



Федеральное государственное автономное образовательное

учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

## **Отчёт по лабораторной работе 2**

Дисциплина: программирование

Вариант: 92911

Выполнил: студент группы Р3115 Храбров Артём Алексеевич

Проверил: Кулинич Ярослав Вадимович

Дата сдачи: 23.11.2024

Санкт-Петербург, 2024

## **Содержание**

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

## Задание

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)

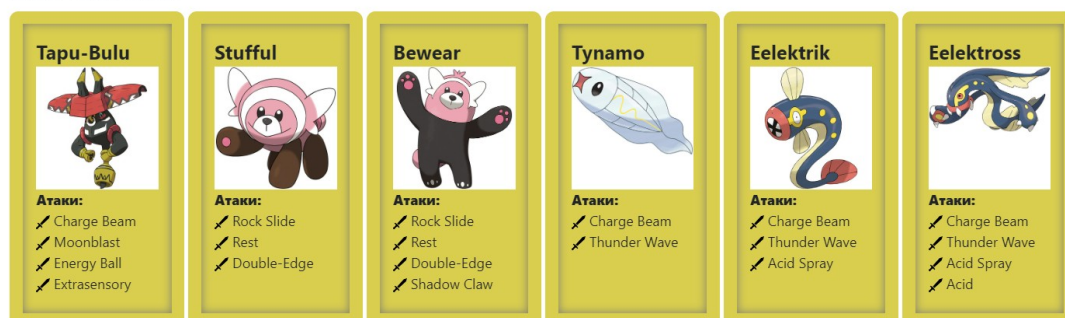
Классы покемонов должны наследоваться в соответствии с цепочкой эволюции покемонов. На основе базовых классов **PhysicalMove**, **SpecialMove** и **StatusMove** реализовать свои классы для заданных видов атак. Все разработанные классы, не имеющие наследников, должны быть реализованы таким образом, чтобы от них нельзя было наследоваться.

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.

Введите вариант:

Ваши покемоны:



## Исходный код

<https://github.com/artem961/itmo/tree/main/PROGA/lab2>

<https://github.com/artem961/itmo/blob/main/PROGA/lab2/documents/output.log>

```

classDiagram
    class Ekans {
        Ekans()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class MoudSw {
        MoudSw()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class EnergyBall {
        EnergyBall()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class TapuBulu {
        TapuBulu(name: String, level: Integer)
    }
    class ShadowClaw {
        ShadowClaw()
        + canCritHit(pokemon: Pokemon, pokemon: Pokemon) void
        + describe() String
    }
    class RockSlide {
        RockSlide()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class DoubleEdge {
        DoubleEdge()
        + applySetDamage(pokemon: Pokemon, damage: Double) void
        + describe() String
    }
    class Berserker {
        Berserker(name: String, level: Integer)
    }
    class Shuffler {
        Shuffler(name: String, level: Integer)
    }
    class Rest {
        Rest()
        + applySetEffect(pokemon: Pokemon) void
        + describe() String
    }
    class CarpetSew {
        CarpetSew()
        + applySetEffect(pokemon: Pokemon) void
        + describe() String
    }
    class Electross {
        Electross(name: String, level: Integer)
    }
    class Electabiz {
        Electabiz(name: String, level: Integer)
    }
    class Tynamo {
        Tynamo(name: String, level: Integer)
    }
    class ThunderBolt {
        ThunderBolt()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class Aerofury {
        Aerofury()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class Axi {
        Axi()
        + applyOptEffect(pokemon: Pokemon) void
        + describe() String
    }
    class Ekans --> TapuBulu
    class MoudSw --> TapuBulu
    class EnergyBall --> TapuBulu
    class TapuBulu --> ShadowClaw
    class TapuBulu --> RockSlide
    class TapuBulu --> DoubleEdge
    class TapuBulu --> Berserker
    class TapuBulu --> Shuffler
    class TapuBulu --> Rest
    class TapuBulu --> CarpetSew
    class TapuBulu --> Electross
    class TapuBulu --> Electabiz
    class TapuBulu --> Tynamo
    class TapuBulu --> ThunderBolt
    class TapuBulu --> Aerofury
    class TapuBulu --> Axi
    class ShadowClaw --> Berserker
    class ShadowClaw --> Shuffler
    class ShadowClaw --> Rest
    class ShadowClaw --> CarpetSew
    class ShadowClaw --> Electross
    class ShadowClaw --> Electabiz
    class ShadowClaw --> Tynamo
    class ShadowClaw --> ThunderBolt
    class ShadowClaw --> Aerofury
    class ShadowClaw --> Axi
    class RockSlide --> Berserker
    class RockSlide --> Shuffler
    class RockSlide --> Rest
    class RockSlide --> CarpetSew
    class RockSlide --> Electross
    class RockSlide --> Electabiz
    class RockSlide --> Tynamo
    class RockSlide --> ThunderBolt
    class RockSlide --> Aerofury
    class RockSlide --> Axi
    class DoubleEdge --> Berserker
    class DoubleEdge --> Shuffler
    class DoubleEdge --> Rest
    class DoubleEdge --> CarpetSew
    class DoubleEdge --> Electross
    class DoubleEdge --> Electabiz
    class DoubleEdge --> Tynamo
    class DoubleEdge --> ThunderBolt
    class DoubleEdge --> Aerofury
    class DoubleEdge --> Axi
    class Berserker --> Berserker
    class Berserker --> Shuffler
    class Berserker --> Rest
    class Berserker --> CarpetSew
    class Berserker --> Electross
    class Berserker --> Electabiz
    class Berserker --> Tynamo
    class Berserker --> ThunderBolt
    class Berserker --> Aerofury
    class Berserker --> Axi
    class Shuffler --> Berserker
    class Shuffler --> Shuffler
    class Shuffler --> Rest
    class Shuffler --> CarpetSew
    class Shuffler --> Electross
    class Shuffler --> Electabiz
    class Shuffler --> Tynamo
    class Shuffler --> ThunderBolt
    class Shuffler --> Aerofury
    class Shuffler --> Axi
    class Rest --> Berserker
    class Rest --> Shuffler
    class Rest --> Rest
    class Rest --> CarpetSew
    class Rest --> Electross
    class Rest --> Electabiz
    class Rest --> Tynamo
    class Rest --> ThunderBolt
    class Rest --> Aerofury
    class Rest --> Axi
    class CarpetSew --> Berserker
    class CarpetSew --> Shuffler
    class CarpetSew --> Rest
    class CarpetSew --> CarpetSew
    class CarpetSew --> Electross
    class CarpetSew --> Electabiz
    class CarpetSew --> Tynamo
    class CarpetSew --> ThunderBolt
    class CarpetSew --> Aerofury
    class CarpetSew --> Axi
    class Electross --> Electross
    class Electross --> Electabiz
    class Electross --> Tynamo
    class Electross --> ThunderBolt
    class Electross --> Aerofury
    class Electross --> Axi
    class Electabiz --> Electross
    class Electabiz --> Electabiz
    class Electabiz --> Tynamo
    class Electabiz --> ThunderBolt
    class Electabiz --> Aerofury
    class Electabiz --> Axi
    class Tynamo --> Electross
    class Tynamo --> Electabiz
    class Tynamo --> Tynamo
    class Tynamo --> ThunderBolt
    class Tynamo --> Aerofury
    class Tynamo --> Axi
    class ThunderBolt --> Electross
    class ThunderBolt --> Electabiz
    class ThunderBolt --> Tynamo
    class ThunderBolt --> ThunderBolt
    class ThunderBolt --> Aerofury
    class ThunderBolt --> Axi
    class Aerofury --> Electross
    class Aerofury --> Electabiz
    class Aerofury --> Tynamo
    class Aerofury --> ThunderBolt
    class Aerofury --> Aerofury
    class Aerofury --> Axi
    class Axi --> Electross
    class Axi --> Electabiz
    class Axi --> Tynamo
    class Axi --> ThunderBolt
    class Axi --> Aerofury
    class Axi --> Axi
  
```

В ходе лабораторной работы я познакомился с основными концепциями ООП в Java. Научился подключать jar файлы к проекту, наследовать классы, переопределять методы.