       Delilah Hebert R 1780.73
WJ455   JNB      2015-05-16 11:45      PLZ 2015-05-16 13:15
WJ377   PLZ      2015-05-18 20:15      JNB 2015-05-18 21:40

312969437       Tamara Booth   R 1457.81
ZF401   PMB      2015-05-20 10:00      PLZ 2015-05-20 10:50
ZF843   PLZ      2015-05-22 15:05      PMB 2015-05-22 16:00


Most Frequent Flyer
-------------------
Wanda Hutchinson 1505
```