GRADE 12 MEMO: INFORMATION TECHNOLOGY (P1)

QUESTION ONE: JAVA DATABASE CONNECTIVITY

Mark allocation

Question ONE – Marking Grid			
Name of learner:_____			
Question	**Aspect**	**Max Mark**	**Learner's Mark**
1.1	SELECT(1) FROM EmployeesTB (1) ORDER BY(1) EmpSurname (1)	**4**	
1.2	SELECT (1) EmpNumber,FORMAT(AmountRequested,'0.00')(1) AS NewAmount,purpose (1) FROM FundsTB (1) WHERE Purpose (1) LIKE (1) '%Workshop' (1)	**7**	
1.3	SELECT(1) DISTINCT (1) (Purpose) (1) FROM (1) FundsTB (1)	**5**	
1.4	SELECT (1) EmpNumber,FORMAT(AmountRequested) (1) AS Amount,Purpose,FORMAT(DateRequested) (1) AS RequestedDate (1) FROM FundsTB (1) WHERE MONTH(DateRequested)< 4 (1)	**6**	
1.5	SELECT (1) EmployeesTB.EmpName,EmployeesTB.EmpSurname,EmployeesTBEmpType,FORMAT(FundsTB.AmountRequested) (1) FundsTB.Purpose (1) FROM EmployeesTB,FundsTB (1) WHERE (1) EmployeesTB.EmpNumber = FundsTB.EmpNumber (1) AND AmountRequested >= 200 (1)	**7**	
1.6	SELECT (1) FORMAT(SUM(AmountRequested),'0.00' (2)) (1) AS totAmount (1) FROM FundsTB (1)	**6**	
1.7	Input (2) DELETE (1) FROM EmployeesTB (1) WHERE EmpNumber = inputValue (1)	**5**	
	TOTAL	**40**	

```
import java.awt.*;
import java.sql.*;
import java.io.*;

public class Employee
{

   Connection conn;

   public Employee()
   {
      if(loadDriver())
      {
         connectDB();
      }
   }
   public boolean loadDriver()
   {
      try
      {
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        return true;
      }
      catch(ClassNotFoundException ex)
      {
         System.out.println("failed to load Driver");
      }
   return false;
  }
   public void connectDB()
   {
try
{
String filename = "E:/Question1/SchoolDB.mdb";
String database = "jdbc:odbc:Driver={Microsoft Access
Driver (*.mdb)};DBQ=";
database += filename.trim() +
";DriverID=22;READONLY=true}";
conn = DriverManager.getConnection(database, "", "");
System.out.println ("DB is successfully connected");
}
catch(Exception ex)
{
 System.out.println ("Failed to connect DB. Make sure the"+
 " file exists in a specified location.");
}
```

```
public void displayAllEmployees()
{
  try
  {
   String query = "SELECT✓ * FROM EmployeesTB✓ ORDER BY✓
                   empSurname";✓
   Statement stmt = conn.createStatement();
   ResultSet rs = stmt.executeQuery(query);
   System.out.printf("%-10s%-8s%-15s%-10s%-
10s%10s","Employee#","Title","Type","Name","Surname","Phone
No");
   System.out.println();

System.out.println("==================================
========================");
         while(rs.next())
         {
            String empNumber = rs.getString("empNumber");
            String empTitle = rs.getString("empTitle");
            String empType = rs.getString("empType");
            String empName = rs.getString("empName");
            String empSurname = rs.getString("empSurname");
            String empPhoneNo = rs.getString("empPhoneNo");

System.out.printf("%-10s%-8s%-15s%-10s%-10s%-
10s",empNumber,empTitle,empType,empName,empSurname,empPhone
No);
            System.out.println();
         }
         stmt.close();
      }
      catch(SQLException sqlex)
      {
         sqlex.printStackTrace();
      }

   }
   //-------------------------------------------------1.1
```

```
public void displayWorkshops()
    {
     try
     {
     String query = "SELECT ✓
      EmpNumber,FORMAT(AmountRequested,'0.00') ✓ AS
      NewAmount,purpose✓ FROM FundsTB✓ WHERE Purpose✓ LIKE✓
      '%Workshop'";✓
          Statement stmt = conn.createStatement();
          ResultSet rs = stmt.executeQuery(query);
          System.out.printf("%-10s%-10s%-
      15s","Employee#","AmountReq","Purpose");
      System.out.println();
      System.out.println("==================================
      ==========");
          while(rs.next())
          {
             String empNumber = rs.getString("EmpNumber");
             String amountReq = rs.getString("NewAmount");
             String purpose = rs.getString("purpose");
             System.out.printf("%-10s%-10s%-
      15s",empNumber,"R"+amountReq,purpose);
             System.out.println();
          }
          stmt.close();
       }
       catch(SQLException sqlex)
       {
          sqlex.printStackTrace();
       }
  }
   //-------------------------------------------------1.2

   public void displayAllRequests()
   {
      try
      {
   String query = "SELECT✓ DISTINCT✓ (Purpose) ✓ FROM✓
                          FundsTB";✓
          Statement stmt = conn.createStatement();
          ResultSet rs = stmt.executeQuery(query);
   System.out.println("Type of meetings
                    Attended\n=========================");
          while(rs.next())
          {
             String purpose = rs.getString("Purpose");
             System.out.println(purpose);
```

```
            }
            stmt.close();
        }
        catch(SQLException sqlex)
        {
            sqlex.printStackTrace();
        }
    }
    //-----------------------------------------------------1.3

    public void requestBeforeApril()
    {
      try
      {
      String query = "SELECT✓
      EmpNumber,FORMAT(AmountRequested) ✓ AS
      Amount,Purpose,FORMAT(DateRequested) ✓AS RequestedDate
      FROM FundsTB✓ WHERE✓ MONTH(DateRequested)< 4";✓
          Statement stmt = conn.createStatement();
          ResultSet rs = stmt.executeQuery(query);
          System.out.printf("%-10s%-15s%-15s%-
      15s","Employee#","AmountReq","Purpose","Date
      Requested");
      System.out.println();

      System.out.println("==================================
      ====================");
          while(rs.next())
          {
             String empNumber = rs.getString("EmpNumber");
             String amountReq = rs.getString("Amount");
             String purpose = rs.getString("Purpose");
             String dateReq = rs.getString("RequestedDate");

             System.out.printf("%-10s%-10s%-20s%-
             20s",empNumber,"R"+amountReq,purpose,dateReq);
             System.out.println();
          }
          stmt.close();
      }
      catch(SQLException sqlex)
      {
          sqlex.printStackTrace();
      }
    }

//-----------------------------------------------------1.4
```

```
public void amountRequested()
{
 try
 {
 String query = "SELECT✓
                          ✓
  EmployeesTB.EmpName,EmployeesTB.EmpSurname,EmployeesTB
  .EmpType,FORMAT(FundsTB.AmountRequested) ✓
  ,FundsTB.Purpo se "+✓"FROM EmployeesTB,FundsTB "+✓
  "WHERE✓ EmployeesTB.EmpNumber = FundsTB.EmpNumber AND
  AmountRequested >= 200";✓
      Statement stmt = conn.createStatement();
      ResultSet rs = stmt.executeQuery(query);
      System.out.printf("%-10s%-10s%-15s%-10s%-
      10s","Name","Surname","Type","Amount","Purpose");
      System.out.println();

  System.out.println("==================================
  ============================");
      while(rs.next())
      {
         String empName = rs.getString(1);
         String empSurname = rs.getString(2);
         String empType = rs.getString(3);
         String amount = rs.getString(4);
         String purpose = rs.getString(5);

         System.out.printf("%-10s%-10s%-15s%-10s%-
  10s",empName,empSurname,empType,"R"+amount,purpose);
         System.out.println();
      }
      stmt.close();
   }
   catch(SQLException sqlex)
   {
      sqlex.printStackTrace();
   }
}

//-------------------------------------------------1.5
```

```
public void totalAmountRequested()
{
  try
  {
  String query = "SELECT✓  FORMAT(SUM✓✓
  (AmountRequested),'0.00') ✓ AS totAmount✓ FROM
  FundsTB";✓
  Statement stmt = conn.createStatement();
  ResultSet rs = stmt.executeQuery(query);
      while(rs.next())
      {
         String totAmount = rs.getString("totAmount");
         System.out.println("The total amount requested
         is: R"+ totAmount);
      }
      stmt.close();
   }
   catch(SQLException sqlex)
   {
      sqlex.printStackTrace();yyyy
   }
}

//-----------------------------------------------------1.6

public void deleteEmployeeRecord() throws IOException
{
   InputStreamReader in = new
   InputStreamReader(System.in);
   BufferedReader buff = new BufferedReader(in);
   System.out.print("Enter the Employee#: ");
   String input = buff.readLine();✓
   try
   {
   String query = "DELETE✓ FROM EmployeesTB✓ WHERE✓
                   EmpNumber = '"+input.trim()+"'";✓
      Statement stmt = conn.createStatement();
      int result = stmt.executeUpdate(query);

      if(result == 1)
      {
         System.out.println("Record deleted!!!");
      }
```

```
            else
            {
               System.out.println("Invalid Employee number");
            }
            stmt.close();
         }
         catch(SQLException sqlex)
         {
            sqlex.printStackTrace();
         }
      }

      //-----------------------------------------------------1.7
}
```

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING (OOP) SOLUTION

Mark allocation

Question TWO – Marking Grid			
Name of learner:_____			
Question	**Aspect**	**Max Mark**	**Learner's Mark**
2.1	Object class		
2.1.1	Declare private fields (subtract marks for errors,max mark 2 (1 each))	**2**	
2.1.2	Contructor: parameters declared (1), correct data type (1)	**2**	
2.1.3	Accessor and mutator methods declared (max 4) (1 each)	**4**	
2.1.4	**toString** (1)method: returns type string (1), in code returns string (1), format funds to two decimal places(1)	**4**	
	Subtotal	**12**	
2.2			
2.2.1	Create an array of objects (2)	**2**	
2.2.2	Test if the file exists(2). If it exists, initialize a loop to read data(1). Display a suitable message if the file does not exist(1) terminate the program (1). Read a line of text from a text file(2). Check if the line from the text file is not an empty string (1) If the line is not empty, separate the text into the **funds** and the **year** in which the funds were received. (2) Use the information to create a new **funds** object and place the object into the array(1). Close file (1) (12/2 = 6)	**6**	
2.2.3	**Print ALL funds:** Suitable heading (1),loop (1),if(funds[var] != null)(1), toString call (1)	**4**	

	Print ALL funds greater than R600000.00:
	
	Suitable heading (1),
	If(funds[var].getFunds() > 600000) (1)
	
	Print Total funds raised between the year 200 and 2005:
	
	Declare variable (1),
	`if(funds[index].getYear() >= 2000 (1)&& funds[index].getYear() <= 2005 (1)),`
	`var +=(1) funds[index].getFunds()1)`
	
	Add new funds:
	
	Input(2)
	Declare FileWriter(1), declare Printwriter(1),
	Loop(1)
	Write to a file(1), add new object to an array(1),
	Print message (1)
	(8/2 = 4)
	Sort According to funds in descending order:
	
	`If-condition(1)`
	`read initial values from array(2)`
	use set (1)and get(1) methods from **Fund** when swapping
	swap values(2)
	print sorted array (1)
	
	Print the Highest fund:
	
	Read the first value of funds from the array(1)
	Read the first value of year from the array(1)
	Loop(1)
	Condition to compare values(1)
	Code to assign vales with highest fund to variables(1)
	Print message (1)
		2
		
		6
		
		4
		
		8
		
		6	

	Subtotal	**38**	
	TOTAL	**50**	

```
import java.text.*;

public class Fund
{
   DecimalFormat df = new DecimalFormat("0.00");
   private double funds; ✓
   private int year; ✓
                                  ✓
   public Fund(double sFunds, int sYear) ✓
   {
      funds = sFunds;
      year = sYear;
   }
   public void setFunds(double sFunds) ✓
   {
      funds = sFunds;
   }
   public double getFunds()✓
   {
      return funds;
   }
   public void setYear(int sYear) ✓
   {
      year = sYear;
   }
   public int getYear()✓
   {
      return year;
   }
   public String✓ toString()✓
   {
                        ✓
      return "R"+df.format(funds) +"\t"+year; ✓
   }
}


//-----------------------------------------------------2.1
```

```
import java.io.*;
import java.awt.*;
import java.text.*;

public class TestFund
{

 public static void main(String[] args) throws IOException{
 DecimalFormat df = new DecimalFormat("0.00");
 //ConstructorToCall constructor = new ConstructorToCall();
 Fund[] funds = new Fund[20]; ✓✓
 char option = ' ';
 File f = new File("E:/Question2/Fundss.txt");
 if(f.exists())✓
 {
   BufferedReader br = new BufferedReader(new
     FileReader(f)); ✓
         String line = br.readLine();✓
         int i = 0;

        do✓  //any loop
         {
            if(!line.equals(""))✓
            {
               String[] s = line.split("#");✓
               int year = Integer.parseInt(s[0]); ✓
               double money = Double.parseDouble(s[1]); ✓
               funds[i] = new Fund(money,year); ✓
               line = br.readLine();
               i++;
            }

         }
         while(line != null);
         br.close();✓
      }
      else
      {
         System.out.println("The file does not exist");✓
         System.exit(0); ✓
      }
      InputStreamReader in = new
      InputStreamReader(System.in);
      BufferedReader buffer = new BufferedReader(in);
```

```
do
{
   option = ' ';
 System.out.println("=============MAIN
    MENU============="+
    "\n A. Print ALL funds"+
    "\n B. Print ALL funds greater than R600000.00"+
    "\n C. Print Total funds raised between the year
    2000 and 2005"+
    "\n D. Add new funds"+
    "\n E. Sort According to Funds in descending
    order"+
    "\n F. Print the Highest Fund"+
    "\n Q. Quit\n");
   System.out.print("Enter your option: ");
   option =
  buffer.readLine().toUpperCase().charAt(0);
   switch(option)
   {
      case 'A'://Code here

            System.out.println("Funds\t\tYear");✓
            System.out.println("================");
            for(int j = 0;j<funds.length;j++)✓
            {
               if(funds[j] != null) ✓
               {
    System.out.println(funds[j].toString());✓
               }
            }

         break;
      case 'B'://code here

 System.out.println("Funds greater than R600000.00");
 System.out.println("============================");✓
            for(int k = 0;k<funds.length;k++)
            {
               if(funds[k] != null)
               {
                  if(funds[k].getFunds() > 600000) ✓
                  {

 System.out.println(funds[k].toString());
                  }
               }
```

```
        }

      break;
  case 'C'://code here
        double scFunds = 0.0; ✓
        for(int k = 0;k<funds.length;k++)
        {
           if(funds[k] != null)
           {
if(funds[k].getYear() >= 2000✓ &&
   funds[k].getYear() <= 2005) ✓
           {
        scFunds +=✓ funds[k].getFunds();✓
           }
        }

        }
System.out.println("The total funds is: R"+
df.format(scFunds)); ✓

          break;
        case 'D'://code here
        FileWriter fr = new FileWriter(f,true); ✓
                                              ✓
        PrintWriter pr = new PrintWriter(fr,true);
        System.out.print("Enter Funds: ");
        double aFunds =
        Double.parseDouble(buffer.readLine());✓
        System.out.print("Enter Year: ");
        int aYear =
        Integer.parseInt(buffer.readLine());✓

        for(int s = 0;s<funds.length;s++)✓
        {
          if(funds[s] == null)
          {
            pr.println(aYear+"#"+aFunds); ✓
            funds[s] = new Fund(aFunds,aYear); ✓
            System.out.println("Record is
            added!!!"); ✓
           break;
           }
        }
          break;
```

```
      case 'E'://code here
        for(int p = 0; p < funds.length - 1;p++)
        {
          for(int j = p+1;j< funds.length;j++)
          {
            if(funds[j] != null)
             {
              if(funds[p].getFunds() < ✓
                   funds[j].getFunds())
              {
               double tempFunds =
               funds[p].getFunds();✓
               int tempYear =
               funds[p].getYear();✓
                              ✓
   funds[p].setFunds(funds[j].getFunds());
   funds[p].setYear(funds[j].getYear());✓
   funds[j].setFunds(tempFunds); ✓
   funds[j].setYear(tempYear); ✓
         }
        }
       }
      }

  System.out.println("Funds\t\tYear");
  System.out.println("================");
  for(int k = 0;k< funds.length;k++)
  {
   if(funds[k] != null)
   {                                  ✓
  System.out.println(funds[k].toString());
   }
  }
   break;
case 'F'://code here
      double highValue =
 funds[0].getFunds();✓
      int fYear = funds[0].getYear();✓
      for(int s = 1;s<funds.length;s++)
      {
       if(funds[s] != null)
       {                            ✓
        if(funds[s].getFunds() >highValue)
```

```
                    {                          ✓
                      highValue = funds[s].getFunds();
                      fYear = funds[s].getYear();✓
                        }
                     }
                  }                            ✓
         System.out.println("The highest fund is R"+
   df.format(highValue)+" obtained in "+fYear);
              break;

        }
//-------------------------------------------------------2.2

          }
          while(option !='Q');
       }
   }
```

QUESTION 3: JAVA PROGRAMMING

Mark allocation

Question THREE – Marking Grid			
Name of learner:_____			
Question	**Aspect**	**Max Mark**	**Learner's Mark**
3.1			
3.1.1	Heading(1) Lop(1) Code to read data from first column (1) Code to read data from second column (1) Print output (1)	**5**	
3.1.2	Declare variable to hold the total (1) Loop(1) Convert data to double (1) Add values (1) Assign calculated value to a variable (1) Print message (1)	**7**	
3.1.3	Create File object (1) Create FileWriter object (1) Create PrintWriter object (1) Loop(1) Inner loop (1) Read data from 2D array (1) Print data to a text file (1) Print line in a text file.(1) Print message (1) Close file(1)	**10**	
3.1.4	Create a constructor (2) Call relevant method to run the program (6: 2 each)	**8**	
	TOTAL	**30**	

import java.io.*;

```
import java.io.*;

public class Requisition
{

String[][] request = {
                     {"MJ Smith","Technical","137000.00"},
                     {"PR Kobus","Sciences","121000.00"},
                     {"DD Mdluli","Languages","111000.00"},
                     {"PP Mathe","Commerce","133000.00"}};

   public Requisition()
   {

   }
   public void displayDepartments()
   {
   System.out.println("HOD
          NAME\tDEPARTMENT\n======================");✓
     for(int i = 0; i < request.length;i++)✓
     {
                                  ✓              ✓
     System.out.println(request[i][0]+"\t"+request[i][1]);✓
     }
   }
//-------------------------------------------------3.1.1
   public void totalRequisition()
   {
     double total = 0; ✓
     for(int i = 0; i<request.length;i++)✓
     {
                           ✓                ✓
       total +=✓Double.parseDouble(request[i][2]); ✓
     }
     System.out.println("The total amount requested is
     R"+total); ✓
   }
//-------------------------------------------------3.1.2
   public void saveRecords() throws IOException
   {
   File f = new File("E:/Trial QP 2009/Question
     3/Requisitions.txt");✓
      FileWriter fr = new FileWriter(f); ✓
      PrintWriter pr = new PrintWriter(fr); ✓
      for(int i = 0;i< request.length;i++)✓
      {
```

```
            for(int j = 0;j<request[i].length;j++)✓
            {
                                    ✓
               pr.print(request[i][j]+"\t");✓
            }
            pr.println();✓
         }
         System.out.println("Record is saved!!!"); ✓
         pr.close();✓

      }
//-------------------------------------------------------3.1.3
}
//=====================================================

import java.io.*;

public class TestRequisition
{


  public static void main(String[] args) throws IOException
   {

    BufferedReader br = new BufferedReader(new
     InputStreamReader(System.in));

      //Constructor con = new Constructor();
      Requisition r = new Requisition();✓✓
      char option = ' ';

      do
      {
         System.out.print("MAIN MENU"+
                          "\n=================="+
                          "\n A: Display ALL Dep[artments"+
                          "\n B: Calculate Total
                           requisitions"+
                          "\n C: Save records"+
                          "\n Q: Quit!"+
                          "\n\n Enter your option: ");

        option = br.readLine().toUpperCase().charAt(0);

         switch(option)
         {
            case 'A':r.displayDepartments();✓✓
```

```
                // method call here

                break;

            case 'B':r.totalRequisition();✓✓
                // method call here


                break;

            case 'C':r.saveRecords();✓✓

                // method call here

                break;

        }

    }
    while(option != 'Q');
  }
}
```

GRAND TOTAL 120