НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

**ЛАБОРАТОРНАЯ РАБОТА №3**

Дисциплина «ПРОГРАММИРОВАНИЕ»

Выполнил студент

Здор Матвей Максимович

Группа № Р3124

Преподаватель: Пименов Данила Дмитриевич

г. Санкт-Петербург

2023

[Задание: 3](#_Toc1119568884)

[Описание работы 3](#_Toc1457161774)

[Отчет 3](#_Toc982404038)

[UML: 3](#_Toc1360956927)

[Результат работы программы: 4](#_Toc1922880165)

[Вывод: 4](#_Toc585785861)

# Задание:

**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().
4. Программа должна содержать как минимум один перечисляемый тип (enum).

**Порядок выполнения работы:**

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

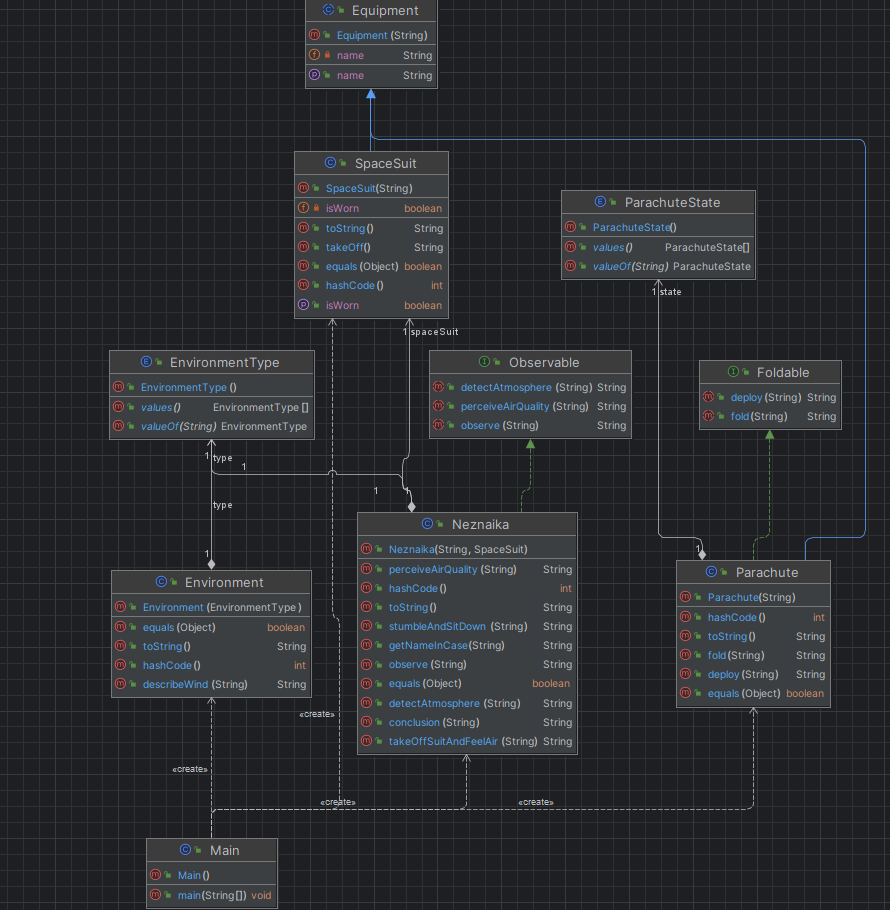
# Описание работы

Но Незнайка опасался напрасно, так как небольшой крылатый парашют, который был у него за спиной, замедлил падение. Правда, от неожиданного толчка ноги у Незнайки подкосились и он сел прямо на землю. Парашют автоматически сложился у него за спиной, приняв вид капюшона. Незнайка огляделся по сторонам и увидел, что окружен кустиками с какими-то крошечными зелеными листиками. Заметив, что листочки на кустах колебались, Незнайка сделал вывод, что вокруг имеется атмосфера, то есть воздух. Ведь обычно листья на деревьях колеблются не сами по себе; в действительности листья колеблет ветер, а ветер, как теперь всем известно, это не что иное, как движение воздуха. Придя к такому умозаключению, Незнайка снял с себя космический скафандр и почувствовал, что не только не задыхается, но даже вполне свободно может дышать. Ему даже показалось, что воздух вокруг гораздо лучше того, которым он дышал на Земле. Но это ему, конечно, только так показалось, потому что он долго пробыл в скафандре и немного отвык от свежего воздуха.

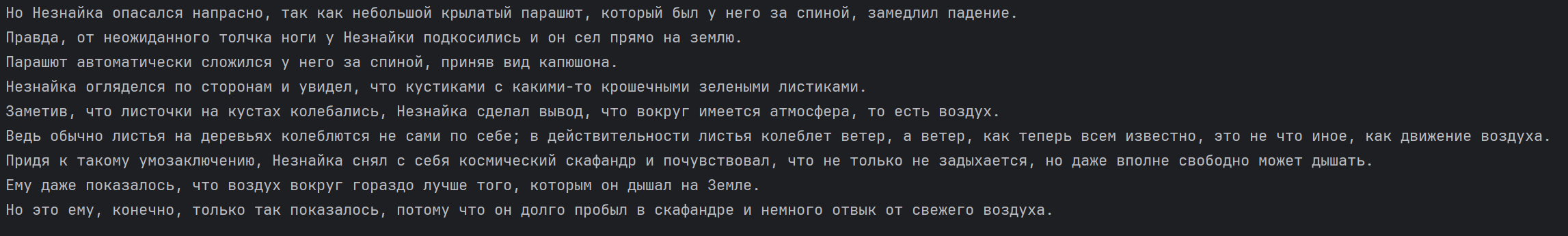
# Отчет

[Код программы](https://github.com/matviikk/itmo/tree/main/term_1/programming/lab3)

# UML:



# Результат работы программы:



Вывод:

Во время выполнения 3й лабораторной работы я познакомился с новыми для меня понятиями. Начну с приципа SOLID. Это акроним, образованный от первых букв пяти основных принципов ООП и расшифровывается как: Single Responsibility Principle, Open/Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle. Этот принцип помогает создавать программистам более устойчивые, гибкие и поддерживаемые системы. Помимо принципа SOLID я познакомился с принципов STUPID, который также является акронимом, означающий Singleton, Tight Coupling, Untestability, Premature Optimization, Indescriptive Naming, Duplication. STUPID как раз означает плохой принцип разработки системы. После этого я познакомился с модулем Object, методы которого ( equals(), toString() и hashCode()) мне нужно было переопределить в своем коде для корректного сравнения, хэширования и представления объектов. Я использовал Абстрактный класс Creature для определения общего поведения и структуры для наследников, позволяя реализовать общие методы и оставлять некоторые методы для реализации в подклассах. Помимо абстрактного класса Creature я использовал интерфейсы LightSource и CreatureMovement для определения контрактов без реализации поведения. Это обеспечивало гибкость и позволяло классам иметь различные реализации этих интерфейсов. В ходе работы был активно использован перечисляемый тип данных (enum), что позволило организовать удобное и надежное представление набора фиксированных констант (например, состояния персонажей). Использование модификаторов static и final в коде обеспечило неизменяемость определенных значений и методов, что повысило безопасность и предсказуемость программы. Например, методы equals(), hashCode(), и toString() в классах были объявлены как final, чтобы предотвратить их изменение в подклассах. В целом, реализация лабораторной работы продемонстрировала глубокое понимание объектноориентированного программирования и принципов SOLID. Работа над проектом обеспечила ценный опыт в применении различных аспектов языка Java, включая наследование, инкапсуляцию, полиморфизм, и абстракцию, что является важным вкладом в развитие навыков программирования.