Державний університет «Одеська Політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота №2
з дисципліни «Об’єктно-орієнтоване програмування»
Тема: «Створення класів та об’єктів»

Виконав:
студент групи АІ-201
Мартинюк Д.В.

Прийняв:
доц. Годовиченко М.А.

Одеса 2021

Цель лабораторной работы:

- изучить синтаксис и процесс создания классов;
- научиться создавать объекты созданных классов.

Ход работы:

1) Теоретическое введение в ООП

2) Реализация классов исходя из методических указаний

3) Выводы

1. Класс TimeSpan

В данном классе вместо стандартных громоздких рассчетов с int'ами было использовано библиотеки для работы с датами и временем.

```
// Constructors
TimeSpan(int hours  int minutes)
{

    // If any wrong input, just zero it out as default value
    if(hours < 0 || minutes < 0)
    {
        System.out.println("Time has been zeroed out cause of invalid input!\nKeep
            mind  next time!");


        hours = minutes = 0;
    }


    // Set current day values as date and given hours/minutes
    currentTime = new Date
            (
                    Year.now().getValue()
                    Calendar.getInstance().get(Calendar.MONTH)
                    Calendar.getInstance().get(Calendar.DATE)
                    hours
                    minutes
            );

}
```

Все дальнейшие функции имеют довольно короткую реализацию благодаря этому.

2. Класс BankAccount

Метод *withdraw*:

В данном методе в мервую очередь идет проверка на то , не пытаемся ли мы списать отрицательное значение либо же 0 - если пытаемся, пользователь узнает об этом и метод завершит свою работу.

Далее мы проверяем можем ли мы вообще списать такую сумму (с учетом комиссии), если можем, то списываем, если нет, то метод завершает свою работу и сообщает об этом пользователю.

```
public void withdraw(double amount)
{

    // Amount to withdraw can't be negative or 0
    if(amount <= 0)
    {
        System.out.println("Impossible to finish withdraw, invalid amount!");
        return ;
    }
    else
    {

        // Check if it is possible to withdraw and stay balance positive or 0
        if(balance >= amount - transactionFee)
        {
            System.out.println("Operation has been successful!");
            balance -= amount - transactionFee;
        }
        else
        {
            System.out.println("Impossible to finish withdraw, invalid balance!");
            return;
        }

    }

}
```

Метод *transfer* :

В данном методе мы пересылаем деньги другому пользователю и чтобы это сделать первым делом проверяем сколько, собственно, денег мы хотим отправить. Если эта сумма меньше 1й условной единицы, метод завершит свою работу и сообщит об ошибке.

Если сумма удовлетворяет мы должны проверить что будет если мы спишем эту сумму(мы не можем уйти в минус), если списание возможно, мы списываем и сообщаем пользователю, если же нет — метод завершает свою работу и также сообщает пользователю об ошибке.

```
public void transfer(BankAccount receiver, double amount)
{

    if(amount < 1)
    {
        System.out.println("Can't transfer incorrect amount!");
        return ;
    }
    else
    {

        // Check if we can transfer validly
        if(this.balance >= amount + transactionFee)
        {
            this.balance -= amount + transactionFee;
            receiver.balance += amount;

            System.out.println("Transfer has been successfully done!");
        }
        else
        {
            System.out.println("Can't transfer, invalid balance!");
            return;
        }

    }

}
```

3. Класс Student:

В данных трех методах был использован функционал библиотечной коллекции Vector чтобы работать с набором дисциплин Студента.

```
public void addCourse(String discipline)
{
    this.disciplines.add(discipline);
}
public void dropAll()
{
    this.disciplines.clear();
}
public int getCourseCount()
{
   return this.disciplines.size();
}
```

Выводы к лабораторной работе:

В ходе выполнения лабораторной работы номер 2 было изучено основные приемы при работе с ООП в языке программирования Java, было рассмотрено делегирование конструкторов(довольно специфичное), библиотечные функции для работы с датами , а также коллекция Vector в рамках необходимого функционала.