

Федеральное государственное автономное образовательное учреждение высшего  
образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1

Вариант №310812

Выполнил:  
Решетников Сергей Евгеньевич  
Группа Р3108  
Проверил:

Санкт-Петербург 2024

## Оглавление

1. Задание.....	3
2. UML-диаграмма.....	3
3. Исходный код программы.....	4
4. Результат работы программы.....	4
5. Вывод.....	4

# 1. Задание

## Комментарии

Цель работы: на простом примере разобраться с основными концепциями ООП и научиться использовать их в программах.

Что надо сделать (краткое описание)

1. Ознакомиться с [документацией](#), обращая особое внимание на классы **Pokemon** и **Move**. При дальнейшем выполнении лабораторной работы читать документацию еще несколько раз.
2. Скачать файл `Pokemon.jar`. Его необходимо будет использовать как для компиляции, так и для запуска программы. Распаковывать его не надо! Нужно научиться подключать внешние jar-файлы к своей программе.
3. Написать минимально работающую программу и посмотреть как она работает.

```
Battle b = new Battle();
Pokemon p1 = new Pokemon("Чужой", 1);
Pokemon p2 = new Pokemon("Хищник", 1);
b.addAlly(p1);
b.addFoe(p2);
b.go();
```
4. Создать один из классов покемонов для своего варианта. Класс должен наследоваться от базового класса **Pokemon**. В конструкторе нужно будет задать типы покемона и его базовые характеристики. После этого попробуйте добавить покемона в сражение.
5. Создать один из классов атак для своего варианта (лучше всего начать с физической или специальной атаки). Класс должен наследоваться от класса **PhysicalMove** или **SpecialMove**. В конструкторе нужно будет задать тип атаки, ее силу и точность. После этого добавить атаку покемону и проверить ее действие в сражении. Не забудьте переопределить метод `describe`, чтобы выводилось нужное сообщение.
6. Если действие атаки отличается от стандартного, например, покемон не промахивается, либо атакующий покемон также получает повреждение, то в классе атаки нужно дополнительно переопределить соответствующие методы (см. документацию). При реализации атак, которые меняют статус покемона (наследники **StatusMove**), скорее всего придется разобраться с классом **Effect**. Он позволяет на один или несколько ходов изменить состояние покемона или модификатор его базовых характеристик.
7. Доделать все необходимые атаки и всех покемонов, распределить покемонов по командам, запустить сражение.

## Комментарии

Цель работы: на простом примере разобраться с основными концепциями ООП и научиться использовать их в программах.







Что надо сделать (краткое описание)

1. Ознакомиться с [документацией](#), обращая особое внимание на классы **Pokemon** и **Move**. При дальнейшем выполнении лабораторной работы читать документацию еще несколько раз.
2. Скачать файл `Pokemon.jar`. Его необходимо будет использовать как для компиляции, так и для запуска программы. Распаковывать его не надо! Нужно научиться подключать внешние jar-файлы к своей программе.
3. Написать минимально работающую программу и посмотреть как она работает.

```
Battle b = new Battle();
Pokemon p1 = new Pokemon("Чужой", 1);
Pokemon p2 = new Pokemon("Хищник", 1);
b.addAlly(p1);
b.addFoe(p2);
b.go();
```
4. Создать один из классов покемонов для своего варианта. Класс должен наследоваться от базового класса **Pokemon**. В конструкторе нужно будет задать типы покемона и его базовые характеристики. После этого попробуйте добавить покемона в сражение.
5. Создать один из классов атак для своего варианта (лучше всего начать с физической или специальной атаки). Класс должен наследоваться от класса **PhysicalMove** или **SpecialMove**. В конструкторе нужно будет задать тип атаки, ее силу и точность. После этого добавить атаку покемону и проверить ее действие в сражении. Не забудьте переопределить метод `describe`, чтобы выводилось нужное сообщение.
6. Если действие атаки отличается от стандартного, например, покемон не промахивается, либо атакующий покемон также получает повреждение, то в классе атаки нужно дополнительно переопределить соответствующие методы (см. документацию). При реализации атак, которые меняют статус покемона (наследники **StatusMove**), скорее всего придется разобраться с классом **Effect**. Он позволяет на один или несколько ходов изменить состояние покемона или модификатор его базовых характеристик.
7. Доделать все необходимые атаки и всех покемонов, распределить покемонов по командам, запустить сражение.

Введите вариант:

Ваши покемоны:

<b>Groudon</b>  <b>Атаки:</b> ✓ Overheat ✓ Hammer Arm ✓ Rock Polish ✓ Stone Edge	<b>Kabuto</b>  <b>Атаки:</b> ✓ Rock Polish ✓ Ancient Power ✓ Ice Beam	<b>Kabutops</b>  <b>Атаки:</b> ✓ Rock Polish ✓ Ancient Power ✓ Ice Beam ✓ X-Scissor	<b>Flabebe</b>  <b>Атаки:</b> ✓ Magical Leaf ✓ Double Team	<b>Floette</b>  <b>Атаки:</b> ✓ Magical Leaf ✓ Double Team ✓ Tackle	<b>Florges</b>  <b>Атаки:</b> ✓ Magical Leaf ✓ Double Team ✓ Tackle ✓ Calm Mind
---	---	--	---	--	--

## 2. UML-диаграмма

Доступна по ссылке —

[https://github.com/NF-coder/ITMO\\_repo/tree/main/programming/sem1/lab2/uml.png](https://github.com/NF-coder/ITMO_repo/tree/main/programming/sem1/lab2/uml.png)

### **3. Исходный код программы**

Исходники доступны по ссылке -

[https://github.com/NF-coder/ITMO\\_repo/tree/main/programming/sem1/lab2](https://github.com/NF-coder/ITMO_repo/tree/main/programming/sem1/lab2)

### **4. Результат работы программы**

[https://github.com/NF-coder/ITMO\\_repo/tree/main/programming/sem1/lab2/out.txt](https://github.com/NF-coder/ITMO_repo/tree/main/programming/sem1/lab2/out.txt)

### **5. Вывод**

В ходе данной работы я научился подключать внешние jar файлы и использовать их. Научился работать с документацией, изучил объектно-ориентированный подход на языке Java. Научился работать с классами, конструкторами, полями и модификаторами доступа.