NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2011

INFORMATION TECHNOLOGY: PAPER II

MARKING GUIDELINES

Time: 3 hours 120 marks

These marking guidelines were used as the basis for the official IEB marking session. They were prepared for use by examiners and sub-examiners, all of whom were required to attend a rigorous standardisation meeting to ensure that the guidelines were consistently and fairly interpreted and applied in the marking of candidates' scripts.

At standardisation meetings, decisions are taken regarding the allocation of marks in the interests of fairness to all candidates in the context of an entirely summative assessment.

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, and different interpretations of the application thereof. Hence, the specific mark allocations have been omitted.

```
// QUESTION 1

1.1   SELECT * FROM tblCountries ORDER BY country;                    (3)

1.2   SELECT * FROM tblEvents WHERE category = 'Swimming';            (3)

1.3   SELECT athlete FROM tblResults WHERE medal = 'Silver' OR medal =
      'Gold'(or any correct condition, -1 to a max of 2 for mistakes);  (4)

1.4   SELECT continent, COUNT(*) AS countries FROM tblCountries GROUP BY
      continent;        (4)

1.5   UPDATE tblCountries SET continent = 'Europe' WHERE code = 'RUS';  (3)

1.6   INSERT INTO tblResults (eventID, athlete, medal, code)(correct field
      list, 'record' is optional) VALUES (47, 'Kerron Stewart', 'Silver',
      'JAM') (correct values, appropriate for fields, -1 to a max of 2 for
      mistakes);  (4)

1.7   SELECT athlete FROM tblResults INNER JOIN tblEvents ON
      tblResults.eventID = tblEvents.eventID WHERE description = '100
      metres';                                                        (5)

1.8   SELECT country, medal(both country and medal must be present),
      COUNT(*) FROM tblResults INNER JOIN tblCountries ON tblResults.code =
      tblCountries.code GROUP BY country, medal;                      (7)

1.9   SELECT SUM(officials) * 500 AS Cost FROM tblEvents WHERE description
      LIKE '*freestyle*'(accept also '*freestyle' / Could also use
      RIGHT(description, 9)(both parameters correct) = 'freestyle');        (7)
                                                                      [40]
```

SECTION B:

```
NB: Program should output the values:
Before validation:
OR 8
WR 4
After validation:
OR 5
WR 3

// QUESTION 2 - JAVA
public class RaceResult
{
    private(at least one is) String(at least one is correctly typed) athlete(at
                                least one is correctly named);
    private String country; (all other instance vars correct)
    private int position;
    private String time;

    public RaceResult(name) (String a, String c, int p, String t) (parameters)
    {
        athlete = a; (assignments are correct)
        country = c;
        position = p;
        time = t;
    }

    public String(at least one) getAthlete ()(at least one named correctly)
    {
        return athlete; (at least one returns correctly)
    }

    public String getCountry ()(all other methods correct)
    {
        return country;
    }

    public int getPosition ()
    {
        return position;
    }

    public String getTime ()
    {
        return time;
    }

    public String toString ()(toString correctly typed)
    {
        return athlete + "(" + country + "): " + position; (format is as
                                required, -1 for each mistake, max of 2)
    }

}
```


Class name, at least one property is: private, correctly typed and correctly named; other instance vars correct, constructor named correctly, 4 parameters, parameter assignments correct; at least one accessor method: correctly typed, correctly named, correct return, all other accessors correct; toString correct, correct fields and format (-1 to a max of -2 for errors)

```
// QUESTION 3 - JAVA
public class OlympicRecord extends RaceResult
{
    public OlympicRecord(named) (String a, String c, int p, String t)
                            (parameters)
    {
        super (a, c, p, t);(calls parent constructor)
    }

    public boolean isValid ()(type and name)
    {
        if (getPosition () == 1)(if-statement correct)
        {
            return true;(returns correct true/false values)
        }
        else
        {
            return false;
        }
    }

    // QUESTION 6.1 - JAVA
    public double getTimeInSeconds ()
    {
        double sec = 0;
        String time = getTime ();

        if (time.contains ("m"))(check to see how to parse)
        {
            sec = Integer.parseInt (time.substring (0, time.indexOf ("m"))) *
                            60;(extract minutes)
            sec += Double.parseDouble (time.substring (time.indexOf ("m") +
                            1)); (extract seconds)
        }
        else
        {
            // just seconds
            sec = Double.parseDouble (time); (extract seconds)
        }
        return sec (double value);
    }

    // QUESTION 6.2 - JAVA
    public String toString ()
    {
        return super.toString () + " " + getTimeInSeconds ()
                            (mark for append only, candidate is penalised in
                            4.3 for incorrect override of toString);
    }
```



```
}
```

Question 3:

3.1 Class name OlympicRecord, extends RaceResult (2)

3.2 Constructor named correctly, 4 parameters, calls parent constructor with relevant parameters (3)

3.3 Method named isValid, if-statement condition (getPosition() == 1) , return appropriate values

[9]

Question 6:

6.1 Determine whether contains "m", extract minutes extract seconds correct calculation (m*60+s), return double value of seconds (5)

6.2 Correct method header, mark for appending getTimeInSeconds method results to toString (2)

[7]

```
// QUESTION 4 - JAVA
public class WorldRecord extends OlympicRecord(same method as WR extends, even
                            if wrong)
{
    public WorldRecord (String a, String c, int p, String t) (method header
                            correct)
    {
        super (a, c, p, t) (use parent constructor);
    }

    public String toString ()
    {
        return super.toString () + " World Record";
    }
}
```

Question 4:

4.1 Class extends OlympicRecord (1)

4.2 Constructor method is named correctly, uses parent constructor. (2)

4.3 toString method correct header, call parent toString, append "World Record" text. (3)

[6]

```
// QUESTION 5 - JAVA
import java.io.BufferedReader;
import java.io.FileReader;

public class RecordManager
{
    private RaceResult [] results = new RaceResult[500];
    private int counter = 0;

    public RecordManager ()
    {
        BufferedReader br = null;
        try
        {
```

```
            br = new BufferedReader (new FileReader ("resultdata.txt"));(open
                                file)

            String line = br.readLine ();
            while (line != null)(correct loop)
            {
                String[] details = line.split ("#")(file line parse);
                RaceResult temp = null;
                if (details[3].equals ("---"))(determine type)
                {
                    temp = new RaceResult (details[0], details[1],
                              Integer.parseInt (details[2])(int parse),
                              details[4])(parameters);
                }
                else if (details[3].equals ("OLR"))
                {
                    temp = new OlympicRecord (details[0], details[1],
                              Integer.parseInt (details[2]), details[4]) ;
                }
                else if (details[3].equals ("WRC"))
                {
                    temp = new WorldRecord (details[0], details[1],
                              Integer.parseInt (details[2]), details[4]) ;
                }
                results[counter] = temp;
                counter ++;
                line = br.readLine ();
            }
        }
        catch (Exception ex)
        {
            // ignore
        }
    }

    public String toString ()(toString method)
    {
        String temp = "";
        for (int i = 0; i < counter; i ++)(correct loop)
        {
            temp = temp +(append) results[counter].toString ()(each element's
                              toString) + "\n";(newline character)
        }
        return temp;
    }

    public void delete (int position)
    {
        for (int i = position; i < counter; i ++)(correct loop)
        {
            results[i] = results[i + 1];(correct shift)
        }
        counter --;
    }

    public int countOlympicRecords ()(method header)
```



```
    {
        int ORcount = 0;
        for (int i = 0; i < counter; i ++)(loop)
        {
            if (results[i] instanceof OlympicRecord)(correct check)
            {
                ORcount ++(increment);
            }
        }
        return ORcount(return);
    }

    public int countWorldRecords ()(same idea, different class)
    {
        int WRcount = 0;
        for (int i = 0; i < counter; i ++)
        {
            if (results[i] instanceof WorldRecord)
            {
                WRcount ++;
            }
        }
        return WRcount;
    }

    public void validate ()(method header)
    {
        for (int i = 0; i < counter; i ++)(iterate through array)
        {
            if (results[i] instanceof OlympicRecord) (OR check)
            {
                if (!((OlympicRecord) results[i])(type cast).isValid
                            ()(validity – remember 'not'))
                {
                    delete (i)(delete);
                }
            }
        }
    }
}
```

Question 5:

5.1 Class called RecordManager (1)

5.2 Array of RaceResults declared, 500 elements, int counter. (4)

5.3 Constructor method RecordManager, opens file for reading, loops correctly, parses line for '#' character, determines type of record ('---', 'OLR', 'WRC'), creates appropriate objects with correct parameters for at least one, adds to array, increments counter, reads in next line for parsing. (12)

5.4 toString method added, for-loop through elements, appent to temp String using object's toString, add new line character. (5)

5.5 Method called delete (or appropriate name, "remove" etc.) , accepts integer parameter, correct for-loop, shift elements , decrease counter. (5)

5.6 Method given appropriate name (e.g. countOlympicRecords) , for-loop, "instanceof" check, increment counter, return counter. Second method (countWorldRecords) employs same logic. (6)

5.7 Method called validate, iterates through array, checks for "instanceof" OlympicRecord, type-casts correctly, checks isValid method, uses delete method (Q5.5) to remove element from array. (6)

[39]

```
// QUESTION 7 - JAVA
public class Interface
{
    public static void main (String[] args)
    {
        RecordManager rm = new RecordManager ()(instantiation);

        System.out.println (rm.toString ()(generate toString output));
        System.out.println ("OR: " + rm.countOlympicRecords ());
        System.out.println ("WR: " + rm.countWorldRecords ())(output at both WR
                              and OR at least once each);
        System.out.println ("Validating data...");
        rm.validate ()(perform validation);
        System.out.println ("OR: " + rm.countOlympicRecords ());
        System.out.println ("WR: " + rm.countWorldRecords ());
    }
}
```

7 Instantiates a RecordManager object, calls toString method, outputs results of both countOR and countWR methods at least once, validates data. (4)

QUESTION 2 – DELPHI

```
NB: Program should output the values:
Before validation:
OR 8
WR 4
After validation:
OR 5
WR 3
```

```
unit uRaceResult;

interface

uses SysUtils;

type TRaceResult = class
  private
    athlete : String;
    country : String;
    position : Integer;
    raceTime : String;
  public
    constructor Create(a,c : String; p : Integer; rt : String); virtual;
    function getAthlete : String;
    function getCountry : String;
    function getPosition : Integer;
    function getTime : String;
    function toString : String; virtual;
end;

implementation

{ TRaceResult }

constructor TRaceResult.Create (a, c: String; p: Integer; rt: String);
begin
  athlete := a;
  country := c;
  position := p;
  raceTime := rt;
end;

function TRaceResult.getAthlete: String;
begin
  Result := athlete;
end;

function TRaceResult.getCountry: String;
begin
  Result := country;
end;


function TRaceResult.getPosition: Integer;
begin
  Result := position;
end;
```

```
function TRaceResult.getTime: String;
begin
  Result := raceTime;
end;

function TRaceResult.toString: String;
begin
  Result := getAthlete + ' (' + getCountry + '): ' +
IntToStr(getPosition);
end;

end.
```

Class name, at least one property is: private, correctly typed and correctly named; other instance vars correct, constructor named correctly, 4 parameters, parameter assignments correct; at least one accessor method: correctly typed, correctly named, correct return, all other accessors correct; toString correct, correct fields and format (-1 to a max of -2 for errors)

QUESTION 3 & 6 – DELPHI

```
unit uOlympicRecord;

interface

uses uRaceResult, SysUtils;

type TOlympicRecord = class (TRaceResult)
  public
    constructor Create(a,c : String; p : Integer; rt : String);
override;
    function isValid : boolean;
    function getTimeInSeconds : real;
    function toString : String; override;
end;

implementation

{ TOlympicRecord }

constructor TOlympicRecord.Create (a, c: String; p: Integer; rt:
String);
begin
  inherited Create(a, c, p, rt);
end;

function TOlympicRecord.getTimeInSeconds: real;
var
  timeInSecs : real;
  t : String;
  posOfM : Integer;
begin
  t := getTime;
  posOfM := Pos('m', t);
  if (posOfM = 0) then
    begin
      timeInSecs := StrToFloat(t);
```

```
    end
  else
    begin
      timeInSecs := StrToInt(Copy(t, 1, posOfM - 1)) * 60;
      timeInSecs := timeInSecs + StrToFloat(Copy(t, posOfM + 1,
Length(t) - posOfM));
    end;
  Result := timeInSecs;
end;

function TOlympicRecord.isValid: boolean;
begin
  if  (getPosition = 1) then
    begin
      Result := true;
    end
  else
    begin
      Result := false;
    end;
end;

function TOlympicRecord.toString: String;
begin
  Result := inherited ToString + ' ' + FloatToStr(getTimeInSeconds);
end;

end.
```

Question 3:

3.1 Class name OlympicRecord, extends RaceResult (2)

3.2 Constructor named correctly, 4 parameters, calls parent constructor with relevant parameters (3)

3.3 Method named isValid, if-statement condition (getPosition() == 1) , return appropriate values

[9]

Question 6:

6.1 Determine whether contains “m”, extract minutes extract seconds correct calculation (m*60+s), return double value of seconds (5)

6.2 Correct method header, mark for appending getTimeInSeconds method results to toString (2)

[7]

QUESTION 4

```
unit uWorldRecord;

interface

uses uOlympicRecord;

type TWorldRecord = class (TOlympicRecord)
  public
    constructor Create(a,c : String; p : Integer; rt : String);
override;
    function toString : String; override;
end;

implementation
```

```
{ TWorldRecord }

constructor TWorldRecord.Create (a, c: String; p: Integer; rt: String);
begin
  inherited Create(a, c, p, rt);
end;

function TWorldRecord.toString: String;
begin
  Result := inherited toString + ' World Record';
end;

end.
```

Question 4:

4.1 Class extends OlympicRecord (1)

4.2 Constructor method is named correctly, uses parent constructor. (2)

4.3 toString method correct header, call parent toString, append "World Record" text. (3)

[6]

QUESTION 5

```
unit uRecordManager;

interface

uses uRaceResult, uOlympicRecord, uWorldRecord, SysUtils;

type TRecordManager = class
  private
    resArray : array [1..500]  of TRaceResult;
    count : integer;
  public
    constructor Create;
    function toString : String;
    procedure deleteResult(position : Integer);
    function getWRCount : Integer;
    function getORCount : Integer;
    procedure validate;
end;

implementation

{ TRecordManager }

constructor TRecordManager.Create;
var
  line : String;
  infile : textfile;
  athleteName, country, recordType, raceTime : String;
  position : Integer;
begin

  AssignFile(infile, 'ResultData.txt');
  Reset(inFile);

  count := 0;
```

```
  while NOT(EOF(inFile))  do
    begin
      Inc(count);
      ReadLn(inFile, line);

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

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

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

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

      raceTime := line;

      if (recordType = '---') then
        begin
          resArray[count]  := TRaceResult.Create(athleteName, country,
position, raceTime);
        end
      else if (recordType = 'OLR') then
        begin
          resArray[count] := TOlympicRecord.Create(athleteName, country,
position, raceTime);
        end
      else if (recordType = 'WRC') then
        begin
          resArray[count] := TWorldRecord.Create(athleteName, country,
position, raceTime);
        end;
    end;

  CloseFile(infile);

end;

procedure TRecordManager.deleteResult (position: Integer) ;
var
  loop : Integer;
begin
  for loop := position to count - 1 do
    begin
      resArray[loop] := resArray[loop + 1];
    end;

  Dec(count);
end;

function TRecordManager.getORCount: Integer;
var
  loop, totalOR : Integer;
begin
  totalOR := 0;

  for loop := 1 to count do
```

```
    begin
      if (resArray[loop] is TOlympicRecord) then
        begin
          Inc(totalOR);
        end;
    end;
    Result := totalOR;
end;

function TRecordManager.getWRCount: Integer;
var
  loop, totalWR : Integer;
begin
  totalWR := 0;

  for loop := 1 to count do
    begin
      if (resArray[loop] is TWorldRecord) then
        begin
          Inc(totalWR);
        end;
    end;

    Result := totalWR;
end;

function TRecordManager.toString: String;
var
  rString : String;
  loop : Integer;
begin
  rString := '';

  for loop := 1 to count do
    begin
      rString := rString + resArray[loop].toString + #13;
    end;

  Result := rString;
end;

procedure TRecordManager.validate;
var
  loop : Integer;
  tempOR : TOlympicRecord;
begin
  for loop := 1 to count do
    begin
      if (resArray[loop] is TOlympicRecord) then
        begin
          tempOR := resArray[loop] as TOlympicRecord;
          if (tempOR.isValid = false) then
            begin
              deleteResult(loop);
            end;
        end;
    end;
end;
```

```
end.
```


Question 5:

5.1 Class called RecordManager (1)

5.2 Array of RaceResults declared , 500 elements , int counter . (4)

5.3 Constructor method RecordManager , opens file for reading , loops correctly , parses line for '#' character , determines type of record ('---', 'OLR', 'WRC'), creates appropriate objects with correct parameters for at least one, adds to array , increments counter , reads in next line for parsing . (12)

5.4 toString method added , for-loop through elements , appent to temp String using object's toString , add new line character .(5)

5.5 Method called delete (or appropriate name, "remove" etc.) , accepts integer parameter , correct for-loop , shift elements , decrease counter . (5)

5.6 Method given appropriate name (e.g. countOlympicRecords) , for-loop , "instanceof" check , increment counter , return counter . Second method (countWorldRecords) employs same logic . (6)

5.7 Method called validate , iterates through array , checks for "instanceof" OlympicRecord , type-casts correctly , checks isValid method , uses delete method (Q5.5) to remove element from array . (6)

[39]

QUESTION 7

```
unit uFrmMain;

interface

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

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

var
  frmMain: TfrmMain;
```

```
implementation

{$R *.dfm}

procedure TfrmMain.FormActivate(Sender: TObject);
var
  rm : TRecordManager;
begin
  rm := TRecordManager.Create;

  rchOutput.Lines.Add(rm.toString);

  rchOutput.Lines.Add('Number of Olympic Records : ' +
IntToStr(rm.getORCount));
  rchOutput.Lines.Add('Number of World Records : ' +
IntToStr(rm.getWRCount));

  rm.validate;
  rchOutput.Lines.Add('Records validated');

  rchOutput.Lines.Add('Number of Olympic Records : ' +
IntToStr(rm.getORCount));
  rchOutput.Lines.Add('Number of World Records : ' +
IntToStr(rm.getWRCount));
end;

end.
```

7 Instantiates a RecordManager object, calls toString method, outputs results of both countOR and countWR methods at least once, validates data. (4)