Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет

ИТМО»

*Факультет программной инженерии и компьютерной техники*

*Направление подготовки: 09.03.04 – Нейротехнологии и программирование*

*Дисциплина «Программирование»*

**Отчет**

**По лабораторной работе №2**

**Вариант №34005**

Выполнил:

Мухин Никита Денисович

Группа: Р3123

Преподаватель:

Данилов Павел Юрьевич

Г. Санкт-Петербург, 2023 г.

**Оглавление**

[Задание 2](#_Toc114643031)

[Основные этапы вычисления 3](#_Toc114643032)

[Программное решение: 6](#_Toc114643033)

[Результат выполнения программы: 6](#_Toc114643034)

[Заключение 7](#_Toc114643035)

[Список литературы. 8](#_Toc114643036)

# 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.

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](https://se.ifmo.ru/documents/10180/660917/Pokemon.jar/a7ce60af-6ee6-47d0-a95e-e5ed9a697bd2) (обновлен 9.10.2018, исправлен баг с добавлением атак и кодировкой). Документация в формате javadoc - [здесь](https://se.ifmo.ru/~tony/doc/).

Информацию о покемонах, цепочках эволюции и атаках можно найти на сайтах [http://poke-universe.ru](http://poke-universe.ru/), [http://pokemondb.net](http://pokemondb.net/),<http://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](https://se.ifmo.ru/~tony/doc/), 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);

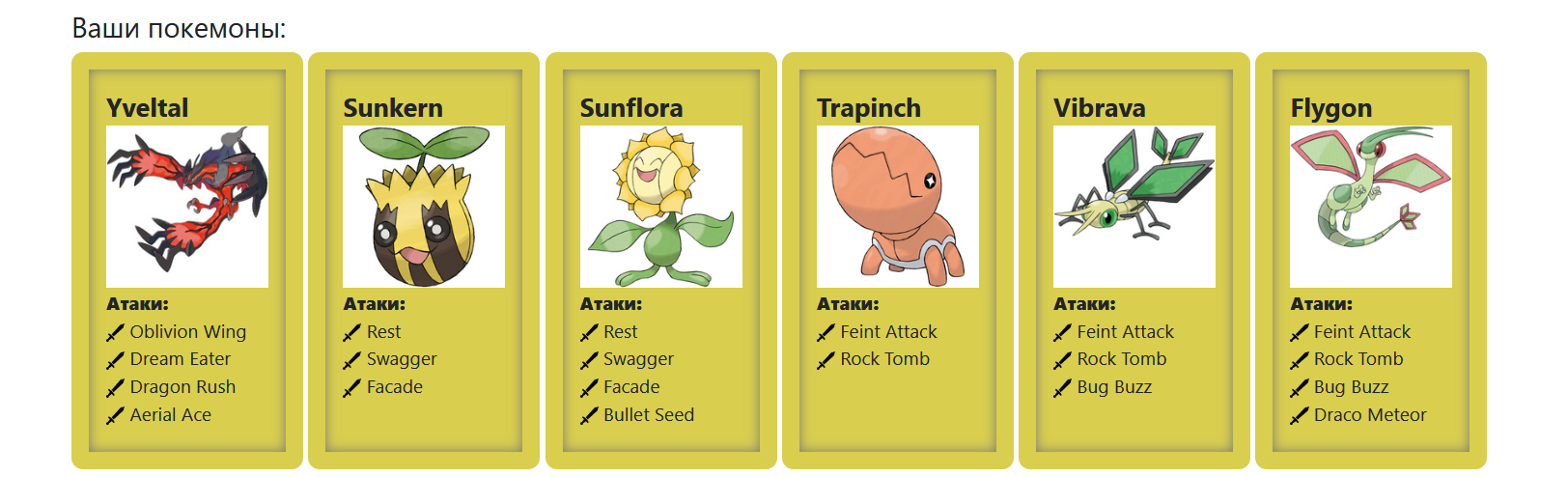
b.go();

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. Cоздать 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.

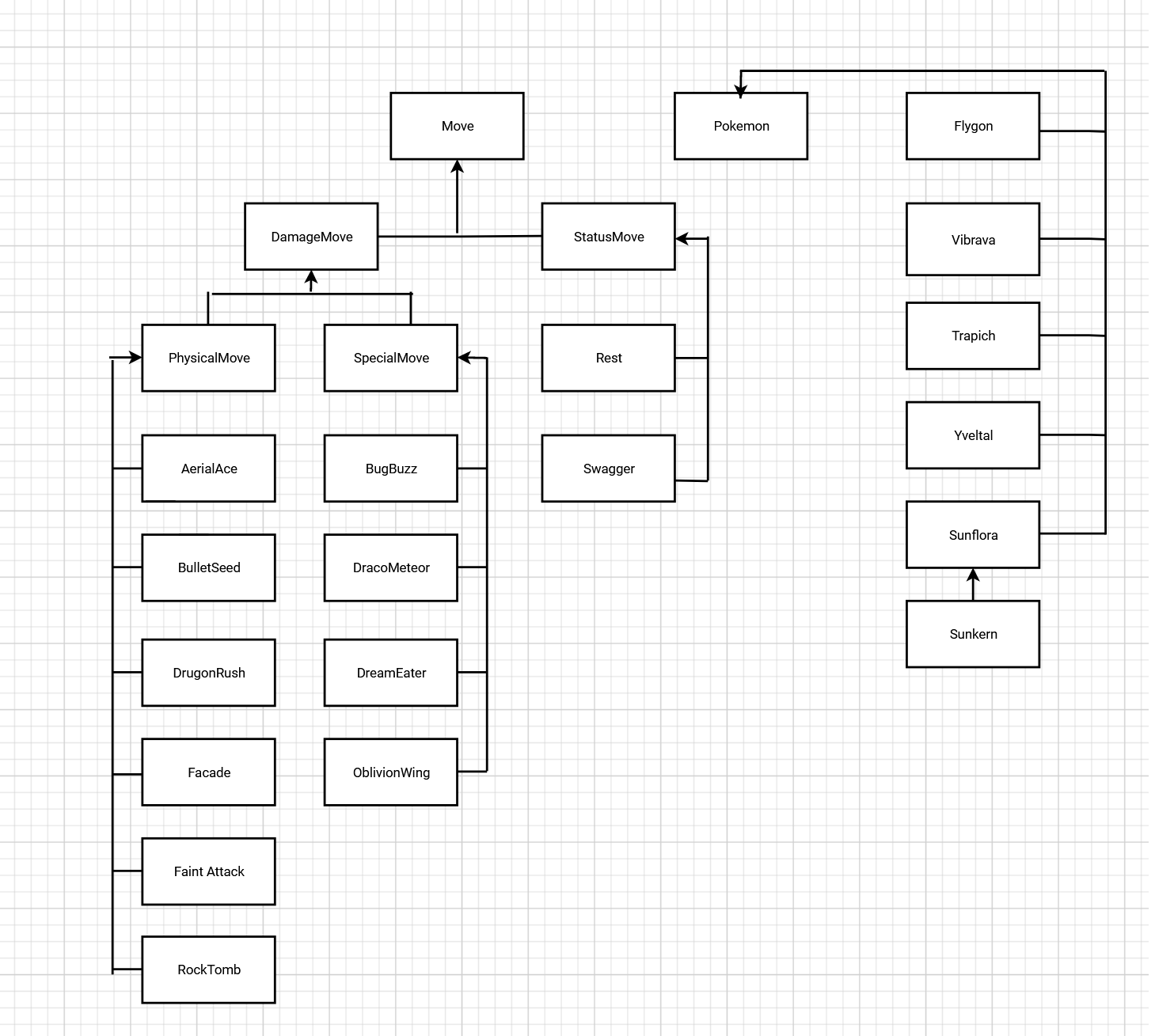
6. Если действие атаки отличается от стандартного, например, покемон не промахивается, либо атакующий покемон также получает повреждение, то в классе атаки нужно дополнительно переопределить соответствующие методы (см. документацию). При реализации атак, которые меняют статус покемона (наследники StatusMove), 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.



# 

# UML-диаграмма классов:



**Код программы**

# Покемоны:

package pokemons;  
  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Flygon extends Pokemon {  
 public Flygon(String name, int level) {  
 super(name, level);  
 setStats(80, 100, 80, 80, 80, 100);  
 setType(Type.*GROUND*, Type.*DRAGON*);  
 setMove(new FeintAttack(), new RockTomb(), new BugBuzz(), new DracoMeteor());  
 }  
}

package pokemons;  
  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Sunflora extends Pokemon {  
 public Sunflora(String name, int level) {  
 super(name, level);  
 setStats(75, 75, 55, 105, 85, 30);  
 setType(Type.*GRASS*);  
 setMove(new Rest(), new Swagger(), new Facade(), new BulletSeed());  
 }  
}

package pokemons;  
  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Sunkern extends Pokemon {  
 public Sunkern(String name, int level) {  
 super(name, level);  
 setStats(30, 30, 30, 30, 30, 30);  
 setType(Type.*GRASS*);  
 setMove(new Rest(), new Swagger(), new Facade());  
 }  
}

package pokemons;  
  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Trapinch extends Pokemon {  
 public Trapinch(String name, int level) {  
 super(name, level);  
 setStats(45, 100, 45, 45, 45, 10);  
 setType(Type.*GROUND*);  
 setMove(new FeintAttack(), new RockTomb());  
 }  
}

package pokemons;  
  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Vibrava extends Pokemon {  
 public Vibrava(String name, int level) {  
 super(name, level);  
 setStats(50, 70, 50, 50, 50, 70);  
 setType(Type.*GROUND*, Type.*DRAGON*);  
 setMove(new FeintAttack(), new RockTomb(), new BugBuzz());  
 }  
}

package pokemons;  
  
import moves.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Yveltal extends Pokemon {  
 public Yveltal(String name, int level) {  
 super(name, level);  
 setStats(126, 131, 95, 131, 98, 99);  
 setType(Type.*DARK*, Type.*FLYING*);  
 setMove(new OblivionWing(), new DreamEater(), new DragonRush(), new AerialAce());  
 }  
}

# Атаки:

package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class AerialAce extends PhysicalMove {  
 public AerialAce() {  
 super(Type.*FLYING*, 60, 100);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.setMod(Stat.*SPEED*, -1);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Aerial Ace!";  
 }  
}

package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class BugBuzz extends SpecialMove {  
 public BugBuzz() {  
 super(Type.*BUG*, 90, 100);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.setMod(Stat.*SPECIAL\_DEFENSE*, -1);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Bug Buzz!";  
 }  
}

package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class BulletSeed extends PhysicalMove {  
 public BulletSeed() {  
 super(Type.*GRASS*, 5, 100);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*SPEED*, -1);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Bullet Seed!";  
 }  
}

package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class DracoMeteor extends SpecialMove {  
 public DracoMeteor() {  
 super(Type.*DRAGON*, 120, 90);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*SPECIAL\_ATTACK*, -2);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Draco Meteor!";  
 }  
}  
package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class DragonRush extends PhysicalMove {  
 public DragonRush() {  
 super(Type.*DRAGON*, 120, 80);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*SPECIAL\_DEFENSE*, -1);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Dragon Rush!";  
 }  
}

package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class DreamEater extends SpecialMove {  
 public DreamEater() {  
 super(Type.*PSYCHIC*, 100, 100);  
 }  
  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*HP*, 50);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Dream Eater!";  
 }  
}  
package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Facade extends PhysicalMove {  
 public Facade() {  
 super(Type.*NORMAL*, 70, 100);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*ATTACK*, 2);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Facade!";  
 }  
}  
package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class FeintAttack extends PhysicalMove {  
 public FeintAttack() {  
 super(Type.*NORMAL*, 40, 100);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.setMod(Stat.*SPEED*, 0);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Feint Attack!";  
 }  
}  
package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class OblivionWing extends SpecialMove {  
 public OblivionWing() {  
 super(Type.*DARK*, 180, 100);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.setMod(Stat.*HP*, -75);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Oblivion Wing!";  
 }  
}  
import ru.ifmo.se.pokemon.\*;  
  
public class Rest extends StatusMove {  
 public Rest() {  
 super(Type.*NORMAL*, 0, 100);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.setMod(Stat.*HP*, 100);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Rest!";  
 }  
}  
package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class RockTomb extends PhysicalMove {  
 public RockTomb() {  
 super(Type.*ROCK*, 60, 80);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*SPEED*, -2);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Rock Tomb!";  
 }  
}  
package moves;  
  
import ru.ifmo.se.pokemon.\*;  
  
public class Swagger extends StatusMove {  
 public Swagger() {  
 super(Type.*NORMAL*, 0, 85);  
 }  
  
 @Override  
 protected void applySelfEffects(Pokemon p) {  
 p.setMod(Stat.*ATTACK*, 2);  
 }  
  
 @Override  
 protected void applyOppEffects(Pokemon p) {  
 p.setMod(Stat.*ATTACK*, 2);  
 }  
  
 @Override  
 protected String describe() {  
 return "is using Swagger";  
 }  
}

# Главный класс (Main.java):

import pokemons.\*;  
import ru.ifmo.se.pokemon.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Battle b = new Battle ();  
 Yveltal p1 = new Yveltal("Ивельталь", 5);  
 Sunkern p2 = new Sunkern("Сункерн", 5);  
 Sunflora p3 = new Sunflora("Подсолнечник", 5);  
 Trapinch p4 = new Trapinch("Трапинч", 5);  
 Vibrava p5 = new Vibrava("Вибрава", 5);  
 Flygon p6 = new Flygon("Флайгон", 5);  
 b.addAlly(p1);  
 b.addAlly(p2);  
 b.addAlly(p3);  
 b.addFoe(p4);  
 b.addFoe(p5);  
 b.addFoe(p6);  
 b.go();  
 }  
}

# Результат работы программы:

Yveltal Ивельталь from the team Blue enters the battle!

Trapinch Трапинч from the team Purple enters the battle!

Yveltal Ивельталь is using Aerial Ace!.

Trapinch Трапинч loses 14 hit points.

Trapinch Трапинч decreases speed.

Trapinch Трапинч is using Feint Attack!.

Yveltal Ивельталь loses 4 hit points.

Yveltal Ивельталь is using Aerial Ace!.

Trapinch Трапинч loses 8 hit points.

Trapinch Трапинч decreases speed.

Trapinch Трапинч faints.

Vibrava Вибрава from the team Purple enters the battle!

Yveltal Ивельталь is using Dream Eater!.

Vibrava Вибрава loses 11 hit points.

Vibrava Вибрава is using Rock Tomb!.

Critical hit!

Yveltal Ивельталь loses 24 hit points.

Vibrava Вибрава decreases speed.

Vibrava Вибрава is using Bug Buzz!.

Critical hit!

Yveltal Ивельталь loses 14 hit points.

Yveltal Ивельталь decreases special defense.

Yveltal Ивельталь faints.

Sunkern Сункерн from the team Blue enters the battle!

Vibrava Вибрава is using Rock Tomb!.

Critical hit!

Sunkern Сункерн loses 9 hit points.

Vibrava Вибрава decreases speed.

Sunkern Сункерн is using Facade!.

Vibrava Вибрава loses 6 hit points.

Sunkern Сункерн increases attack.

Sunkern Сункерн is using Rest!.

Vibrava Вибрава loses 100 hit points.

Vibrava Вибрава faints.

Flygon Флайгон from the team Purple enters the battle!

Flygon Флайгон is using Bug Buzz!.

Sunkern Сункерн loses 15 hit points.

Sunkern Сункерн decreases special defense.

Sunkern Сункерн faints.

Sunflora Подсолнечник from the team Blue enters the battle!

Flygon Флайгон is using Draco Meteor!.

Critical hit!

Sunflora Подсолнечник loses 20 hit points.

Flygon Флайгон decreases special attack.

Sunflora Подсолнечник is using Facade!.

Flygon Флайгон loses 5 hit points.

Sunflora Подсолнечник increases attack.

Flygon Флайгон is using Draco Meteor!.

Sunflora Подсолнечник restores 1 hit points.

Flygon Флайгон decreases special attack.

Sunflora Подсолнечник is using Rest!.

Flygon Флайгон loses 100 hit points.

Flygon Флайгон faints.

Team Purple loses its last Pokemon.

The team Blue wins the battle!

# Заключение

В процессе выполнения лабораторной работы я получил навыки использования

объектно-ориентированного подхода программирования при использовании языка Java.

Научился работать с объектами, классами, супер-классами, конструкторами, полями и модификаторами доступа. Понял основные концепции ООП. Узнал о наследовании как классов, так и конструкторов. Научился создавать UML-диаграммы.

# Список литературы.

1. https://se.ifmo.ru/courses/programming
2. <https://pokemondb.net/pokedex/>
3. <https://javarush.ru/groups/posts/1981-dlja-chego-v-java-nuzhnih-interfeysih>
4. <https://javarush.ru/groups/posts/591-vot-tak-final>
5. <https://www.youtube.com/watch?v=3Xo6zSBgdgk>
6. <https://javarush.ru/groups/posts/1985-raznica-mezhdu-abstraktnihmi-klassami-i-interfeysami>
7. <https://skillbox.ru/media/base/klassy-i-obekty-v-java/>
8. <https://www.jetbrains.com/help/idea/class-diagram.html#analyze_graph>
9. <https://metanit.com/java/tutorial/3.3.php>
10. JAVA from EPAM, 2-е издание, исправленное. И. Н. Блинов, В. С. Романчук
11. <https://javarush.ru/groups/posts/1967-otnoshenija-mezhdu-klassami-nasledovanie-kompozicija-i-agregirovanie->
12. <https://javarush.ru/groups/posts/599-inicializacija-poley>
13. https://javarush.ru/groups/posts/2347-klass-string-v-java