NATIONAL SENIOR CERTIFICATE EXAMINATION
MAY 2024

INFORMATION TECHNOLOGY: PAPER I

MARKING GUIDELINES

Time: 3 hours 150 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

QUESTION 1.1

```
SELECT *
FROM tblChildren
ORDER BY Lastname
```

QUESTION 1.2

```
SELECT *
FROM tblActivities
WHERE CostPerPerson <=75 AND AllYear = TRUE;
```

Accept % for mysql

QUESTION 1.3

```
SELECT *
FROM tblActivities
WHERE ActivityDays LIKE '*Sa*';
```

QUESTION 1.4

```
SELECT Lastname, ParentCellphone, DOB
FROM tblChildren
WHERE (YEAR (NOW)  - YEAR(DOB)  = 13 AND MONTH(DOB) >= MONTH(NOW)
)  OR MONTH(DOB) IN (6 , 12)
```

QUESTION 1.5

```
SELECT TOP 3 *

FROM tblActivities

ORDER BY CostPerPerson DESC;
```

QUESTION 1.6

```
SELECT Description, COUNT(*) AS TotalSignUps, SUM(CostPerPerson)
as TotalAmount
FROM tblSignUps, tblActivities
WHERE tblSignUps.ActivityID = tblActivities.ActivityID
GROUP BY Description
```


QUESTION 1.7

```
UPDATE tblChildren
SET ParentCellphone =   '(' & LEFT(ParentCellphone, 3)  & ') ' &
MID(ParentCellphone, 4,3)  &'-' & MID(ParentCellphone,7,
LEN(ParentCellphone) )  CONCATENATION – CORRECT FORMAT
OR use 10 as all cellphones have 10 digits.
```

QUESTION 1.8

```
SELECT Lastname,Firstname,SUM(CostPerPerson) as AmountOwed
FROM tblChildren, tblSignUps, tblActivities
WHERE tblChildren.ChildID = tblSignUps.ChildID AND
tblActivities.ActivityID = tblSignUps.ActivityID
GROUP BY  Lastname,Firstname
HAVING SUM(CostPerPerson) > 250
```

QUESTION 1.9

```
INSERT INTO tblSignUps(ActivityID,ChildID,   SignUpDate)
SELECT ActivityID, ChildID, #2024/12/10#
FROM tblSignUps
WHERE ChildID in (1,5,7,9)
```


SECTION B OBJECT ORIENTATED PROGRAMMING

JAVA SOLUTION

QUESTION 2 SchoolClub.java

```
//Q2.1 - 4
//class header correct
public class SchoolClub {
    //all fields private
    //typed correctly
    //named correctly
    private String clubName;
    private LocalTime meetTime;
    private String teacher;

    //Q2.2 - 2
    //field declared as public static
    //and final - naming conventions MUST be followed as per class
diagram
    public static final double CLUBCOST = 135.75;


    //Q2.3 - 3
    //header correct
    //parameters named and typed correctly
    public SchoolClub( String inCN, LocalTime inMT, String inTR)
    {
        //fields assigned correctly
        clubName = inCN;
        meetTime = inMT;
        teacher = inTR;
    }

    //Q2.4 - 3
For all 3:
    //header return type correct
    //named correctly
    //returning correct instance variable and type
    public String getClubName() {
        return clubName;
    }

    public LocalTime getMeetTime() {
        return meetTime;
    }

    public String getTeacher() {
        return teacher;
    }
```



```
    //Q2.5 - 4
    //header correct
    public String toString()
    {
        //contains all fields
        //formatting correct
        //return correct
        return clubName + " @ " + meetTime +  " Teacher: " +
teacher;
    }
}
```


QUESTION 3 ExternalClub.java

```
//Q3.1 - 2
//class named correctly
//extends SchoolClub
public class ExternalClub extends SchoolClub{

    //3.2 - 1
    //private fields typed and named correctly
    private String manager;
    private double rate;

    //Q3.3 - 4
    //constructor header correct
    public ExternalClub(String inCN, LocalTime inMT, String inTR ,
String inMR, double inRT)
    {
        //parent constructor called
        //correct arguments to parent constructor
        super(inCN, inMT, inTR);
        //child class fields assigned using parameters
        manager = inMR;
        rate = inRT;
    }

    //Q3.4 - 6
    //header correct
    public double getExternalClubCost()
    {
        //using constant from parent class
        //dividing rate by 100
        //formula correct
        //round to two decimals
        //return real number
        return Math.round( SchoolClub.CLUBCOST * (rate /
100.0)*100)/100.0 ); //or use DecimalFormat
    }

    //Q3.5 - 4
    //header correct and matching return
    public String toString()
    {
        //calling toString from parent
        //append manager field
        // method call - getExternalClubCost
        return super.toString() + " Manager: " + manager + " R" +
getExternalClubCost();
    }

}
```


QUESTION 4 & 6.1 ClubManager.java

```
//Q4.1 - 1
//class header
public class ClubManager {

    //Q4.2 - 4
    //both fields private
    //array of 20  objects
    //type of parent SchoolClubs
    //size field correct type
    private SchoolClub cArr[] = new SchoolClub[20];
    private int size = 0;

    //Q4.3 - 11
    //constructor header
    public ClubManager()
    {
        try
        {
            //open file
            Scanner sc = new Scanner(new File("clubs.txt"));

            //loop through file
            while(sc.hasNextLine())
            {
                //read line from text file
                String line = sc.nextLine();
                //extract fields – any method
                String tokens[] = line.split("#");
                String clubName = tokens[0];
                //create appropriate time object
                LocalTime meetTime = LocalTime.parse(tokens[1]);
                String teacher = tokens[2];

                //check for ExternalClub
                if(tokens.length > 3)
                {
                    //get manager and rate
                    String manager = tokens[3];
                    double rate = Double.parseDouble(tokens[4]);
                    //create ExternalClub object and add to array
                    ExternalClub e = new ExternalClub( clubName,
meetTime, teacher, manager, rate);
                    cArr[size] = e;
                }
                else
                {
                    //create SchoolClub object and add to array
                    SchoolClub c = new SchoolClub(clubName,
meetTime, teacher);
                    cArr[size] = c;
                }
```

```
                //increment size
                size++;
            }

            sc.close();
        }
        catch(FileNotFoundException fne)
        {
            System.out.println("File Missing " + fne);
        }
    }

    //Q4.4 - 4
    //header correct and return string
    public String toString()
    {
        String r = "";
        //loop through array - check if size decremented in
constructor and loop accordingly
        for (int i = 0; i < size; i++) {
            //append to string
            //add blank line
            r = r + cArr[i] + "\n";
        }

        return r;
    }

    //Q4.5 - 8
    //header correct
    public void sortByMeetTime()
    {
        //outside loop correct
        for (int i = 0; i < size -1; i++) {
            //inside loop correct
            for (int j = i+1; j < size; j++) {
                //compare correct elements
                //using meet time

if(cArr[i].getMeetTime().isAfter(cArr[j].getMeetTime()))
                {
                    //temp with correct type
                    SchoolClub temp = cArr[i];
                    //swop
                    cArr[i] = cArr[j];
                    cArr[j] = temp;
                }
            }
        }
    }

    //Q6.1 - 30
    //header correct
    public String checkClubs()
```

```
    {
        // initialize list for all school club clashes
        String list = "";
        *Alternative initialize with heading added*
        // initialize list for external club clashes
        String eList = "";




        //loop through array
        for (int i = 0; i < size; i++) {

            //storing clashes for school club
            int clashes = 0;
            int eClashes = 0;
            //get the meetTime
            LocalTime endTime = cArr[i].getMeetTime();
            //add the 45 min club end time accounting for
inclusive check
            endTime = endTime.plusMinutes(46);

            //loop through other club meeting times
            for (int j = i+1; j < size ; j++) {
                //checking for external clubs for both the current
club and the one being checked against
                if(cArr[i] instanceof ExternalClub && cArr[j]
instanceof ExternalClub)
                {

                    //check if a club falls in the range
                    //or if they have the same start time
                    if(
(cArr[i].getMeetTime().isBefore(cArr[j].getMeetTime()) &&
cArr[j].getMeetTime().isBefore(endTime) ) ||
cArr[i].getMeetTime().equals(cArr[j].getMeetTime()))
                    {
                        //add clashing clubs
                        eClashes++;

                    }

                }
                //checking for only SchoolClub objects to match
                else if ( !(cArr[i] instanceof ExternalClub) &&
!(cArr[j] instanceof ExternalClub) )
                {
                     //check if a club falls in the range
                    //or if they have the same start time
                    if(
(cArr[i].getMeetTime().isBefore(cArr[j].getMeetTime()) &&
cArr[j].getMeetTime().isBefore(endTime) ) ||
cArr[i].getMeetTime().equals(cArr[j].getMeetTime()))
                    {
```

```
                        //add clashing clubs
                        clashes++;

                    }

                }


            }

            //check again for ExternalClub
            if(cArr[i] instanceof ExternalClub)
            {
                //Casting to ExternalClub
                ExternalClub temp = (ExternalClub) cArr[i];
                //check if there were clashes
                if(eClashes > 0)
                {
                    //append to list with heading
                    //appropriate formatting
                    eList = eList + temp.getClubName() + "\n-
Clashes with " + eClashes + " other clubs\n";
                }
                else
                {
                    //append no clashes
                    eList = eList + temp.getClubName() + "\n==NO
CLASHES==\n";
                }
                //calculating the venue cost using
getExternalClubCost
                //correct calculation
                //rounding
                //appending
                eList = eList + "Venue Cost: R" + Math.round(
temp.getExternalClubCost() * 0.15 * 100) / 100.0 + "\n\n";
            }
            else
            {
                //check school based club clashes
                if(clashes > 0)
                {
                    //append to list with heading
                    list = list + cArr[i].getClubName() + "\n-
Clashes with " + clashes + " other clubs\n\n";

                }
                else
                {
                    //append no clashes
                    list = list + cArr[i].getClubName() + "\n==NO
CLASHES==\n\n";
                }
            }
```

```
        }

        //return both lists with appropriate headings
        return "School Clubs \n" + list + "\n\n" + "External Clubs
\n" + eList;

    }



}
```

QUESTION 5 & 6.2 ClubUI.java

```
//Q5.1 - 1
//Class header correct
public class ClubUI {
    public static void main(String[] args) {

        //Q5.2 - 2
        //ClubManager object created
        //in correct place
        ClubManager cm = new ClubManager();

        //Q5.3 - 2
        //call toString method
        // in output statement
        System.out.println(cm.toString());

        //5.4 - 2
        //sort method called
        cm.sortByMeetTime();
        //toString called
        System.out.println(cm);

        //Q6.2 - 2
        //correct method name
        //in an output statement (typed method)
        System.out.println(cm.checkClubs());


    }
}
```

SECTION B OBJECT ORIENTATED PROGRAMMING

DELPHI SOLUTION

QUESTION 2 uSchoolClub.pas

```
unit uSchoolClub;

interface
uses SysUtils, DateUtils;

//Q2.1 - 4
//class header correct
type tSchoolClub = class
  private
  //all fields private
  //typed correctly
  //named correctly
  clubName : string;
  meetTime : TTime;
  teacher : string;

  public
    const
      //Q2.2 - 2
      //field declared as public
      //and final
      CLUBCOST = 135.75;

    constructor Create(inCN : string; inMT : TTime ; inTR :
string);
    function getClubName() : string;
    function getMeetTime() : TTime;
    function getTeacher() : string;
    function toString() : string; virtual;
end;

implementation

{ TSchoolClub }

//Q2.3 - 3
//header correct
//parameters named and typed correctly
constructor TSchoolClub.Create(inCN: string; inMT: TTime; inTR:
string);
begin
//fields assigned correctly
  clubName := inCN;
  meetTime := inMT;
  teacher := inTR;
end;
```



```
 //Q2.4 - 3
For all 3:

//header return type correct
//named correctly
//returning correct instance variable and type

function TSchoolClub.getClubName: string;
begin
  Result := clubName;
end;

function TSchoolClub.getMeetTime: TTime;
begin
  Result := meetTime;
end;

function TSchoolClub.getTeacher: string;
begin
  Result := teacher;
end;

//Q2.5 - 4
//header correct
function TSchoolClub.toString: string;
begin
  //contains all fields
  //formatting correct
  //return correct
  Result := clubName + ' @ ' +TimeToStr(meetTime) + ' Teacher:' +
teacher;
end;

end.
```

QUESTION 3 uExternalClub.pas

```
unit uExternalClub;

interface
uses SysUtils, DateUtils ,  uSchoolClub;
//Q3.1 - 2
//class named correctly
//extends SchoolClub
type tExternalClub = class(tSchoolClub)
  private
    //3.2 - 1
    // private fields typed and named correctly
    manager : string;
    rate : double;

    public
    constructor  Create(inCN : string; inMT : TTime ; inTR :
string ; inMR:string; inRT:double);
    function getExternalClubCost() : double;
    function toString : string; override;
end;
implementation

{ tExternalClub }
 //Q3.3 - 4
//constructor header correct
constructor tExternalClub.Create(inCN: string; inMT: TTime; inTR,
  inMR: string; inRT: double);
begin
  //parent constructor called
  //correct parameters given
  Inherited Create(inCN, inMT, inTR);
  //child class fields assigned using parameters
  manager := inMR;
  rate := inRT;
end;
//Q3.4 - 6
//header correct
function tExternalClub.getExternalClubCost: double;
begin
  //using constant from parent class
  //dividing rate by 100
  //formula correct
  //round to two decimals
  //return real number
  Result := Round(tSchoolClub.CLUBCOST * rate/100 * 100)/100;
end;


//Q3.5 - 4
//header correct and return
function tExternalClub.toString: string;
begin
```

```
  //calling toString from parent
  //append manager field
  //calling method
  Result:= Inherited toString + ' Manager: ‘ + manager + ‘R' +
FloatToStr(getExternalClubCost());
end;

end.
```

QUESTION 4 & 6.1 uClubManager.pas

```
unit uClubManager;

interface

uses SysUtils, DateUtils ,  uSchoolClub , uExternalClub;
 //Q4.1 - 1
//class header
type tClubManager = class
  private
   //Q4.2 - 4
  // both fields private
  //array of 20
  //type of parent SchoolClubs
  //size field correct type
  cArr : array[1..20] of tSchoolClub;
  size : integer;

  public
  constructor Create();
  function toString(): string;
  procedure sortByMeetTime();
  function checkClubs() : string;
end;

implementation

{ tClubManager }


//Q4.3 - 11
//constructor header
constructor tClubManager.Create;
var
 inFile : textFile;
 line, clubName, mt, teacher, manager, h, m : string ;
 meetingTime : TTime;
 rate : double;
begin
//open file
  if FileExists('clubs.txt') <> true then
  begin
    WriteLn('File Missing');
  end
  else
  begin
    AssignFile(inFile ,'clubs.txt');
    Reset(inFile);

    size:= 0;
    //loop through file
    while NOT EOF(inFile) do
    begin
```

```
      //read line from text file
      ReadLn(inFile,line);
      //increment size
      Inc(size);

       //extract fields correctly
      clubName := Copy(line , 1 , Pos('#', line) - 1);
      Delete(line , 1 , Pos('#',line));

      mt := Copy(line , 1 , Pos('#', line) - 1);
      Delete(line , 1 , Pos('#',line));
        //create appropriate time object
      meetingTime := StrToTime(mt + ':00');

       //check for ExternalClub
      if Pos('#', line) > 0 then
          begin
           //get manager and rate
            teacher := Copy(line , 1 , Pos('#', line) - 1);
            Delete(line , 1 , Pos('#',line));
            manager := Copy(line , 1 , Pos('#', line) - 1);
            Delete(line , 1 , Pos('#',line));
            rate := StrToFloat(line);
              //create ExternalClub object and add to array
            cArr[size] :=
TExternalClub.Create(clubName,meetingTime, teacher, manager,
rate);
          end
          else
          begin
            //create SchoolClub object and add to array
            teacher := line;
            cArr[size] := TSchoolClub.Create(clubName,meetingTime,
teacher);
          end;


    end;

  end;

end;

//Q4.4 - 4
//header correct and return string
function tClubManager.toString: string;
var
  i : integer;
  output : string;
begin
   output := '';
   //loop through array
  for i := 1 to size do
    begin
```

```
      //append to string
      //add blank line
      output := output + cArr[i].toString() + #13#10 + #13#10;
    end;

  Result := output;
end;


//Q4.5 - 8
//header correct
procedure tClubManager.sortByMeetTime;
var
 i , j : integer;
 temp : TSchoolClub;
begin
    //outside for loop correct
  for i := 1 to size do
  begin
    //inside for loop correct
    for j := 1 to size-1 do
      begin
        //compare correct elements
        //using meet time
        if CompareTime(cArr[j].getMeetTime() ,
cArr[j+1].getMeetTime()) > 0 then
        begin
            //temp with correct type
            //swop
             temp := cArr[j];
             cArr[j] := cArr[j+1];
             cArr[j+1] := temp;
        end;

      end;
  end;
end;

//Q6.1 - 30
//header correct
function tClubManager.checkClubs: string;
var
  list, eList: string;
  i, j, clashes, eClashes: Integer;
  endTime: TTime;
  temp : TExternalClub;
begin
    //initialize list for all school club clashes
    list := '';
    // initialize list for external club clashes
    eList := '';
    //loop through array
    for i := 1 to size do
    begin
```

```
          //storing clashes for school club
          clashes := 0;
          eClashes := 0;
          //get the meetTime
          endTime := cArr[i].getMeetTime();
          //add the 45 min club end time accounting for inclusive
check
          endTime := IncMinute(endTime, 46);
          //loop through other club meeting times
          for j := i + 1 to size do
          begin
          //checking for external clubs for both the current club
and the one being checked against
           if (cArr[i] is TExternalClub) AND (cArr[j] is
TExternalClub) then
          begin
            //check if a club falls in the range
            //or if they have the same start time
            if  ( (
CompareTime(cArr[i].getMeetTime,cArr[j].getMeetTime) < 0 ) AND
(CompareTime(cArr[j].getMeetTime,endTime) < 0) ) OR (
CompareTime(cArr[i].getMeetTime,cArr[j].getMeetTime) = 0) then
            begin
              //increment clashing clubs
              Inc(eClashes);
            end;

          end
           //checking for only SchoolClub objects to match
          else if  NOT(cArr[i] is TExternalClub) AND NOT(cArr[j]
is TExternalClub) then
          BEGIN
            //check if a club falls in the range
            //or if they have the same start time
            if  ( (
CompareTime(cArr[i].getMeetTime,cArr[j].getMeetTime) < 0 ) AND
(CompareTime(cArr[j].getMeetTime,endTime) < 0) ) OR
(CompareTime(cArr[i].getMeetTime,cArr[j].getMeetTime) = 0) then
            begin
              //increment clashing clubs
              Inc(clashes);
            end;
          END;



          end;
          //check again for ExternalClub
          if cArr[i] is TExternalClub then
          begin
            //Casting to ExternalClub
            temp := cArr[i] as TExternalClub;
             //check if there were clashes
            if eClashes > 0 then
```



```
            begin
                 //append to list with heading
                 //appropriate formatting
                 eList := eList + temp.getClubName + #13#10 + '-
Clashes with ' + IntToStr(eClashes) + ' other clubs' + #13#10;
            end
            else
            begin
                 //append no clashes
                 eList := eList + temp.getClubName + #13#10 +
'==NO CLASHES==' + #13#10;

            end;
             //calculating the venue cost using getExternalClubCost
             //correct calculation
             //rounding
             //appending
            eList := eList + 'Venue Cost ' + FloatToStr(
Round(temp.getExternalClubCost * 0.15 * 100)/100) + #13#10 +
#13#10;

         end
         else
         begin
             //check school based club clashes
             if clashes > 0 then
             begin
                //append to list with heading
                list := list + cArr[i].getClubName + #13#10 + '-
Clashes with ' + IntToStr(clashes) + ' other clubs' + #13#10+
#13#10
             end
             else
             begin
                //append no clashes
                list := list + cArr[i].getClubName() + #13#10  +
'==NO CLASHES==' + #13#10+ #13#10;
             end;

         end;


    end;
    //return both lists with appropriate headings
    Result := Result + 'School Clubs' + #13#10 + list +  #13#10 +
#13#10 + 'External Clubs' + #13#10 + eList;
end;

end.
```

QUESTION 5 & 6.2 ClubUI.dpr

```
//Q5.1 - 1
//Class header correct
program ClubUI;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils, DateUtils ,
  uSchoolClub in 'uSchoolClub.pas',
  uExternalClub in 'uExternalClub.pas',
  uClubManager in 'uClubManager.pas';

var

  cm : tClubManager;
begin
  try


     //Q5.2 - 2
    //ClubManager object created
    //in correct place
    cm := tClubManager.Create();
    //Q5.3 - 2
    //call toString method
    // in output method
    WriteLn(cm.toString());
     //5.4. - 2
    //sort method called
    cm.sortByMeetTime();
    //toString called
    WriteLn(cm.toString());

    //Q6.2 - 2
    //correct method name
    //in an output statment (typed method)
    WriteLn(cm.checkClubs());
    ReadLn;
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```