Data structures

Using Java.

:linked list

class Node {

String data;

Node next;

public Node(String data) {

this.data = data;

this.next = null;

}

}

class LinkedList {

Node head;

public LinkedList() {

this.head = null;

}

public void addStudent(String name) {

Node newNode = new Node(name);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void removeStudent(String name) {

if (head == null) {

System.out.println("The list is empty.");

return;

}

if (head.data.equals(name)) {

head = head.next;

return;

}

Node current = head;

while (current.next != null) {

if (current.next.data.equals(name)) {

current.next = current.next.next;

return;

}

current = current.next;

}

System.out.println("Student not found.");

}

public void displayStudents() {

if (head == null) {

System.out.println("The list is empty.");

return;

}

Node current = head;

while (current != null) {

System.out.println(current.data);

current = current.next;

}

}

}

OUTPUT

List of Students: Alice

Bob

Charlie

Updated List of Students:

Alice Charlie

//OOC

Abstraction

// Abstract class representing a Bank Account

abstract class BankAccount {

private String accountNumber;

private double balance;

public BankAccount(String accountNumber) {

this.accountNumber = accountNumber;

this.balance = 0.0;

}

// Abstract method to be implemented by subclasses

public abstract void deposit(double amount);

// Abstract method to be implemented by subclasses

public abstract void withdraw(double amount);

public double getBalance() {

return balance;

}

public String getAccountNumber() {

return accountNumber;

}

}

// Concrete subclass representing a Savings Account

class SavingsAccount extends BankAccount {

private double interestRate;

public SavingsAccount(String accountNumber, double interestRate) {

super(accountNumber);

this.interestRate = interestRate;

}

@Override

public void deposit(double amount) {

// Implement deposit logic for savings account

// Add interest based on the interestRate

double interest = amount \* (interestRate / 100);

double totalAmount = amount + interest;

super.balance += totalAmount;

}

@Override

public void withdraw(double amount) {

// Implement withdraw logic for savings account

if (super.balance >= amount) {

super.balance -= amount;

} else {

System.out.println("Insufficient funds.");

}

}

}

// Concrete subclass representing a Checking Account

class CheckingAccount extends BankAccount {

private double overdraftLimit;

public CheckingAccount(String accountNumber, double overdraftLimit) {

super(accountNumber);

this.overdraftLimit = overdraftLimit;

}

@Override

public void deposit(double amount) {

// Implement deposit logic for checking account

super.balance += amount;

}

@Override

public void withdraw(double amount) {

// Implement withdraw logic for checking account

if (super.balance + overdraftLimit >= amount) {

super.balance -= amount;

} else {

System.out.println("Exceeded overdraft limit.");

}

}

}

public class Main {

public static void main(String[] args) {

// Example usage of the abstract classes and subclasses

SavingsAccount savingsAccount = new SavingsAccount("SAV123", 3.0);

CheckingAccount checkingAccount = new CheckingAccount("CHK456", 1000.0);

savingsAccount.deposit(1000);

savingsAccount.withdraw(500);

checkingAccount.deposit(500);

checkingAccount.withdraw(1500);

System.out.println("Savings Account Balance: $" + savingsAccount.getBalance());

System.out.println("Checking Account Balance: $" + checkingAccount.getBalance());

}

}

Output:

ruby

Copy code

Savings Account Balance: $505.0

Checking Account Balance: $-1000.0