دانشکده ریاضی و علوم کامپیوتر

برنامه‌سازی پیشرفته و کارگاه

تمرین کپسوله‌سازی

استاد درس

دکتر مهدی قطعی

استاد دوم

بهنام یوسفی مهر

نگارش

کیانا پهلوان

بهار ۱۴۰۳

فهرست

مقدمه

این هفته Encapsulation و مباحث مربوط به اون رو یاد گرفتید. توی این تمرین قراره از این مبحث و مباحث قبلی‌ای که از شی‌گرایی یاد گرفتید، استفاده کنید تا یک اپلیکیشن ساده برای کنترل حساب‌های بانکی کاربران شبیه‌سازی کنید. در ادامه تمام چیزهایی که شما باید پیاده‌سازی کنید و نکاتی که نیازه بهشون توجه کنید، گفته شده.

حساب بانکی

کلاس ابسترکت BankAccount

این کلاس نشون‌دهنده‌ی حساب‌های بانکی کاربرهاست.

Fields

final private String accountNumber: شماره حساب کاربر

private String accountHolderName: نام کاربر

protected double balance: موجودی حساب کاربر

Methods

public BankAccount(String accountNumber, String accountHolderName, double balance):

کانستراکتور BankAccount که شماره حساب کاربر، نام کاربر و موجودی حساب اون رو مقداردهی می‌کنه.

public abstract void calculateInterest():

این تابع برای انواع مختلف حساب‌های کاربری، متفاوت عمل می‌کنه. توی دو کلاسی که جلوتر معرفی می‌کنیم کارکردش رو توضیح می‌دیم.

public void deposit(double amount):

در اینجا بررسی کنید که مبلغ وارد شده صحیح باشه. مثلاً 20- ورودی اشتباهیه. در صورت صحیح بودن amount، اون رو به موجودی حساب کاربر اضافه می‌کنه. همچنین یک پیام حاوی اینکه چه

مبلغی به چه شماره حسابی اضافه شده، پرینت شه. در صورت نادرست بودن هم باید پیامی پرینت شه که نشون بده مبلغ نادرسته.

public void withdraw(double amount):

این متد برعکس متد قبلی عمل می‌کنه؛ یعنی از حساب کاربر پول برمی‌داره. اینجا هم درستی مبلغ رو بررسی کنید و مثل متد قبلی پیام‌های مناسب رو چاپ کنید. اگر مقدار amount مثبت بود اون رو از موجودی کاربر کم کنید.

public double getBalance()

public String getAccountNumber()

public String getAccountHolderName()

زیرکلاس‌های BankAccount

دو کلاس SavingAccount و TransactionAccount بسازید که از کلاس BankAccount ارث‌بری کنن. این دو کلاس به‌ترتیب حساب‌های پس اندازی و حساب های جاری رو نشون می‌دن. برای هر کدوم از اونا کانستراکتور مناسب رو هم تعریف کنید.

کلاس SavingAccount

Fields

private final double interestRate

که اون رو برای این نوع حساب 3% قرار می‌دیم.

Methods

public SavingsAccount(String accountNumber, String accountHolderName, double initialBalance)

public void calculateInterest():

این متد overrideشده و در اون مقدار interestRate * balance را ورودی متد deposit می‌ذاریم و این مقدار رو به موجودی کاربر اضافه می‌کنیم. اینجا یک پیام حاوی مبلغ افزوده‌شده و نوع حساب پرینت کنید. این مقدار سود حساب کاربره.

کلاس TransactionAccount

Fields

private final double overDraftLimit()

مقدار اون رو برابر با 500 می‌ذاریم. این عدد نشون‌دهنده مقداریه که کاربر می‌تونه بیشتر از موجودی‌ش از حساب برداشت کنه.

Methods

public AccountTransaction (String accountHolderName, String accountNumber, double initialBalance)

public void calculateInterest()

این متد overrideشده و فقط یک پیام پرینت می‌کنه که نشون می‌ده برای این حساب هیچ interestی وجود نداره.

Public void withdraw (double amount):

این متد override شده. اینجا هم، همچنان شرط‌های صحیح بودن ورودی بررسی می‌شه با این تفاوت که می‌شه به اندازه‌ی balance + overDraftLimit از حساب برداشت کرد. پیام‌های مناسب رو در هر مرحله پرینت کنید.

کلاس BankCustomers

Fields

private ArrayList<BankAccount> accountsList

توی این لیست تمام حساب‌های کاربران بانک ریخته می‌شه.

Methods

public void addAccount(BankAccount account)

در اون حساب‌ها به لیست حساب‌های بانک افزوده می‌شن.

public void showAllBalances()

اطلاعات تمامی حساب‌ها (نام کاربر، شماره حساب کاربر و موجودی حساب کاربر) رو پرینت می‌کنه.

public BankAccount findAccount(String accountNumber)

با استفاده از شماره حساب به دنبال اکانت کاربر در accountsList می‌گرده. اگه اون رو پیدا کنه، کاربر رو برمی‌گردونه و اطلاعات اون رو پرینت می‌کنه و اگه نکنه، null رو برمی‌گردونه و یک پیام حاوی خطا پرینت می‌کنه.

کلاس Main

در کلاس Main و تابع main باید یک بانک، سه حساب جاری و سه حساب پس‌اندازی (به ترتیب TransactionAccont و SavingAccount) بسازید و اون‌ها رو به بانک اضافه کنید. با متد findAccount یک حساب موجود و یک حساب ناموجود را جست‌وجو کنید. موجودی اولیه یکی از حساب‌ها رو پرینت کنید. به همون حساب مبلغی اضافه کنید و دوباره موجودی اون رو پرینت کنید. بعد به یک حساب دیگه، یک مبلغ نادرست اضافه کنید. برای یک حساب پس اندازی و یک حساب جاری سود اون رو (تابع calculateInterest) حساب کنید و بعد موجودی اون‌ها رو پرینت کنید. سه برداشت از حساب‌های جاری و سه برداشت دیگه از حساب‌های پس‌اندازی انجام بدید. سه برداشت باید شامل برداشت صحیح، برداشت با عدد نادرست و برداشت بیش از حد مجاز از موجودی باشن. در نهایت اطلاعات تمام حساب‌ها را پرینت کنید.

UML

UML زیر برای راحتی شما در اختیارتون قرار گرفته. توی اون می‌تونید به‌طور خلاصه، همه‌ی کلاس‌های موردنیاز تمرینتون رو به همراه اجزایی که باید داشته باشن (فیلدها و متدها) ببینید. و همچنین ارتباط superclass و subclass هم مشخص شده.