Пример 1: Программа "Bank Account"

Цель примера

Продемонстрировать принцип работы со скрытными полями, а также вспомогательными методами для их доступа.

Поэтапное создание программы

В объектно-ориентированном подходе важно строить логику программы на их взаимодействии. Однако, зачастую предоставлять прямой доступ к своим свойствам бывает опасно, поскольку их можно неконтролируемо изменять. Для того, чтобы этого избежать, все свойства класса скрывают от внешнего доступа и оставляют только вспомогательные методы для взаимодейтсвия с ними.

Методы для получения состояния называются геттерами (`getter`), а для изменения состояния – сеттерами (`setter`).

В рамках демонстрации создадим класс, который представляет собой банковский счет со следующими характеристиками:

- Свойства
  - Сумма денег на счету `balance`
- Методы
  - Операция занесения денег на счет
  - Операция снятия денег со счета

Шаг 1.

Создадим класс `BankAccount` с подготовленными характеристиками:

```
public class BankAccount {

    public int balance;

    public void deposit(int amount) {
        this.balance += amount;
    }

    public void withdraw(int amount) {
        this.balance -= amount;
    }

}
```

ℹ️ **Информация**: Операторы `+=` и `-=` являются сокращенной формой записи `this.balance = balance + amount` и `this.balance = balance - amount` соответственно.

Шаг 2.

Продемонстрируем работу программы:

```
public class BankAccountDemo {

    public static void main(String[] args) {

      BankAccount account = new BankAccount();

      System.out.println("Balance = " + account.balance);

      balance.deposit(500);

      System.out.println("Balance = " + account.balance);

      balance.withdraw(200);

      System.out.println("Balance = " + account.balance);

    }

}
```

Результат работы программы:

```
Balance = 0 Balance = 500 Balance = 300
```

Шаг 3.

Проблема заключается в том, что сейчас можно изменить текущее значение баланса в обход методов `deposit()` и `withdraw()`:

```
public class BankAccountDemo {

    public static void main(String[] args) {

        BankAccount account = new BankAccount();

        System.out.println("Balance = " + account.balance);

        account.balance = 50000;

        System.out.println("Balance = " + account.balance);

        account.balance = 1345;

        System.out.println("Balance = " + account.balance);

    }

}
```

Результат работы программы:

```
Balance = 0 Balance = 50000 Balance = 1345
```

Потенциально это может привести к неожиданным ошибкам, например несанкционированное изменение значения, отсутсвие реакции на изменение или ввод недопустимых значений.

Шаг 4.

Для решения этой проблемы достаточно изменить модификатор доступа к полю `balance` и доступ вне класса к нему будет закрыт. Процесс скрытия данных от внешнего использования называется инкапсуляцией и в Java достигается за счет модификатора доступа `private`:

```
public class BankAccount {

    private int balance;

    public void deposit(int amount) {
        this.balance += amount;
    }

    public void withdraw(int amount) {
        this.balance -= amount;
    }

}
```

Шаг 5.

Для того, чтобы была возможность получить текущее значение свойства, обычно добавляют публичный геттер метод:

```
public class BankAccount {

    private int balance;

    public void deposit(int amount) {
        this.balance += amount;
    }

    public void withdraw(int amount) {
        this.balance -= amount;
    }

    public int getBalance() {
        return this.balance;
    }

}
```

Шаг 6.

С учетом последних правок работа с классом выглядит следующим образом:

```
public class BankAccountDemo {

    public static void main(String[] args) {

      BankAccount account = new BankAccount();

      System.out.println("Balance = " + account.getBalance());

      balance.deposit(500);

      System.out.println("Balance = " + account.getBalance());

      balance.withdraw(200);

      System.out.println("Balance = " + account.getBalance());

    }

}
```

Рекомендации:

- Запустить программу и сравнить результаты;
- Попробовать изменить значение переменной у инкапсулированного поля при помощи метода `void setBalance(int balance)` ;