NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 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

Question 1.1

Access/MySQL/JavaDB:

```
SELECT *
FROM tblMusicians
WHERE Solo AND Province = "Gauteng"
Accept Solo = TRUE or any variation of yes/no
```

Question 1.2

Access/MySQL/JavaDB:

```
SELECT FestName, FestDate (both fields), Capacity * TicketCost AS
Income
FROM tblFestivals
```

Question 1.3

```
SELECT AVG (TicketCost)
FROM tblFestivals
WHERE Capacity < 10000
```

Question 1.4

Access/MySQL:

```
SELECT FestName, COUNT(*)
FROM tblFestivals
WHERE FestDate < NOW() Accept DATE()
GROUP BY FestName
HAVING COUNT(*) > 1
ORDER BY FestName

Accept ColumnAlias > 1 for MySQL
```

JavaDB:

```
SELECT FestName, COUNT(*)
FROM tblFestivals
WHERE FestDate < CURRENT_DATE
GROUP BY FestName
HAVING COUNT(*) > 1
ORDER BY FestName
```


Question 1.5

Access/MySQL:

```
SELECT LEFT (StageName, 3) OR (MID(StageName, 1, 3)
FROM tblMusicians
WHERE MusID NOT IN (SELECT MusID FROM tblBookings)
```

JavaDB:

```
SELECT SUBSTR (StageName, 1, 3)
FROM tblMusicians
WHERE MusID NOT IN (SELECT MusID FROM tblBookings)
```

ALTERNATE SOLUTION:

Access/MySQL:

```
SELECT LEFT (StageName, 3)
FROM tblMusicians LEFT JOIN tblBookings
ON tblMusicians.MusID = tblBookings.MusID
WHERE tblBookings.MusID IS NULL
```

JavaDB:

```
SELECT SUBSTR (StageName, 1, 3)
FROM tblMusicians LEFT JOIN tblBookings
ON tblMusicians.MusID = tblBookings.MusID
WHERE tblBookings.MusID IS NULL
```

Accept FestID or SetTime IS NULL

Question 1.6

Access:

```
SELECT FestName, FestDate & " " & SetTime
FROM tblFestivals, tblBookings
WHERE tblFestivals.FestID = tblBookings.FestID
```

MySQL:

```
SELECT FestName, CONCAT (FestDate, ' ', SetTime)
FROM tblFestivals, tblBookings
WHERE tblFestivals.FestID = tblBookings.FestID
```

JavaDB:

```
SELECT FestName, FestDate || ' ' || SetTime
FROM tblFestivals, tblBookings
WHERE tblFestivals.FestID = tblBookings.FestID
```

Accept INNER JOINS for all languages

Question 1.7

Access/MySQL/JavaDB:

```
SELECT FestName, FestDate, SUM (Fee)
FROM tblMusicians, tblFestivals, tblBookings
```



```
WHERE tblMusicians.MusID = tblBookings.MusID AND
tblFestivals.FestID = tblBookings.FestID
GROUP BY FestName, FestDate
ORDER BY FestDate DESC

ALTERNATE SOLUTION:
SELECT FestName, FestDate, SUM(Fee)
FROM tblMusicians INNER JOIN
    (tblFestivals INNER JOIN tblBookings ON tblFestivals.FestID =
tblBookings.FestID)
ON tblMusicians.MusID = tblBookings.MusID
GROUP BY FestName, FestDate
ORDER BY FestDate DESC
```

Question 1.8

Access/MySQL/JavaDB:

```
UPDATE tblFestival
SET TicketCost = 0
WHERE FestName = 'Coffee and Cake'
```

Question 1.9

Access/MySQL/JavaDB:

```
INSERT INTO tblFestivals (FestName, FestDate, Capacity,
                          TicketCost)
SELECT FestName, #2025-01-01#, Capacity, TicketCost + 20
FROM tblFestivals
WHERE FestName LIKE '*Fire* AND YEAR(FestDate)  = 2024
```

Accept FestDate >= #204-01-01# AND FestDate <= #2024-12-31#
Accept FestDate BETWEEN #2024-01-01# AND #2024-12-31#
Accept '2025-01-01' AND "%Fire%" for MySQL and JavaDB

SECTION B

JAVA SOLUTION

QUESTION 2

```
//Question 2
//Question 2.1 - 3
//class header
//all fields private
//all correctly typed with correct names
public class Song {
    private String artistName;
    private String title;
    private int likes;
    private int dislikes;

    //Question 2.2 - 3
    //correct header
    //correct parameter names and types and order
    //fields set to parameters

    public Song (String inAN, String inTitle, int inL,
                 int inD) {
        artistName = inAN;
        title = inTitle;
        likes = inL;
        dislikes = inD;
    }

    //Question 2.3 - 2
    //correct headers for all
    //correct return statements for all
    //if one or two accessors are missing and/or incorrect award
    only 1 mark
    public String getArtistName() {
        return artistName;
    }

    public String getTitle() {
        return title;
    }

    public int getLikes() {
        return likes;
    }

    //Question 2.4 - 3
    //correct header with correct return type
    //calculate likes - dislikes
    //return correct result
    public boolean getWellLiked() {
        return (likes - dislikes)  > 0;
    }
```



```
    ALTERNATE SOLUTION:
    public boolean getWellLiked()
        if (likes > dislikes)
            return true;
        else
            return false
OR: return likes > dislikes

    //Question 2.5 - 2
    //correct header
    //return all fields, correctly formatted
    public String toString() {
        return title + "\t" + likes + "\t" + dislikes;
    }
}
```

QUESTION 3

```
//Question 3
//Question 3.1 - 3
//class header
//all fields private
//all fields correctly typed with correct names
public class Musician {

    private String name;
    private String stageName;
    private int genre;
    private LocalDate start;

    //Question 3.2 - 2
    //constants declared with public static final and named and
    typed correctly
    //assigned correct value to correct name
    public static final int ROCK = 1;
    public static final int GQOM = 2;
    public static final int AMAPIANO = 3;
    public static final int TECHNO = 4;

    //Question 3.3 - 5
    //constructor named correctly
    //parameters correct and in correct order
    //assign correct values to name, genre and start
    //check for inStage empty
    //correctly assign inName or inStage to stageName
    public Musician (String inName, String inStage, int inGenre,
                     LocalDate inStart) {
        name = inName;
        genre = inGenre;
        start = inStart;
        if (inStage.equals("")) {
            stageName = inName;
        } else {
            stageName = inStage;
        }
    }

    //Question 3.4 - 2
    //correct headers for all accessors
    //correct returns for all accessors
    //if one accessor missing or incorrect award only 1 mark
    public String getName() {
        return name;
    }

    public LocalDate getStart() {
        return start;
    }
    //Question 3.5 - 4
    //selection (switch/if) on genre and use of constants
```



```
    //use of switch or nested if
    //"Uncategorised" returned for ANY value outside 1-4
    //return correct string for all other cases
    public String getGenre() {
        switch (genre)  {
            case ROCK:
                return "Rock";
            case GQOM:
                return "Gqom";
            case AMAPIANO:
                return "Amapiano";
            case TECHNO:
                return "Techno";
            default:
                return "Uncategorised";
        }
    }
```

ALTERNATE SOLUTION – accept this programming structure where if statements are used in condition with a return for every path.

```
    public String getGenre() {
        if (genre == ROCK)
            return "Rock";
        if (genre == GQOM)
            return "Gqom";
        if (genre == AMAPIANO)
            return "Amapiano";
        if (genre == TECHNO)
            return "Techno";
        return "Uncategorised";
    }
```

```
    //Question 3.6 - 2
    //correct header with parameter
    //assign parameter to field
    public void setGenre(int inGenre)  {
        genre = inGenre;
    }
```

```
    //Question 3.7 - 2
    //return all fields in a correctly formatted string
    //call to genGenre
    public String toString() {
        return stageName + "\t" + getGenre() + "\t" + start;
    }

}
```


QUESTION 4

```
//Question 4
//Question 4.1 - 1
//class created correctly
public class MusicManager {

    //Question 4.2 - 3
    //all properties private
    //both arrays declared and instantiated correctly with correct
    size
    //both sizes declared correctly - do not have to be
    initialized to 0
    private Musician[] musArr = new Musician[100];
    private int musSize = 0;
    private Song[] songArr = new Song[100];
    private int songSize = 0;

    //Question 4.3 - 15
    //Constructor header correct

    //open file for reading, accept Scanner, BufferredReader or
    any correct method
    //loop through first 7 lines
    //read next line from file
    //split line into required parts, accept any correct method
    (e.g. .split() or indexOf() with substring()
    //extract information from line
    //attempt to parsedate
    //parse correct date pattern
    //create Musician object
    //increment musSize

    //loop through all remaining lines
    //read next line from file
    //extract information from line
    //create Song object
    //increment songSize

    //try catch implemented correctly to handle exception with
    appropriate error message

    public MusicManager() {
       try {
           Scanner scFile = new Scanner(new File
                                        ("MusicData.txt"));
           for (int i = 0; i < 7; i++) {
               Scanner scLine = new Scanner(scFile.nextLine());
               scLine.useDelimiter(";");
               String name = scLine.next();
               String tag = scLine.next();
               int genre = scLine.nextInt();
               DateTimeFormatter dtf =
                  DateTimeFormatter.ofPattern("yyyy MM dd");
               LocalDate contractStart =
```

```
                LocalDate.parse(scLine.next(), dtf);
            musArr[musSize] = new Musician(name, tag, genre,
                                           contractStart);
            musSize++;
        }
        while (scFile.hasNextLine()) {
            Scanner scLine = new Scanner(scFile.nextLine());
            scLine.useDelimiter(";");
            String name = scLine.next();
            String title = scLine.next();
            int likes = scLine.nextInt();
            int dislikes = scLine.nextInt();
            songArr[songSize] = new Song(name, title, likes,
                                         dislikes);
            songSize++;
        }
    } catch (FileNotFoundException ex) {
        System.out.println("Error, could not find file.");
    }
}

//Question 4.4 - 6

//method header correct
//initialise string to heading
//loop through at least ONE array correctly
//append Music object correctly
//append Song object correctly
//return string with some attempt at headings AND new lines
public String toString() {
    String ret = "Musicians\n";
    for (int i = 0; i < musSize; i++) {
        ret = ret + musArr[i] + "\n";
    }
    ret = ret + "\nSongs\n";
    for (int i = 0; i < songSize; i++) {
        ret = ret + songArr[i] + "\n";
    }
    return ret;
}

//Question 4.5 - 6
//loop through Musician array
//getStart date
//calculate difference in years between start and NOW
//if inside loop to check number of years > 5
//add name and number of years to string
//return correctly formatted string

public String getLongTermMusicians() {
    String ret = "";
    for (int i = 0; i < musSize; i++) {
        LocalDate start = musArr[i].getStart();
        Period length = Period.between(start,
```

```
                                            LocalDate.now());
            if (length.getYears() > 5) {
                ret = ret + musArr[i].getName() + " " +
    length.getYears() + " years.\n";
            }
        }
        return ret;
    }
```

Alternate solution for Q4.5

```
    public String getLongTermMusiciansAlternateSolution() {
        String ret = "";
        for (int i = 0; i < musSize; i++) {
            LocalDate start = musArr[i].getContractStart();
            int yearStart = start.getYear();
            int monthStart = start.getMonthValue();
            int dayStart = start.getDayOfMonth();
            int yearNow = LocalDate.now().getYear();
            int monthNow = LocalDate.now().getMonthValue();
            int dayNow = LocalDate.now().getDayOfMonth();
            int diff = yearNow - yearStart;
            if (monthNow < monthStart || monthNow == monthStart &&
    dayNow < dayStart) {
                diff--;
            }
            if (diff > 5)  {
                ret = ret + musArr[i].getName() + " " + diff + "
    years.\n";
            }
        }
        return ret;
    }
```

Accept all working variants of LocalDate, such as Date. Also accept conversion to string and extracting year, month, day using text functions. Accept variants of Period (e.g. ChronoUnits)

QUESTION 6.1 & 6.3

```
//Question 6.1 – 6
//string and integer parameters sent to method
//loop through musician array
//if statement inside loop
//if statement compares array name to parameter
//call setGenre inside if statement
// correct parameter sent to setGenre method
public void adjustGenre(String inName, int inGenre) {
    for (int i = 0; i < musSize; i++) {
        if  (musArr[i].getName().equals(inName) ) {
            musArr[i].setGenre  (inGenre);
        }
    }
}

ALTERNATE SOLUTION
public void adjustGenre(String inName, int inGenre) {
    boolean found = false;
    int i = 0;
    while (!found && i < musSize) {
    if (musArr[i].getName().equals(inName) {
       musArr[i].setGenre(inGenre);
       found = true;
    }
    i++;
}

//Question 6.3 – 6
//method header correct – private
//create and initialise counter variable
//loop through song array
//if statement inside loop AND compares array name to
parameter correctly
//increment counter inside if statement
//return correct counter value
private int countSongs(String inName) {
    int count = 0;
    for (int i = 0; i < songSize; i++) {
        if (songArr[i].getArtistName().equals(inName))  {
            count++;
        }
    }
    return count;
       }
```


QUESTION 7.1

```
    //Question 7.1 - 17
    //method header correct
    //create empty string
    //loop through musician array
    //add name and " Easy Listeners\n" to string
    //check if artist has recorded any songs
    //set flag for easy listeners exist to false
    //loop through songs
    //check if current song has > 5000 likes
    //AND check if current song is well liked
    //AND check if current song artist matches name from outer
    //add song title inside if statement
    //set flag for easy listeners to true
    //check if NO easy listeners found
    //add "No suitable songs" correctly inside if
    //check if artist has NOT recorded any songs (else)
    //add "No songs recorded" correctly inside else (or another
    if)
    //return correct string
    public String listEasyListeners() {
      String ret = "";
      for (int i = 0; i < musSize; i++) {
         String name = musArr[i].getName();
         ret = ret + name + " Easy Listeners\n";
         if (countSongs(name) != 0)  {
           boolean foundEasy = false;
             for (int j = 0; j < songSize; j++) {
               if (songArr[j].getLikes() > 5000 &&
                   songArr[j].getWellLiked() &
                   songArr[j].getArtistName().equals(name) ) {
                     ret = ret + songArr[j].getTitle() + "\n";
                     foundEasy = true;
                   }
               }
               if (!foundEasy)  {
                   ret = ret + "No suitable songs\n";
               }
           } else {
               ret = ret + "No songs recorded\n";
           }
           ret = ret + "\n";
        }
      return ret;
    }
}
```

QUESTION 5, 6.2, 7.2

```
//Question 5
//Question 5.1 - 1
//class created with main method
public class MusicUI {

    public static void main(String[] args) {

        //Question 5.2 - 1
        //MemberManager created correctly (inside or outside main
        method)
        MusicManager mm = new MusicManager();
        //Question 5.3 - 1
        //print MusicManager object (only penalise ONCE for lack
        of println)
        System.out.println(mm);
        //Question 5.4 - 1
        //call getLongTermMusicians (do not penalise if not inside
        println)
        System.out.println(mm.getLongTermMusicians());

        //Question 6.2 - 2
        //call adjustGenre
        //pass correct parameters to method, MUST use constant
        mm.adjustGenre ("Samuel Ericson", Musician.TECHNO);

        //Question 7.2 - 1
        //call and print listEasyListeners (do not penalise if not
        inside println)
        System.out.println(mm.listEasyListeners());
     }
}
```

Total: 150 marks

DELPHI SOLUTION

QUESTION 2

```
//Question 2
//Question 2.1 - 3
//class header
//all fields private
//all correctly typed with correct names
unit Song;
interface
uses SysUtils;
type TSong = class
  private
    artistName : string;
    title : string;
    likes : integer;
    dislikes : integer;

  public
    constructor Create(inAN : string; inTitle : string; inL :
                Integer; inD : Integer);
    function getArtistName() : string;
    function getTitle() : string;
    function getLikes() : Integer;
    function getWellLiked() : Boolean;
    function toString() : string;
end;

implementation

    //Question 2.2 - 3
    //correct header
    //correct parameter names and types and order
    //fields set to parameters

    constructor TSong.Create (inAN: string; inTitle: string; inL:
                Integer; inD: Integer);
    begin
      artistName := inAN;
      title := inTitle;
      likes := inL;
      dislikes := inD;
    end;

    //Question 2.3 - 2
    //correct headers for all
    //correct return statements for all
    //if one or two accessors are missing and/or incorrect award
    only 1 mark
    function TSong.getArtistName: string;
    begin
      Result := artistName;
     end;
```

```
    function TSong.getTitle: string;
    begin
        Result := title;
    end;

function TSong.getLikes: Integer;
begin
    Result := likes;
end;

  //Question 2.4 - 3
  //correct header with correct return type
  //calculate likes - dislikes
  //return correct result
  function TSong.getWellLiked: Boolean;
  begin
    if likes > dislikes then
    begin
      Result := true;
    end
    else
    begin
      Result := false;
    end;
  end

  //Question 2.5 - 2
  //correct header
  //return all fields, correctly formatted
  function tSong.toString: string;
  begin
    Result := title + #9 + IntToStr(likes) + #9 +
              IntToStr(dislikes);
  end;
}
```

QUESTION 3

```
//Question 3
//Question 3.1 - 3
//class header
//all fields private
//all fields correctly typed with correct names
unit Musician;

interface
uses SysUtils, DateUtils;

type TMusician = class
  private
    name : string;
    stageName : string;
    genre : Integer;
    start : TDate;

  //Question 3.2 - 2
  //constants declared with public const and named and typed
    correctly
  //assigned correct value to correct name
  public
    const
      ROCK = 1;
      GQOM = 2;
      AMAPIANO = 3;
      TECHNO = 4;

    constructor Create(inName : string; inStage : string; inGenre :
     Integer; inStart : TDate);
    function getName() : string;
    function getStart() : TDate;
    function getGenre() : string;
    procedure setGenre(inGenre : Integer);
    function toString() : string;

    //Question 3.3 - 5
    //constructor named correctly
    //parameters correct and in correct order
    //assign correct values to genre and start
    //check for inStage empty
    //correctly assign inName or inStage to stageName
    constructor TMusician.Create (inName: string; inStage: string;
                inGenre: Integer; inStart: TDate);
    begin
      name := inName;
      genre := inGenre;
      start := inStart;
      if inStage = '' then
      begin
        stageName := inName;
      end
```

```
      else
      begin
        stageName := inStage;
      end;
    end;

    //Question 3.4 - 2
    //correct headers for all accessors
    //correct returns for all accessors
    //if one accessor missing or incorrect award only 1 mark
    function TMusician.getName: string;
    begin
      Result := name;
    end;

    function TMusician.getStart: TDate;
    begin
      Result := start;
    end;

    //Question 3.5 - 4
    //selection (case/if) on genre
    //use of switch or nested if
    //"Uncategorised" returned for ANY value outside 1-4
    //return correct string for all other cases
    function TMusician.getGenre: string;
  begin
    case genre of
      ROCK : Result := 'Rock';
      GQOM : Result := 'Gqom';
      AMAPIANO : Result := 'Amapiano';
      TECHNO : Result := 'Techno';
      else Result := 'Uncategorised';
    end;
  end;

    //Question 3.6 - 2
    //correct header with parameter
    //assign parameter to field
    procedure TMusician.setGenre(inGenre: Integer);
    begin
      genre := inGenre;
    end;

    //Question 3.7 - 2
    //return all fields in a correctly formatted string
                                                  //call to genGenre
    function TMusician.toString() : string;
    begin
      Result := stageName + #9 + getGenre() + #9 + DateToStr(start);
    end;

}
```

QUESTION 4

```
//Question 4
//Question 4.1 - 1
//class created correctly
unit MusicManager;

interface
uses SysUtils, DateUtils, Song, Musician;

type TMusicManager = class

    //Question 4.2 - 3
    //all properties private
    //both arrays declared and instantiated correctly with correct
    size
    //both sizes declared correctly
  private
    musArr : array[1..100] of TMusician;
    musSize : integer;
    songArr : array[1..100] of TSong;
    songSize : integer;

    function countSongs(inName : string) : Integer;

  public
    Constructor Create();
    function toString() : string;
    function getLongTermMusicians() : string;
    procedure adjustGenre(inName : string; inGenre : Integer);
    function listEasyListeners() : string;
end;

    //Question 4.3 - 15
    //Constructor header correct

    //open file for reading
    //loop through first 7 lines
    //read next line from file
    //increment musSize
    //extract field from line
    //delete field from line
    //extract year, month, day
    //encode date pattern
    //create Musician object


    //loop through all remaining lines
    //read next line from file
    //increment songSize
    //extract information from line
    //create Song object
```

```
constructor TMusicManager.Create;
  var
    inFile : textfile;
    line, sArtistName, sTitle, mName, mStage, date, y, m, d:
string;
    sLikes, sDislikes, mGenre :  Integer;
    mStart : TDate;
  i: Integer;
  begin
    if FileExists('MusicData.txt') <> true then
    begin
        WriteLn('File not found.');
    end
    else
    begin
        AssignFile(inFile, 'MusicData.txt');
        Reset(inFile);
        musSize := 0;
        songSize := 0;
        for i := 1 to 7 do
        begin
         ReadLN(inFile, line);
         Inc(musSize);
         mName := Copy(line, 1, Pos(';', line) -1);
         Delete(line, 1, Pos(';', line));
         mStage := Copy(line, 1, Pos(';', line) -1);
         Delete(line, 1, Pos(';', line));
         mGenre := StrToInt(Copy(line, 1, Pos(';', line) -1));
         Delete(line, 1, Pos(';', line));
         date := line;
         y := Copy(date, 1, Pos(' ', date) -1);
         Delete(date, 1, Pos(' ', date));
         m := Copy(date, 1, Pos(' ', date) -1);
         Delete(date, 1, Pos(' ', date));
         d := date;
         mStart := EncodeDate(StrToInt(y), StrToInt(m),
StrToInt(d));
         musArr[musSize] := TMusician.Create(mName, mStage,
mGenre, mStart);
        end;
        while NOT EOF(inFile) do
        begin
          ReadLN(infile, line);
          Inc(songsize);
          sArtistName := Copy(line, 1, Pos(';', line) -1);
          Delete(line, 1, Pos(';', line));
          sTitle := Copy(line, 1, Pos(';', line) -1);
          Delete(line, 1, Pos(';', line));
          sLikes := StrToInt(Copy(line, 1, Pos(';', line) -
1));
          Delete(line, 1, Pos(';', line));
          sDislikes := StrToInt(line);
```

```
              songArr[songSize] := Tsong.Create(sArtistname,
    sTitle, sLikes, sDislikes);
            end;
        end;
end;
   //Question 4.4 - 6

   //method header correct
   //initialise string
   //loop through at least ONE array correctly
   //append Music object correctly
   //append Song object correctly
   //return string with some attempt at headings AND new lines
  function TMusicManager.toString: string;
  var
    i : Integer;
    output : string;
  begin
    output := 'Musicians' + #13#10;
    for i := 1 to musSize do
    begi
      output := output + musArr[i].toString() + #13#10;
    end;
    output := output + #13#10 + 'Songs' + #13#10;
    for i := 1 to songSize do
    begin
      output := output + songArr[i].toString() + #13#10;
    end;
    Result := output;
  end;     }

   //Question 4.5 - 6
   //loop through Musician array
   //getContractStart date
   //calculate difference between contractStart and NOW
   //if inside loop to check number of years > 5
   //add name and number of years to string
   //return correctly formatted string

  function TMusicManager.getLongTermMusicians: string;
  var
    i, numYears : Integer;
    output : string;
  begin
    output := '';
    for i := 1 to musSize do
    begin
      numYears := 1 + DateUtils.YearsBetween (Now,
   musArr[i].getStart());
      if DateUtils.IncYear(musArr[i].getStart(), numYears) > Now
   then
      begin
        dec(numYears);
      end;
```

```
      if numYears > 5 then
      begin
        output := output + musArr[i].getName() + ' ' +
 IntToStr(numYears) + ' years.' + #13#10;
      end;
    end;
    Result := output;
end;
```

QUESTION 6.1 & 6.3

```
//Question 6.1 – 6
//string and integer parameters sent to method
//loop through musician array
//if statement inside loop
//if statement compares array name to parameter
//call setGenre inside if statement
// correct parameter sent to setGenre method
procedure TMusicManager.adjustGenre(inName : string; inGenre :
Integer);
var
  i : integer;
begin
  for i := 1 to musSize do
  begin
    if musArr[i].getName() = inName then
    begin
      musArr[i].setGenre (inGenre);
    end;
  end;
end;

//Question 6.3 – 6
//method header correct – private (check declaration above)
//create and initialise counter variable
//loop through song array
//if statement inside loop AND compares array name to
parameter correctly
//increment counter inside if statement
//return correct counter value
function TMusicManager.countSongs(inName: string): Integer;
  var
    i, count: Integer;
  begin
      count := 0;
      for i := 1 to songSize do
      begin
        if songArr[i].getArtistName() = inName then
          Inc(count);
      end;
      Result := count;
end;
```


QUESTION 7.1

```
//Question 7.1 - 17
//method header correct
//create empty string
//loop through musician array
//add name and " Easy Listeners\n" to string
//check if artist has recorded any songs
//set flag for easy listeners exist to false
//loop through songs
//check if current song has > 5000 likes
//AND check if current song is well liked
//AND check if current song artist matches name from outer
//add song title inside if statement
//set flag for easy listeners to true
//check if NO easy listeners found
//add "No suitable songs" correctly inside if
//check if artist has NOT recorded any songs (else)
//add "No songs recorded" correctly inside else (or another
if)
//return correct string

function TMusicManager.listEasyListeners: string;
  var
    output, name : string;
    foundEasy : Boolean;
    i, j : Integer;
  begin
    output := '';
    for i := 1 to musSize do
    begin
      name := musArr[i].getName();
      output := output + name + ' EasyListeners' + #13#10;
      if countSongs(name) <> 0 then
      begin
        foundEasy := false;
        for j := 1 to songSize do
        begin
          if (songArr[j].getLikes() > 5000) and
(songArr[j].getWellLiked()) and (songArr[j].getArtistName() =
name)  then
          begin
              output := output + songArr[j].getTitle() +
#13#10;
              foundEasy := true;
          end;
        end;
        if not foundEasy then
        begin
          output := output + 'No suitable songs' + #13#10;
        end;
      end
      else
      begin
```

```
            output := output + 'No songs recorded' + #13#10;
        end;
        output := output + #13#10;
      end;
      Result := output;
    end;
```


QUESTION 5, 6.2, 7.2

```
//Question 5
//Question 5.1 - 1
//class created
program MusicUI;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils,
  Song in 'Song.pas',
  Musician in 'Musician.pas',
  MusicManager in 'MusicManager.pas';

var
  mm : TMusicManager;

begin
  try
    { TODO -oUser -cConsole Main : Insert code here }
    //Question 5.2 - 1
    //MemberManager created correctly
    mm := TMusicManager.Create();

    //Question 5.3 - 1
    //print MusicManager object
    WriteLn(mm.toString());

   //Question 5.4 - 1
    //call getLongTermMusicians (do not penalise if not inside
        WriteLn
    WriteLn(mm.getLongTermMusicians());

    //Question 6.2 - 2
    //call adjustGenre
    //pass correct parameters to method, MUST use constant
    mm.adjustGenre ('Samuel Ericson', TMusician.TECHNO) ;

    //Question 7.2 - 1
    //call and print listEasyListeners
    WriteLn(mm.listEasyListeners());
    ReadLn;
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

Total: 150 marks