Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление программная инженерия
Образовательная программа системное и прикладное программное обеспечение

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

курса «Программирование»

по теме: «Принципы ООП»

Вариант № 68118

Выполнил студент:
Мухамедьяров Артур Альбертович
группа: P3109

Преподаватель:
Гаврилов А. В.,
Мустафаева А. В.

Санкт-Петербург, 2024 г.

Содержание

Лабораторная работа № 2
Принципы ООП

1. Задание варианта № 68118

, , ,

На основе базового класса `Pokemon` написать свои классы для заданных видов покемонов. Каждый вид покемона должен иметь один или два типа и стандартные базовые характеристики:

- очки здоровья (HP)
- атака (attack)
- защита (defense)
- специальная атака (special attack)
- специальная защита (special defense)
- скорость (speed)

Классы покемонов должны наследоваться в соответствии с цепочкой эволюции покемонов. На основе базовых классов `PhysicalMove`,`SpecialMove` и `StatusMove` реализовать свои классы для заданных видов атак.

Атака должна иметь стандартные тип, силу (power) и точность (accuracy). Должны быть реализованы стандартные эффекты атаки. Назначить каждому виду покемонов атаки в соответствии с вариантом. Уровень покемона выбирается минимально необходимым для всех реализованных атак.

Используя класс симуляции боя `Battle`, создать 2 команды покемонов (каждый покемон должен иметь имя) и запустить бой.

Базовые классы и симулятор сражения находятся в jar-архиве(обновлен 9.10.2018, исправлен баг с добавлением атак и кодировкой). Документация в формате javadoc - здесь.

Информацию о покемонах, цепочках эволюции и атаках можно найти на сайтах https://poke-universe.ru, https://pokemondb.net,https://veekun.com/dex/pokemon

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 PokemonЧужой("", 1);
Pokemon p2 = new PokemonХищник("", 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.

, , ,

2. Выполнение задания.

Задание было выполнено в редакторе кода, позже собрано с помощью `javac` в `jar` файл `itmo_proga_lab1.jar` непосредственно на сервере.

2. 1. Листинги кода

Листинг из файла 1.1

```
import pokemons.*;
import ru.ifmo.se.pokemon.Battle;

public class Main {
    public static void main(String[] args) {
        // initialize the battle
        Battle b = new Battle();

        // initialize pokemons
        Carnivine carnivine = new Carnivine("Carnivine", 1);
        Zorua zorua = new Zorua("Zorua", 1);
        Zoroark zoroark = new Zoroark("Zoroark", 1);
        Igglybuff iglybuff = new Igglybuff("Iglybuff", 1);
        Jigglypuff jigglypuff = new Jigglypuff("Jigglypuff", 1);
        Wigglytuff wigglytuff = new Wigglytuff("Wigglytuff", 1);

        // first team
        b.addAlly(carnivine);
        b.addAlly(zorua);
        b.addAlly(zoroark);

        // second team
        b.addFoe(iglybuff);
        b.addFoe(jigglypuff);
        b.addFoe(wigglytuff);

        // start the battle
        b.go();
    }
}
```

Листинг 1.1: Исходный код программы

3. Результат работы программы.

3. 1. Первый запуск.

4. Вывод

Во время выполнения лабораторной работы я изучил синтаксис языка `Java`, встроенную библиотеку `Math`, научился работать со средством разработки Java (`JDK`). Также в процессе выполения я научился рабоать с типами данных, классами, функциями, массивами и циклами. Полученные мною

знания являются необходимой базой для дальнейшего изучения языка и разработки уже более комлпексных проектов.
Также во время работы над лабораторной, я научился работать с официальной документацией Oracle по встроенной библиотеке Math[2], RandomGenerator[3], а также ознакомился с базовыми командами *NIX[5] и Git[4].

Литература

[1] Ссылка на личный репозиторий GitHub: https://github.com/pozitp/itmo-labs/tree/main/prog/lab1

[2] Ссылка на официальную документацию Oracle для JDK 17 по встроенной библиотеке Math: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html

[3] Ссылка на официальную документацию Oracle для JDK 17 по встроенной библиотеке RandomGenerator: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/random/RandomGenerator.html

[4] Ссылка на официальную документацию Git с базовыми командами для работы с системами конторя версий файлов: https://git-scm.com/docs/giteveryday

[5] Ссылка на официальную документацию GNU по coreutils (базовые команды *NIX): https://www.gnu.org/software/coreutils/manual/coreutils.html