01) The following table contains information about bank customers in ABC bank.

|  |  |  |  |
| --- | --- | --- | --- |
| **Account Number** | **Customer Name** | **Account Type** | **Account Balance (as of 2024.01.01)** |
| 1001 | Kamal Dissanayake | savings | Rs. 500 000 |
| 1002 | Namal Perera | current | Rs. 975 000 |
| 1003 | Sithumm Udovita | current | Rs. 100 000 |
| 1004 | Manel Dias | savings | Rs. 1 250 000 |
| 1005 | Chethiya Munasinghe | savings | Rs. 950 000 |
| 1006 | Sanju Perera | current | Rs. 1 500 000 |
| 1007 | Lahiru Karunarathna | savings | Rs. 600 000 |
| 1008 | Tharanga Prasad | savings | Rs. 400 000 |
| 1009 | Shashi Dayarathna | savings | Rs. 250 000 |
| 1010 | Anju Senanayake | current | Rs. 1 100 000 |

1. Using contiguous list data structure input above details and display them.

1. Calculate the interest amount earned by the customer based on their account balance and following bank interest rates.

**Interest Amount = Account Balance × Annual Interest Rate**

* + Account Balance below Rs.250 000 - Annual Interest Rate = 2.5%
  + Account Balance between Rs.250 000 and Rs.500 000 - Annual Interest Rate = 5.0%
  + Account Balance between Rs.500 000 and Rs.750 000 - Annual Interest Rate = 7.5%
  + Account Balance between Rs.750 000 and Rs.100 0000 - Annual Interest Rate = 8.0%
  + Account Balance between Rs.1 000 000 and Rs.1 500 000 - Annual Interest Rate =

9.5%

1. Calculate the Total Account Balance for each customer after one year. **Hint:**

**Total Account Balance = Account Balance + Interest Amount**

1. Sort the customer details according to Total Account Balance using quick sort and display them.

1. Find and display all customers who have more than 1 000 000 in their account after one year period.

1. Identify customers with savings accounts and calculate the total balance in savings accounts.

1. Calculate the percentage of the total balance in savings accounts relative to the total balance in all accounts.

public class Customers {

    private int accountNumber;

    private String customerName;

    private String accountType;

    private double accountBalance;

    public Customers(int accountNumber, String customerName, String accountType, double accountBalance) {

        this.accountNumber = accountNumber;

        this.customerName = customerName;

        this.accountType = accountType;

        this.accountBalance = accountBalance;

    }

    @Override

    public String toString() {

        return accountNumber + " " + customerName + " " + accountType + " Rs. " + accountBalance;

    }

    // Calculate the interest amount based on account balance

    public double calculateInterest() {

        double interestRate;

        if (accountBalance < 250000) {

            interestRate = 2.5;

        } else if (accountBalance < 500000) {

            interestRate = 5.0;

        } else if (accountBalance < 750000) {

            interestRate = 7.5;

        } else if (accountBalance < 1000000) {

            interestRate = 8.0;

        } else if (accountBalance < 1500000) {

            interestRate = 9.5;

        } else {

            interestRate = 0.0; // optional

        }

        return accountBalance \* (interestRate / 100);

    }

    // Calculate the total balance after adding interest

    public double calculateTotalBalance() {

        return accountBalance + calculateInterest();

    }

}

class Test {

    public static void main(String[] args) {

        // Initialize an array of BankCustomers

        Customers[] customers = new Customers[10];

        customers[0] = new Customers(1001, "Kamal Dissanayake"     , "savings", 500000);

        customers[1] = new Customers(1002, "Namal Perera",           "current", 975000);

        customers[2] = new Customers(1003, "Sithumm Udovita",        "current", 100000);

        customers[3] = new Customers(1004, "Manel Dias",             "savings", 1250000);

        customers[4] = new Customers(1005, "Chethiya Munasinghe",    "savings", 950000);

        customers[5] = new Customers(1006, "Sanju Perera",           "current", 1500000);

        customers[6] = new Customers(1007, "Lahiru Karunarathna",    "savings", 600000);

        customers[7] = new Customers(1008, "Tharanga Prasad",        "savings", 400000);

        customers[8] = new Customers(1009, "Shashi Dayarathna",      "savings", 250000);

        customers[9] = new Customers(1010, "Anju Senanayake",        "current", 1100000);

        // Use index-based for-loop to iterate through the array

        for (int i = 0; i < customers.length; i++) {

            Customers customer = customers[i];

            if (customer != null) { // Check for null to avoid NullPointerException

                System.out.println("Customer " + (i + 1) + ":");

                System.out.println(customer);

                System.out.println("Interest Earned: Rs. " + customer.calculateInterest());

                System.out.println("Total Balance After One Year: Rs. " + customer.calculateTotalBalance());

                System.out.println();

            }

        }

    }

}

public class List {

    private int maxSize;

    private int position;

    private Customers[] listEntry;

    // Constructor

    public List(int size) {

        this.maxSize = size;

        this.listEntry = new Customers[maxSize];

        this.position = -1;

    }

    // Check if the list is empty

    public boolean isListEmpty() {

        return position == -1;

    }

    // Check if the list is full

    public boolean isListFull() {

        return position == maxSize - 1;

    }

    // Insert an element at the end of the list

    public void insertLast(Customers x) {

        if (isListFull()) {

            System.out.println("Attempt to insert at the end of a full list");

        } else {

            listEntry[++position] = x;

        }

    }

    // Insert an element at a specific position

    public void insertList(int p, Customers element) {

        if (isListFull()) {

            System.out.println("Attempt to insert an entry into a full list");

        } else if (p < 0 || p > listSize()) {

            System.out.println("Attempt to insert at an invalid position");

        } else {

            // Shift elements to make room for the new element

            for (int i = listSize(); i > p; i--) {

                listEntry[i] = listEntry[i - 1];

            }

            listEntry[p] = element;

            position++;

        }

    }

    // Delete an element from a specific position

    public Customers deleteList(int p) {

        if (isListEmpty()) {

            System.out.println("Attempt to delete an entry from an empty list");            return null;

        } else if (p < 0 || p >= listSize()) {

            System.out.println("Attempt to delete from an invalid position");

            return null;

        } else {

            Customers element = listEntry[p];

            // Shift elements to fill the gap

            for (int i = p; i < listSize() - 1; i++) {

                listEntry[i] = listEntry[i + 1];

            }

            position--;

            return element;

        }

    }

    // Retrieve an element from a specific position

    public Customers retrieveList(int p) {

        if (isListEmpty()) {

            System.out.println("Attempt to retrieve an entry from an empty list");

            return null;

        } else if (p < 0 || p >= listSize()) {

            System.out.println("Attempt to retrieve from an invalid position");

            return null;

        } else {

            return listEntry[p];

        }

    }

    // Replace an element at a specific position

    public void replaceList(int p, Customers x) {

        if (isListEmpty()) {

            System.out.println("Attempt to replace an entry in an empty list");

        } else if (p < 0 || p >= listSize()) {

            System.out.println("Attempt to replace at an invalid position");

        } else {

            listEntry[p] = x;

        }

    }

    // Traverse the list and print all elements

    public void traverseList() {

        for (int i = 0; i <= position; i++) {

            System.out.println(listEntry[i]);

        }

    }

    // Get the current size of the list

    public int listSize() {

        return position + 1;

    }

}