**ЛАБОРАТОРНАЯ РАБОТА №7 СОЗДАНИЕ КЛАССОВ И РАБОТА С ОБЪЕКТАМИ**

Цель работы: получить основные понятия и навыки по созданию классов, описанию свойств и работы с объектами.

Вариант № 2

а) Смоделировать структуру банка: (в отчет добавить диаграмму классов)

Создать один объект класса Банк, два объекта – Филиала, и три вклада, задать значения свойств, вывести на экран.

б) Связать между собой классы Банк, Филиал и Вклад, так, чтобы каждый банк содержал филиалы, каждый филиал содержал указание в каком банке он находится, и какие вклады он содержит, каждый вклад – в каком филиале он размещен.

- Добавить в класс Банк множественное свойство «филиалы» (get) и метод «добавить филиал» (add)

- Добавить в класс Филиал свойство «банк» (get) и свойство «вклады» (get), а также методы «добавить вклад» (add), «удалить вклад» (remove). Удалить set для свойства «общая сумма вкладов» и сделать так, чтобы данное свойство рассчитывалось автоматически (на основе свойства «вклады»).

- Добавить в класс «Вклад» свойство «Филиал»,

- Удалить из класса «Вклад» set для свойства «сумма вклада» и добавить метод «пополнить счет (сумма)»

Создать один объект класса Банк, в данный банк добавить два филиала, в каждый из филиалов добавить по два вклада. Вывести на экран вклады одного из филиалов, и общую сумму вкладов по этому филиалу. Выполнить

пополнение счета одного из вкладов и снова вывести информация о вкладах на экран.

в) добавить в класс Банк метод, осуществляющий поиск вклада по ФИО вкладчика. Метод содержит входной параметр ФИО (String) и возвращает значение типа Вклад. Осуществить поиск в main(), вывести сумму найденного вклада на экран, и отобразить, в каком филиале он размещен.

Листинг файла Main.java

import java.util.ArrayList;

import java.util.List;

class Bank {

private String name;

private List<Branch> branches;

public Bank(String name) {

this.name = name;

this.branches = new ArrayList<>();

}

public void addBranch(Branch branch) {

branches.add(branch);

}

public List<Branch> getBranches() {

return branches;

}

public Deposit findDepositByFullName(String fullName) {

for (Branch branch : branches) {

for (Deposit deposit : branch.getDeposits()) {

if (deposit.getFullName().equals(fullName)) {

return deposit;

}

}

}

return null; // Вклад не найден

}

}

class Branch {

private String name;

private Bank bank;

private List<Deposit> deposits;

public Branch(String name, Bank bank) {

this.name = name;

this.bank = bank;

this.deposits = new ArrayList<>();

}

public void addDeposit(Deposit deposit) {

deposits.add(deposit);

}

public double getTotalDeposits() {

double total = 0;

for (Deposit deposit : deposits) {

total += deposit.getAmount();

}

return total;

}

public List<Deposit> getDeposits() {

return deposits;

}

public Bank getBank() {

return bank;

}

public String getName() {

return name;

}

}

class Deposit {

private double amount;

private String fullName; // ФИО вкладчика

private Branch branch;

public Deposit(double amount, String fullName, Branch branch) {

this.amount = amount;

this.fullName = fullName;

this.branch = branch;

}

public double getAmount() {

return amount;

}

public String getFullName() {

return fullName;

}

public Branch getBranch() {

return branch;

}

}

public class Main {

public static void main(String[] args) {

Bank bank = new Bank("My Bank");

// Создаем филиалы

Branch branch1 = new Branch("Branch 1", bank);

Branch branch2 = new Branch("Branch 2", bank);

// Добавляем филиалы в банк

bank.addBranch(branch1);

bank.addBranch(branch2);

// Создаем вклады

Deposit deposit1 = new Deposit(1000, "John Doe", branch1);

Deposit deposit2 = new Deposit(2000, "Jane Smith", branch1);

Deposit deposit3 = new Deposit(1500, "Alice Johnson", branch2);

Deposit deposit4 = new Deposit(3000, "Bob Brown", branch2);

// Добавляем вклады в филиалы

branch1.addDeposit(deposit1);

branch1.addDeposit(deposit2);

branch2.addDeposit(deposit3);

branch2.addDeposit(deposit4);

// Поиск вклада по ФИО

String fullNameToFind = "John Doe";

Deposit foundDeposit = bank.findDepositByFullName(fullNameToFind);

if (foundDeposit != null) {

System.out.println("Deposit found for " + fullNameToFind + ": " + foundDeposit.getAmount());

System.out.println("Deposit located in Branch: " + foundDeposit.getBranch().getName());

} else {

System.out.println("Deposit not found for " + fullNameToFind);

}

}

Листинг классов

Bank.class

import java.util.ArrayList;

import java.util.List;

class Bank {

private String name;

private List<Branch> branches;

public Bank(String name) {

this.name = name;

this.branches = new ArrayList();

}

public void addBranch(Branch branch) {

this.branches.add(branch);

}

public List<Branch> getBranches() {

return this.branches;

}

}

Branch.class

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

class Branch {

private String name;

private Bank bank;

private List<Deposit> deposits;

public Branch(String name, Bank bank) {

this.name = name;

this.bank = bank;

this.deposits = new ArrayList();

}

public void addDeposit(Deposit deposit) {

this.deposits.add(deposit);

}

public void removeDeposit(Deposit deposit) {

this.deposits.remove(deposit);

}

public double getTotalDeposits() {

double total = 0.0;

Deposit deposit;

for(Iterator var3 = this.deposits.iterator(); var3.hasNext(); total += deposit.getAmount()) {

deposit = (Deposit)var3.next();

}

return total;

}

public List<Deposit> getDeposits() {

return this.deposits;

}

public Bank getBank() {

return this.bank;

}

}

Deposit.class

class Deposit {

private double amount;

private Branch branch;

public Deposit(double amount, Branch branch) {

this.amount = amount;

this.branch = branch;

}

public void deposit(double amount) {

this.amount += amount;

}

public double getAmount() {

return this.amount;

}

public Branch getBranch() {

return this.branch;

}

}

Результат выполнения поиска вклада по ФИО представлен на рисунке 7.1.

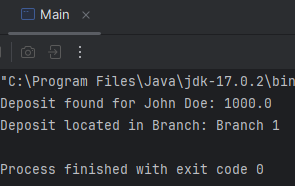


Рисунок 7.1 – Результат выполнения задания

Диаграмма данных классов представлена на рисунке 7.2

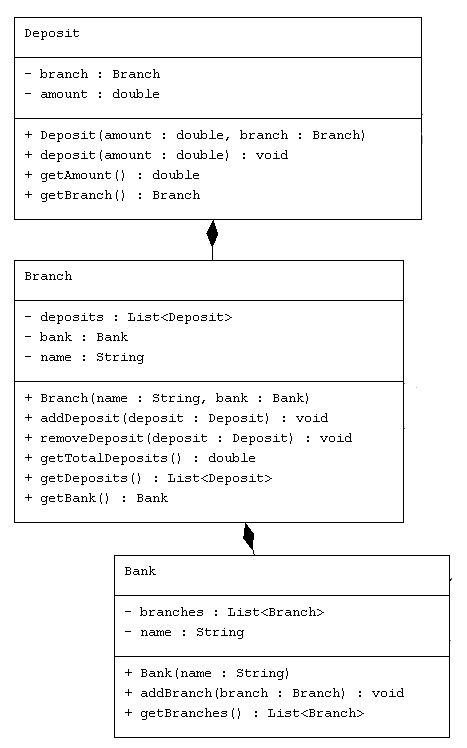


Рисунок 7.2 – Диаграмма классов