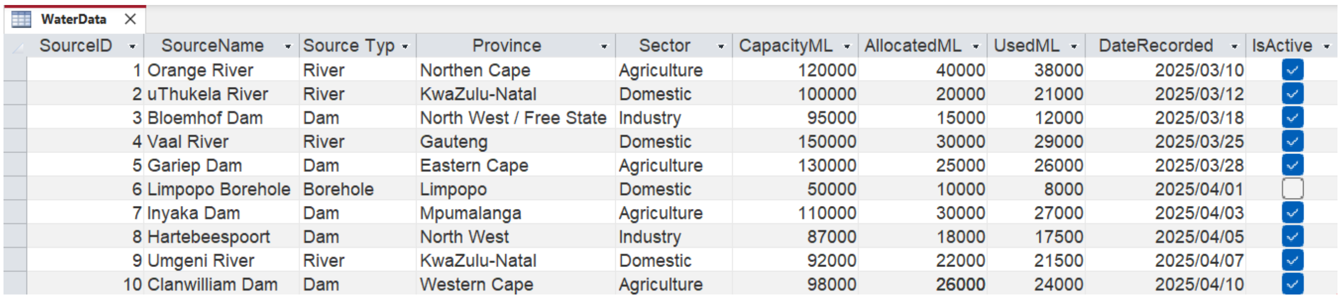
PAT (Practical Assessment Task) – Grade 11: Phase 3

Name Semira Nee-Whang

Database table

(screenshot)

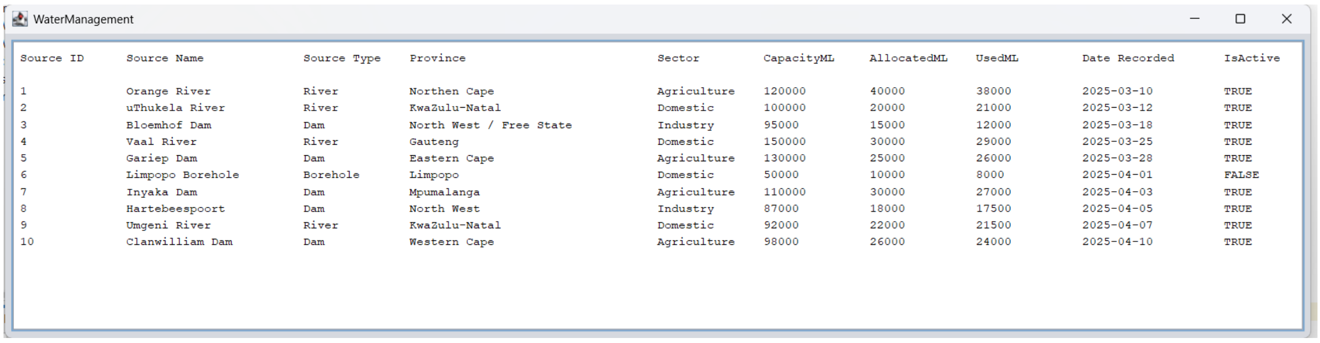


WaterData

SourceID	SourceName	Source Typ	Province	Sector	CapacityML	AllocatedML	UsedML	DateRecorded	IsActive
1	Orange River	River	Northen Cape	Agriculture	120000	40000	38000	2025/03/10	✓
2	uThukela River	River	KwaZulu-Natal	Domestic	100000	20000	21000	2025/03/12	✓
3	Bloemhof Dam	Dam	North West / Free State	Industry	95000	15000	12000	2025/03/18	✓
4	Vaal River	River	Gauteng	Domestic	150000	30000	29000	2025/03/25	✓
5	Gariep Dam	Dam	Eastern Cape	Agriculture	130000	25000	26000	2025/03/28	✓
6	Limpopo Borehole	Borehole	Limpopo	Domestic	50000	10000	8000	2025/04/01	☐
7	Inyaka Dam	Dam	Mpumalanga	Agriculture	110000	30000	27000	2025/04/03	✓
8	Hartebeespoort	Dam	North West	Industry	87000	18000	17500	2025/04/05	✓
9	Umgeni River	River	KwaZulu-Natal	Domestic	92000	22000	21500	2025/04/07	✓
10	Clanwilliam Dam	Dam	Western Cape	Agriculture	98000	26000	24000	2025/04/10	✓

GUI populated

(screenshot)



WaterManagement

```
Source ID   Source Name        Source Type   Province                Sector        CapacityML   AllocatedML   UsedML   Date Recorded   IsActive

1           Orange River       River         Northen Cape            Agriculture   120000       40000         38000    2025-03-10      TRUE
2           uThukela River     River         KwaZulu-Natal           Domestic      100000       20000         21000    2025-03-12      TRUE
3           Bloemhof Dam       Dam           North West / Free State Industry      95000        15000         12000    2025-03-18      TRUE
4           Vaal River         River         Gauteng                 Domestic      150000       30000         29000    2025-03-25      TRUE
5           Gariep Dam         Dam           Eastern Cape            Agriculture   130000       25000         26000    2025-03-28      TRUE
6           Limpopo Borehole   Borehole      Limpopo                 Domestic      50000        10000         8000     2025-04-01      FALSE
7           Inyaka Dam         Dam           Mpumalanga              Agriculture   110000       30000         27000    2025-04-03      TRUE
8           Hartebeespoort     Dam           North West              Industry      87000        18000         17500    2025-04-05      TRUE
9           Umgeni River       River         KwaZulu-Natal           Domestic      92000        22000         21500    2025-04-07      TRUE
10          Clanwilliam Dam    Dam           Western Cape            Agriculture   98000        26000         24000    2025-04-10      TRUE
```

Code

DB Class

```java
package pat;
import java.sql.*;
import javax.swing.*;

public class DB {

    //Declare properties
    private final String driver =
"net.ucanaccess.jdbc.UcanaccessDriver";
    //DB must be in project folder
    private final String url =

"jdbc:ucanaccess://D:/Documents/NetBeansProjects/PAT/WaterDB.accdb";
    private Connection connection;
    private Statement statement;
    private ResultSet resultSet;

     //Constructor method
    public DB() {
        //Load driver
        try {
            Class.forName(driver);
            System.out.println("Driver found");
        } catch (ClassNotFoundException e) //Trap the error if the
driver is not found
        {
            JOptionPane.showMessageDialog(null, "Error: Database
Driver not found");
        }

        //Create connection
        try {
            connection = DriverManager.getConnection(url);
            JOptionPane.showMessageDialog(null, "Connected
successfully!");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Unable to connect: "
                    + e.getMessage());
            e.printStackTrace(); // print detailed error in console
        }
    }

    public ResultSet queryDB(String inStatement) throws SQLException
    //This is the genetic query to execute a SELECT statement
    {
        //Query the database
        statement = connection.createStatement();

        System.out.println(inStatement);

        resultSet = statement.executeQuery(inStatement);
        //Return data as a resultset
        return resultSet;
    }

}
```

FRMWaterManagement

```
package pat;

public class FrmWatermanagemnt extends javax.swing.JFrame {

    private static final java.util.logging.Logger logger =

java.util.logging.Logger.getLogger(FrmWatermanagemnt.class.getName());
    WaterManagementManager waterManagement = new
WaterManagementManager();

    /**
     * Creates new form FrmWatermanagemnt
     */
    public FrmWatermanagemnt() {
        initComponents();
        populateWaterData();
    }

    private void populateWaterData() {
        txaWater.setText(" ");

        txaWater.setText(waterManagement.getWaterData());
    }
```

WaterManagement Class

```
package pat;
import java.util.Date;

public class WaterManagement {

    // Declaration of fields
    private int sourceID;
    private String sourceName;
    private String sourceType;
    private String province;
    private String sector;
    private int capacityML;
    private int allocatedML;
    private int usedML;
    private Date dateRecorded;
    private boolean isActive;

    // Parameterized constructor
    public WaterManagement(int id, String name, String type, String
prov,
                           String sect, int capacity, int allocated,
int used,
                           Date date, boolean active) {
        // Set the fields to the parameter values
        sourceID = id;
        sourceName = name;
        sourceType = type;
        province = prov;
        sector = sect;
        capacityML = capacity;
        allocatedML = allocated;
        usedML = used;
        dateRecorded = date;
        isActive = active;
    }

    // Accessor methods (Getters)
    public int getSourceID() {
        return sourceID;
    }

    public String getSourceName() {
        return sourceName;
    }

    public String getSourceType() {
        return sourceType;
    }

    public String getProvince() {
        return province;
```

```java
    }

    public String getSector() {
        return sector;
    }

    public int getCapacityML() {
        return capacityML;
    }

    public int getAllocatedML() {
        return allocatedML;
    }

    public int getUsedML() {
        return usedML;
    }

    public Date getDateRecorded() {
        return dateRecorded;
    }

    public boolean getIsActive() {
        return isActive;
    }

    // Mutator methods (Setters)
    public void setSourceID(int id) {
        sourceID = id;
    }

    public void setSourceName(String name) {
        sourceName = name;
    }

    public void setSourceType(String type) {
        sourceType = type;
    }

    public void setProvince(String prov) {
        province = prov;
    }

    public void setSector(String sect) {
        sector = sect;
    }

    public void setCapacityML(int capacity) {
        capacityML = capacity;
    }

    public void setAllocatedML(int allocated) {
        allocatedML = allocated;
    }

    public void setUsedML(int used) {
        usedML = used;
    }

    public void setDateRecorded(Date date) {
        dateRecorded = date;
    }

    public void setIsActive(boolean active) {
        isActive = active;
    }

    // Return a text value of whether the water source is active
(Yes/No)
    private String getIsActiveText() {
        return isActive ? "Yes" : "No";
    }

    // Calculate the remaining water in the source
    public int calcRemainingML() {
        return capacityML - usedML;
    }

    // toString method:
    // returns a neat string representation of the WaterManagement
object
    public String toString() {
        return "Source ID: " + sourceID + "\n"
               + "Source Name: " + sourceName + "\n"
               + "Source Type: " + sourceType + "\n"
               + "Province: " + province + "\n"
               + "Sector: " + sector + "\n"
               + "Capacity: " + capacityML + " ML\n"
               + "Allocated: " + allocatedML + " ML\n"
               + "Used: " + usedML + " ML\n"
               + "Remaining: " + calcRemainingML() + " ML\n"
               + "Date Recorded: " + dateRecorded + "\n"
               + "Active: " + getIsActiveText();
    }
}
```

WaterManagementManager Class

```java
package pat;

import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.*;

public class WaterManagementManager {
    // Create a DB object to handle the connection to the database
    private DB watermanagmentDB = new DB();
    // ResultSet stores the data returned by the database query
    private ResultSet rs;
    // Constructor
    public WaterManagementManager() {

    }
     // Start by creating a header row for the output table
    public String getWaterData() {
        String out = "Source ID" + addSpaces("Source ID", 15)
                + "Source Name" + addSpaces("Source Name", 25)
                + "Source Type" + addSpaces("Source Type", 15)
                + "Province" + addSpaces("Province", 35)
                + "Sector" + addSpaces("Sector", 15)
                + "CapacityML" + addSpaces("CapacityML", 15)
                + "AllocatedML" + addSpaces("AllocatedML", 15)
                + "UsedML" + addSpaces("UsedML", 15)
                + "Date Recorded" + addSpaces("Date Recorded", 20)
                + "IsActive" + addSpaces("IsActive", 5) + "\n\n";

          // SQL query to fetch all data from the WaterData table
        String query = "SELECT * FROM WaterData";

        try {
             // Run the query and store the results in the ResultSet
            rs = watermanagmentDB.queryDB(query);

            // Loop through each record (row) in the ResultSet
            while (rs.next()) {
                // Extract values from each column and format them
with spaces
                out += rs.getInt("SourceID") + addSpaces
        ("" + rs.getInt("SourceID"), 15);
                out += rs.getString("SourceName") + addSpaces
        (rs.getString("SourceName"), 25);
                out += rs.getString("Source Type") + addSpaces
        (rs.getString("Source Type"), 15);
                out += rs.getString("Province") + addSpaces
        (rs.getString("Province"), 35);
                out += rs.getString("Sector") + addSpaces
        (rs.getString("Sector"), 15);
                out += rs.getInt("CapacityML") + addSpaces
        ("" + rs.getInt("CapacityML"), 15);
                out += rs.getInt("AllocatedML") + addSpaces
        ("" + rs.getInt("AllocatedML"), 15);
                out += rs.getInt("UsedML") + addSpaces
        ("" + rs.getInt("UsedML"), 15);
                // Only take the first 10 characters of the date
(YYYY-MM-DD)
                out += rs.getString("DateRecorded").substring(0, 10) +
addSpaces
        (rs.getString("DateRecorded").substring(0, 10), 20);
                 // Add IsActive field (Yes/No or True/False)
                out += rs.getString("IsActive") + addSpaces
        (rs.getString("IsActive"), 5) + "\n";
            }
        } catch (SQLException e) {
           // If something goes wrong, show an error message
            JOptionPane.showMessageDialog(null, "Could  not read
data");
        }
        // Return the final formatted string
        return out;
    }
```

```
      /**
     * Utility method that ensures all text lines up neatly by adding
spaces.
     * If a string is shorter than the width, spaces are added.
     * If longer, it gets shortened and "..." is added.
     */
    private static String addSpaces(String s, int width) {
        if (s == null) {
            s = "";
        }
        if (s.length() > width) {
            s = s.substring(0, width - 3) + "...";
        // Trim and add ellipsis
        }
        // Build a string of spaces to fill the gap
        String newString = "";
        for (int i = 0; i < width - s.length(); i++) {
            newString += " ";
        }
        return newString;
    }
}
```