Plan

2023-10-06

1. Practice
2. Working on the Bank application

---

1. Практика
2. Работа над приложением Банк

Theory

▶ **English**

▼ **На русском**

Модификатор `final` может применяться к классам, методам и переменным. Когда он применяется к классу, это означает, что класс не может быть наследован другими классами. Вот как это работает:

1. `final` для класса: Когда класс объявлен как `final`, это означает, что он является "завершенным", и его нельзя расширить или унаследовать. Это полезно, например, когда вы хотите предотвратить создание подклассов, которые могли бы изменить или нарушить работу вашего класса. Пример:

```
final class MyFinalClass {
    // Код класса
}
```

2. `final` для метода: Модификатор `final` также может применяться к методам внутри класса. Когда метод объявлен как `final`, подклассы не могут переопределить этот метод. Это гарантирует, что реализация метода будет постоянной и не изменится в подклассах. Пример:

```
class MyBaseClass {
    final void myFinalMethod() {
        // Код метода
    }
}
```

3. `final` для переменных: Когда переменная объявлена как `final`, ей можно присвоить значение только один раз, и это значение не может быть изменено после этого. Это используется, чтобы создать неизменяемые переменные. Пример:

```
final int myFinalVariable = 10;
```

Использование модификатора `final` помогает создавать более надежный и предсказуемый код в Java, и его следует использовать там, где это необходимо для обеспечения стабильности и безопасности вашего приложения.

Homework

▶ **English**

▼ **На русском**

Задача 1

Дописать методы класса `ATM.java` проекта **Bank**

- написать конструктор
- дописать метод deposit
- дописать метод withdraw
- дописать метод showBalance (не обязательно)

Протестировать дописанные методы.

Задача 2

- Разобраться как работают методы в классе `Bank.java`
- Протестировать работоспособность

Code

code/Lesson_23/src/animal/Animal.java

```
package animal;

/**
 * @author Andrej Reutow
 * created on 06.10.2023
 */
```

```java
public class Animal {

    private final String name;

    public Animal(String name) {
        this.name = name;
    }

    final public void voice() {
        System.out.println("Some sound");
    }

    protected void eat() {
        System.out.println("Animal eating");
    }

    public String getName() {
        return name;
    }
}
```

code/Lesson_23/src/cat/Cat.java

```java
package cat;

import animal.Animal;

/**
 * @author Andrej Reutow
 * created on 06.10.2023
 */
public class Cat extends Animal{

    public Cat(String name) {
        super(name);
    }

//    @Override
//    public void eat() {
//        System.out.println("Cat eating");
//    }
}
```

code/Lesson_23/src/dog/Dog.java

```java
package dog;

import animal.Animal;

/**
 * @author Andrej Reutow
 * created on 06.10.2023
 */
public class Dog extends Animal {

    public Dog(String name) {
        super(name);
    }

    @Override
    public void eat() {
        System.out.println("Dog eating");
    }
}
```

code/Lesson_23/src/dog/Main.java

```java
package dog;

import cat.Cat;

/**
 * @author Andrej Reutow
 * created on 06.10.2023
 */
public class Main {
    public static void main(String[] args) {
        final Cat cat = new Cat("Angelina");
        cat.voice();

//        cat = null;               // так не работает, т.к. переменная cat
//        cat = new Cat("Vajsa");

        Cat cat1 = cat;

//        cat = cat1; // так не работает
    }
```

```
    public static void changeName(final int name) {
     //   name = 1; // так не работает, т.к. переменная name является не изм
    }
}
```

code/Bank/src/constants/AppConstants.java

```
package constants;

/**
 * @author Andrej Reutow
 * created on 06.10.2023
 */
public class AppConstants {

    public static final String PIN_ERROR = "Неправильный пин-код";

    /**
     * Количесвто карт которое может хранить наш банк (class Bank)
     */
    public static final int DEFAULT_CARDS_SIZE = 50;

    public static final String FUNDS_ERROR_MESSAGE = "У вас не достаточно с

    public static final String CLIENT_AMOUNT_LIMIT = "Банк больше не приним

    public static final String CARD_FOUND_MESSAGE = "Карта с номером: ";

    public static final String BALANCE_ERROR_MESSAGE = "На счету не достато
    public static final String BALANCE_INFO = "Текущий баланс: ";
}
```

code/Bank/src/ATM.java

```
import constants.AppConstants;

/**
 * Класс для представления банкомата и выполнения операций с картами.
 */
public class ATM {

    private final Bank bank;
```

```
    public ATM(Bank bank) {
        this.bank = bank;
    }

    /**
     * Метод для внесения денег на карту.
     *
     * @param card   Карта, на которую вносятся деньги.
     * @param amount Сумма, которую нужно внести.
     */
    public void deposit(BankCard card, double amount) {
        if (this.bank.findCard(card)) {
            card.setBalance(card.getBalance() + amount);
            System.out.println("Сумма " + amount + " внесена на счет");
        }
    }

    /**
     * Метод для снятия денег с карты.
     *
     * @param card   Карта, с которой снимаются деньги.
     * @param amount Сумма, которую нужно снять.
     */
    public void withdraw(BankCard card, double amount) {
        if (!this.bank.findCard(card)) {
            return;
        }

        if (card.getBalance() >= amount) {
            card.setBalance(card.getBalance() - amount);
            System.out.println("Сумма " + amount + " выдана");
        } else {
            System.out.println(AppConstants.BALANCE_ERROR_MESSAGE);
        }
    }

    /**
     * Метод для отображения баланса карты.
     *
     * @param card Карта, баланс которой нужно отобразить.
     */
    public void showBalance(BankCard card) {
```

```
            if (!this.bank.findCard(card)) {
                return;
            }

            System.out.println(AppConstants.BALANCE_INFO + card.getBalance());
        }
    }
```

code/Bank/src/Bank.java

```
import constants.AppConstants;

/**
 * Класс для представления банка и хранения карт.
 */
public class Bank {
    private BankCard[] cards;
    private int maxCards;
    private int numCards;

    public Bank(int maxCards) {
        this.maxCards = maxCards;
        this.cards = new BankCard[maxCards];
    }

    public Bank() {
        this.maxCards = AppConstants.DEFAULT_CARDS_SIZE; // 50
        this.cards = new BankCard[this.maxCards]; // cards.l = 50
    }


    // Конструкторы и геттеры/сеттеры

    /**
     * Метод для добавления новой карты в банк.
     *
     * @param card Карта, которую нужно добавить.
     */
    public void addCard(BankCard card) {
        //AndreReutow
        String generatedCardNumber = card.getFirstName() + card.getLastName
        card.setCardNumber(generatedCardNumber);
        card.setBalance(0);
        // todo пройтись по всему массиву и найти пустую ячейку для установ
```

```
            // Если пустой ячейки нет, то указать пользователю, что наш банк бо
            // card[] = card;

            boolean isCardExists = findCard(card);
            if (isCardExists) {
                System.out.println("Карта " + card.getCardNumber() + " уже доба
                return;
            }

            boolean isAdded = false;
            for (int i = 0; i < cards.length; i++) {
                if (cards[i] == null) {
                    cards[i] = card;
                    System.out.println("Крата с номерном " + card.getCardNumber
                    isAdded = true;
                    break;
    //                return;
                }
            }

    //        if (isAdded == false) {
            if (!isAdded) {
                // !true  isAdded == false true
                System.out.println(AppConstants.CLIENT_AMOUNT_LIMIT);
            }
        }

        /**
         * Метод для поиска карты в банке.
         *
         * @param card Карта, которую нужно найти.
         * @return Если карточка найдена true, в противном случае false
         */
        public boolean findCard(BankCard card) {
            if (card == null) {
                return false;
            }

            for (int i = 0; i < cards.length; i++) {
                final BankCard currentElement = cards[i];
                if (currentElement != null && currentElement.equals(card)) {
                    System.out.println(AppConstants.CARD_FOUND_MESSAGE + card.g
                    return true;
```

```
                }
            }

            System.out.println(AppConstants.CARD_FOUND_MESSAGE + card.getCardNu
            return false;
        }
    }
```

code/Bank/src/BankCard.java

```
/**
 * Класс для представления банковской карты.
 */
public class BankCard {
    private String cardNumber;
    private double balance;
    private String firstName;
    private String lastName;

    public BankCard(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }


    // Конструкторы и геттеры/сеттеры

    public String getCardNumber() {
        return cardNumber;
    }

    public void setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public String getFirstName() {
```

```
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    /**
     * Метод для сравнения карт по номеру карты.
     *
     * @param object Карта, с которой сравниваем текущую карту.
     * @return true, значения номера карты одинаковые, в противном случае -
     */
    @Override
    public boolean equals(Object object) {
        if (this == object) return true;
        if (object == null || getClass() != object.getClass()) return false

        BankCard bankCard = (BankCard) object;

        return this.firstName.equals(bankCard.firstName)
                && this.lastName.equals(bankCard.lastName);
    }

    @Override
    public int hashCode() {
        return cardNumber.hashCode();
    }
}
```

code/Bank/src/Main.java

```
/**
 * Главный класс для выполнения операций с банком и банкоматом.
 */
public class Main {
```

```
        public static void main(String[] args) {
            Bank santanderBank = initSantanderBank();
            Bank sparkasseBank = initSparkasseBank();

            ATM sparkasseAtm = new ATM(sparkasseBank);
            ATM santanderAtm = new ATM(santanderBank);

            BankCard myBankCard = new BankCard("Andre", "Reutow");

    //          santanderAtm.deposit(myBankCard, 1000);
    //          santanderAtm.withdraw(myBankCard, 100);
    //          santanderAtm.showBalance(myBankCard);
    //
    //          santanderAtm.deposit(myBankCard, 10);
    //          santanderAtm.withdraw(myBankCard, 500);
    //          santanderAtm.showBalance(myBankCard);


            BankCard bankCardSp = new BankCard("Max", "Mustermann");
            santanderAtm.showBalance(bankCardSp);
            sparkasseAtm.showBalance(bankCardSp);
        }

        public static Bank initSantanderBank() {
            Bank bank = new Bank();

            BankCard bankCard1 = new BankCard("Andre", "Reutow");
            BankCard bankCard2 = new BankCard("Andre", "Reutow");
            BankCard bankCard3 = new BankCard("John", "Doe");

            bank.addCard(bankCard1);
            bank.addCard(bankCard2);
            bank.addCard(bankCard3);

            return bank;
        }

        public static Bank initSparkasseBank() {
            Bank sparkasseBank = new Bank(10);
            BankCard bankCardSp = new BankCard("Max", "Mustermann");
            sparkasseBank.addCard(bankCardSp);

            return sparkasseBank;
```

```
        }
    }
```