

Question One:

A local bank stores details of their clients in a text file called bank.txt. Create a folder called BankDetails to include a text file called bank.txt that stores the following information about clients.

NB. Take note of the negative balances.

B8612387;Nkosi; Bongwiwe;34500
K8768767;Ndaba;Lucky;120000
P4565662;Govender;Keegan;-56100
J4198000;Heuwel;Shanice;-19000
M902002;Brent;Brandon;23000

Field 1: Account number
Field 2: Surname
Field 3: Name
Field 4: Balance

1.1 Create an object class called Bank.java and save the file in the BankDetails folder. This class should include code to do the following:
Define the following private fields.

accNo -represents the account number of the client(String)
surname - represents the surname of the client(String)
name - represents the name of the client(String)
balance - represents the balance of the client(double) (2)

1.2 Create a default constructor. (1)
1.3 Create a parameterized constructor to receive and transfer fields to the respective private variables. (2)
1.4 Write a **toString()** method to return information on a client formatted as follows:

```
B8612387      Nkosi      Bongwiwe      34500
P4565662      Govender   Keegan        -56100 (3)
```

1.5 Write a method called **getAccNo()** to return the account number of the client. (1)
1.6 Write a method called **getSurname()** to return the surname of the client. (1)
1.7 Write a method called **getBalance()** to return the balance of the client. (1)
1.8 Write a method called **setBalance()** to receive a value as a parameter and increase the balance by the new value. (2)
1.9 Write a method called **status()** to return a message "Overdraft" if the balance is negative. Return a blank string if the balance is positive. (2)
1.10 Perform the following in the TestBank class.

1.10.1 Create an array named bankArr that stores a maximum of 20 objects of class Bank. Write suitable code to read information from the text file bank.txt. You may be guided by the following steps to populate the array called bankArr.

- Open the text file and initialize a loop to read the data.
- Read a line of text from the text file.
- Separate the attributes of the object.
- Use the attributes to create a new bank object that must be placed in array bankArr.
- Use a counter to keep track of the number of objects in the array. (10)

1.10.2 Write a method called **display()** to display all information from the Bank class. (4)

Sample output:
List of Clients:

```
=====
B8612387      Nkosi      Bongwiwe      34500
K8768767      Ndaba      Lucky         120000
P4565662      Govender   Keegan        -56100
J4198000      Heuwel    Shanice       -19000
M902002      Brent     Brandon       23000
```

1.10.3 Write a method called **sort()** to sort client objects in alphabetical order of the surname. (5)
1.10.4 Write a method called **displayStatus** to display the account number, balance, blank message for a positive balance and "Overdraft" for a negative message. (5)

Sample output:
Status of Clients:

```
=====
B8612387      34500
K8768767      120000
P4565662      -56100      Overdraft
J4198000      -19000      Overdraft
M902002      23000
```

1.10.5 Write a method called **update()** that allows the user to enter an account number. Search for the required account number. If found, prompt the user to enter an amount to deposit. Increase the balance by the amount entered. Display a suitable message if the account number entered was not found. (6)

Example of a possible display:

Enter the account number to make a deposit: **K8768767**
Enter an amount to deposit: **5500**
The updated details for client **K8768767** are:
K8768767 Ndaba Lucky 125500

1.10.6 Write a main method to create an object of the TestBank class. Invoke all methods in the given menu. (4)

SQL: Simple Query Language / Structured Query language

SQL is a computer language aimed to store, manipulate, and retrieve data stored in relational databases.

The language is not case sensitive, i.e. select is the same as SELECT or Select. The use of variable names and reference to specific data in a table however, will follow the normal rules for programming. There are 4 types of SQL's namely: **SELECT INSERT DELETE UPDATE**. Every SQL statement must begin with one of the four words above.

A few **other key words** in the language are:

From, Where, As, And, Or, Order By, Group By, Distinct, In, Between, Like, Avg, Count, Min, Max, Sum, Format, Round, Into, Set, Values, Year, Month, Day, Currency

Generally, a SELECT sql is used to display, to perform functions like count, sum, avg, min, max, etc and can also be used for formatting fields.

NB: Order of a SELECT SQL: Part 1: Select field / s
Part 2: From table / tables
Part 3: Condition / s (Optional)
Part 4: Order By / Group By (Optional)

The word "WHERE" can only be used once in a SQL statement. If you have more than one condition, you may use the word AND to join conditions.

Rules for working with fields from two tables:

Use a Where clause to link the related tables using the common field name. The Where clause is used after reference is made to the tables.

Eg. LearnerTb contains the fields **AdmNo**, Name, Surname, Grade (AdmNo is the primary key in LearnerTb and Grade is the foreign key)
GradeTb contains the fields **Grade**, FormEducator (Grade is the primary key in GradeTb)

A one to many relationship is formed from GradeTb to LearnerTb.

Question 1: Write a sql statement to display all fields from LearnerTb sorted in alphabetical order of surname.

Answer 1: String sql = "SELECT * FROM LearnerTb ORDER BY Surname";

Question 2: Write a sql statement to display the name of the form educator teaching learner with admission number A0124. Note that admission number is in LearnerTb and form educator is in GradeTb.

Answer 2: String sql = "SELECT FormEducator FROM LearnerTb, GradeTb WHERE LearnerTb.Grade = GradeTb.Grade AND AdmNo = 'A0124'";

Question 3: Write a sql statement to display the Admission number, grade and the name of the form educator teaching learner with admission number A0124.

Answer 3: String sql = "SELECT AdmNo, GradeTb.Grade, FormEducator FROM LearnerTb, GradeTb WHERE LearnerTb.Grade = GradeTb.Grade AND AdmNo ='A0124'";

Note: When using two tables and referring to a common field, you must state the table name before the common field. The table name can either be GradeTb or LearnerTb for this example.

General format of INSERT, DELETE and UPDATE sql's.

INSERT: is used to add a new record into a table. There are two types.

INSERT: Two types

Type 1: Actual, Specific data

INSERT INTO (tablename) **VALUES** (number, 'String', #Date#, #Time#)

Eg, Insert the following record into StaffTb (27, Mary, Jones, 23 / 06 / 1992)

Answer: INSERT INTO (StaffTb) **VALUES** (27, 'Mary', 'Jones', #23 / 06 /1992#)

Type 2: Data from memory

INSERT INTO (tablename) **VALUES** ("+#number+", "+String+", "#+Date+#", "#+Time+#")

Eg, int staffNum = 38;
String name = "Betty";
String surname = "Mkhize";
String dateofBirth = "19 / 01 / 1990";

INSERT INTO (StaffTb) **VALUES** ("+staffNum+", "+name+", "+surname+", "#+dateOfBirth+#")

The Delete SQL is used to remove one or more records from a table.

DELETE * FROM (tablename) Where condition

The Update SQL is used to edit / change data in a table.

UPDATE (tablename) **SET** (field = new value) **WHERE** condition.

Question 2

A Durban based company keeps records of its employees in a database called WorkersDB. There are two tables that store details of the employees namely:

WorkersTB and ProjectsTB.

WorkersTB table stores data that is descriptive of each employee. Fields are defined as follows:

| Field Name | Type | Size |
|------------|-----------|------|
| EmpNo | Text | 30 |
| EmpName | Text | 30 |
| EmpSurname | Text | 30 |
| JodDesc | Text | 30 |
| DateJoined | Date/Time | |
| Branch | Text | 30 |
| Salary | Currency | |

The following is an example of Workers TB

| EmpNo | EmpName | EmpSurname | JobDesc | DateJoined | Branch | Salary |
|-------|---------|------------|-------------|------------|--------|---------|
| 001 | Terry | James | Manager | 9/1/2000 | Durban | 20 000 |
| 002 | Annie | Middleton | Gardener | 2/3/2001 | JHB | 5 000 |
| 003 | Maistry | Naidoo | Manager | 1/1/1999 | Durban | 23 000 |
| 004 | Kenny | Luyie | Office | 1/1/2000 | CT | 16 000 |
| 005 | Mandy | Howard | Manager | 4/3/2000 | Durban | 17 500 |
| 006 | Hanna | Burton | Gardener | 4/30/1999 | CT | 6 500 |
| 007 | Kurt | Flounders | Office | 2/1/1998 | JHB | 12 900 |
| 008 | James | Sonny | Manager | 3/2/2004 | CT | 33 000 |
| 009 | Shaun | Naidoo | Maintenance | 3/10/1990 | JHB | 240 000 |
| 010 | Candy | Madison | Office | 10/20/1995 | Durban | 13 000 |
| 011 | Michael | Down | Manager | 10/10/2000 | Durban | 7 500 |
| 012 | Flanny | Hilton | Maintenance | 10/10/2000 | Durban | 17 500 |
| 013 | Colin | Moodley | Manager | 10/10/2000 | Durban | 9 500 |
| 1119 | Caddy | Smith | Manager | 10/10/2000 | Durban | 18 500 |
| 1120 | Lenny | Dargon | Maintenance | 10/10/2000 | Durban | 16 500 |
| 1359 | Henry | Howard | Manager | 10/10/2000 | Durban | 7 500 |

NB. Branch refers to the place where the worker is employed.

ProjectsTB table stores data that is descriptive of each project constructed. Fields are defined as follows:

| Field Name | Type | Size |
|------------|-----------|------|
| ProjectNo | Number | |
| EmpNo | Text | 30 |
| Area | Text | 30 |
| StartDate | Date/Time | |
| EndDate | Date/Time | |

The following is an example of ProjectsTB

| ProjectNo | EmpNo | Area | StartDate | EndDate |
|-----------|-------|--------|-----------|------------|
| 1 | 006 | Durban | 1/1/2007 | 3/30/2007 |
| 2 | 005 | CT | 1/1/2009 | 3/31/2009 |
| 3 | 003 | JHB | 1/1/2008 | 5/3/2008 |
| 4 | 001 | Durban | 2/8/2008 | 6/21/2008 |
| 5 | 002 | CT | 24/4/2008 | 6/30/2008 |
| 6 | 004 | Bloem | 7/1/2008 | 10/23/2008 |
| 7 | 007 | Durban | 8/1/2008 | 10/21/2008 |
| 8 | 008 | Durban | 8/7/2008 | 5/6/2008 |
| 9 | 009 | Durban | 6/4/2008 | 7/5/2008 |
| 10 | 004 | JHB | 21/2/2009 | 10/4/2009 |

NB. Area refers to the place where the project will be constructed.

Complete the following SQL's

2.1 Complete the code in selectAllQuery() by formulating an SQL statement to display all fields from the WorkersTB table sorted according to EmpSurname. (2)

2.2 Complete the code in the method named selectOfficeWorkerQuery() by formulating an SQL statement to display the EmpNo, EmpName, EmpSurname and Branch of all office workers. (3)

2.3 Complete the code in the method named selectYearsEmployedQuery() by formulating an SQL statement to display the EmpNo, EmpName, DateJoined and the Number of Years Worked which is a calculated field. Store the number of ears worked as [YrsWrkd]. (4)

2.4 Complete the code in the method named updateQuery() by formulating an SQL statement to update records in WorkersTB where Job Description is Gardener to Job Description equal to Horticulturist. (3)

2.5 Complete the code in the method named InsertQuery() by formulating an SQL statement to insert the following record into WorkersTB table. 1359, Henry, Howard, Manager, 2000/10/10, Durban, 7500 (3)

2.6 Complete the code in the method named getProjectQuery() that allows the user to enter the Branch and the Start date of a project. Write an SQL to display the Employee number, Employee name, Employee Surname of all employees that worked on projects from the start date entered and from that Branch. (Where clause is required) Test with Data: Durban, 2007/01/01 (5)

2.7 Complete the code in the method named getWorkersQuery() by formulating an SQL statement to display EmpNo, EmpName, JobDesc of all Managers that have completed projects in the Durban Area. (Where clause is required). (5)

2.8 Complete the code in the method named countMaintenanceQuery() by formulating an SQL statement to count and display the amount of maintenance workers. (2)

2.9 Complete the code in the method named deleteQuery() by formulating an SQL statement to enter a project number and delete the respective record from the ProjectsTB table. (3)

2.10 Complete the code in the method named getAreaQuery() by formulating an SQL statement to display the different areas that were used to construct projects, sort in alphabetical order. Eg: Bloem CT Durban Jhb (3)

2.11 Complete the code in the method named groupByBranchQuery() by formulating an SQL statement to display the different Branches and the total salaries paid by each Branch. (5)

2.12 Complete the code in the method named IndefiniteQuery() by formulating an SQL statement to display the Project No and Area of all projects that do not have an End Date. (5)

2.13 Complete the code in the method named AverageSalaryQuery() by formulating an SQL statement to display the average salary of all managers that started employment as of 2000. (5)

All jGrasp data files for this question can be downloaded from www.k12schools.co.za

