Documents – JFSD PROJECTS

Course – 2

Express Banking System

Source Code

- **By Santhanalakshmi**

```javascript
// Node class for the LinkedList
class Node {
  constructor(data) {
    this.data = data;
    this.next = null;
  }
}

// LinkedList class for storing customer accounts
class LinkedList {
  constructor() {
    this.head = null;
  }

  addNode(data) {
    const newNode = new Node(data);
    if (!this.head) {
      this.head = newNode;
    } else {
      let current = this.head;
      while (current.next) {
        current = current.next;
      }
      current.next = newNode;
    }
  }

  display() {
    let current = this.head;
    while (current) {
      console.log(current.data);
      current = current.next;
    }
  }
}



// TreeNode class for storing account transactions
class TreeNode {
  constructor(data) {
    this.data = data;
    this.left = null;
    this.right = null;
  }
}
```

```
// BankingSystem class
class BankingSystem {
  constructor() {
    this.accounts = new LinkedList();
  }

  createAccount(accountNumber, accountHolder, balance) {
    const accountData = {
      accountNumber,
      accountHolder,
      balance,
      transactions: new TreeNode(null)
    };
    this.accounts.addNode(accountData);
  }

  deposit(accountNumber, amount) {
    let current = this.accounts.head;
    while (current) {
      if (current.data.accountNumber === accountNumber) {
        current.data.balance += amount;
        const transaction = new TreeNode(`Deposited ${amount}`);
        this.addTransaction(current.data.transactions, transaction);
        console.log(`Deposited ${amount} to account ${accountNumber}`);
        return;
      }
      current = current.next;
    }
    console.log("Account not found");
  }

  withdraw(accountNumber, amount) {
    let current = this.accounts.head;
    while (current) {
      if (current.data.accountNumber === accountNumber) {
        if (current.data.balance >= amount) {
          current.data.balance -= amount;
          const transaction = new TreeNode(`Withdrew ${amount}`);
          this.addTransaction(current.data.transactions, transaction);
          console.log(`Withdrew ${amount} from account
${accountNumber}`);
          return;
        } else {
          console.log("Insufficient balance");
          return;
        }
      }
      current = current.next;
    }
    console.log("Account not found");
  }

  addTransaction(root, transaction) {
    if (!root) {
      root = transaction;
    } else {
      if (!root.left) {
        root.left = transaction;
      } else if (!root.right) {
        root.right = transaction;
      } else {
        this.addTransaction(root.left, transaction);
      }
    }
  }

  displayTransactions(accountNumber) {
    let current = this.accounts.head;
    while (current) {
      if (current.data.accountNumber === accountNumber) {
        this.displayTransactionTree(current.data.transactions);
        return;
      }
      current = current.next;
    }
    console.log("Account not found");
  }

  displayTransactionTree(root) {
    if (root) {
      console.log(root.data);
      this.displayTransactionTree(root.left);
      this.displayTransactionTree(root.right);
    }
  }
}

// Create a new banking system
const bankingSystem = new BankingSystem();

// Create accounts
```

```
bankingSystem.createAccount(1, "John Doe", 1000);
bankingSystem.createAccount(2, "Jane Doe", 500);

// Deposit and withdraw money
bankingSystem.deposit(1, 500);
bankingSystem.withdraw(1, 200);
bankingSystem.deposit(2, 300);

// Display transactions
bankingSystem.displayTransactions(1);
bankingSystem.displayTransactions(2);


 // Traverse accounts
 bankingSystem.accounts.display();
```