НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
Факультет ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНЫХТЕХНОЛОГИЙ

ЛАБОРАТОРНАЯ РАБОТА №3

Дисциплина «ПРОГРАММИРОВАНИЕ»

Выполнил студент
Григорьев Никита Александрович
Группа № P3124
Преподаватель: Пименов Данила Дмитриевич

Санкт-Петербург
2023

Оглавление

Задание:

Программа должна удовлетворять следующим требованиям:

1. Доработанная модель должна соответствовать принципам SOLID.
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.

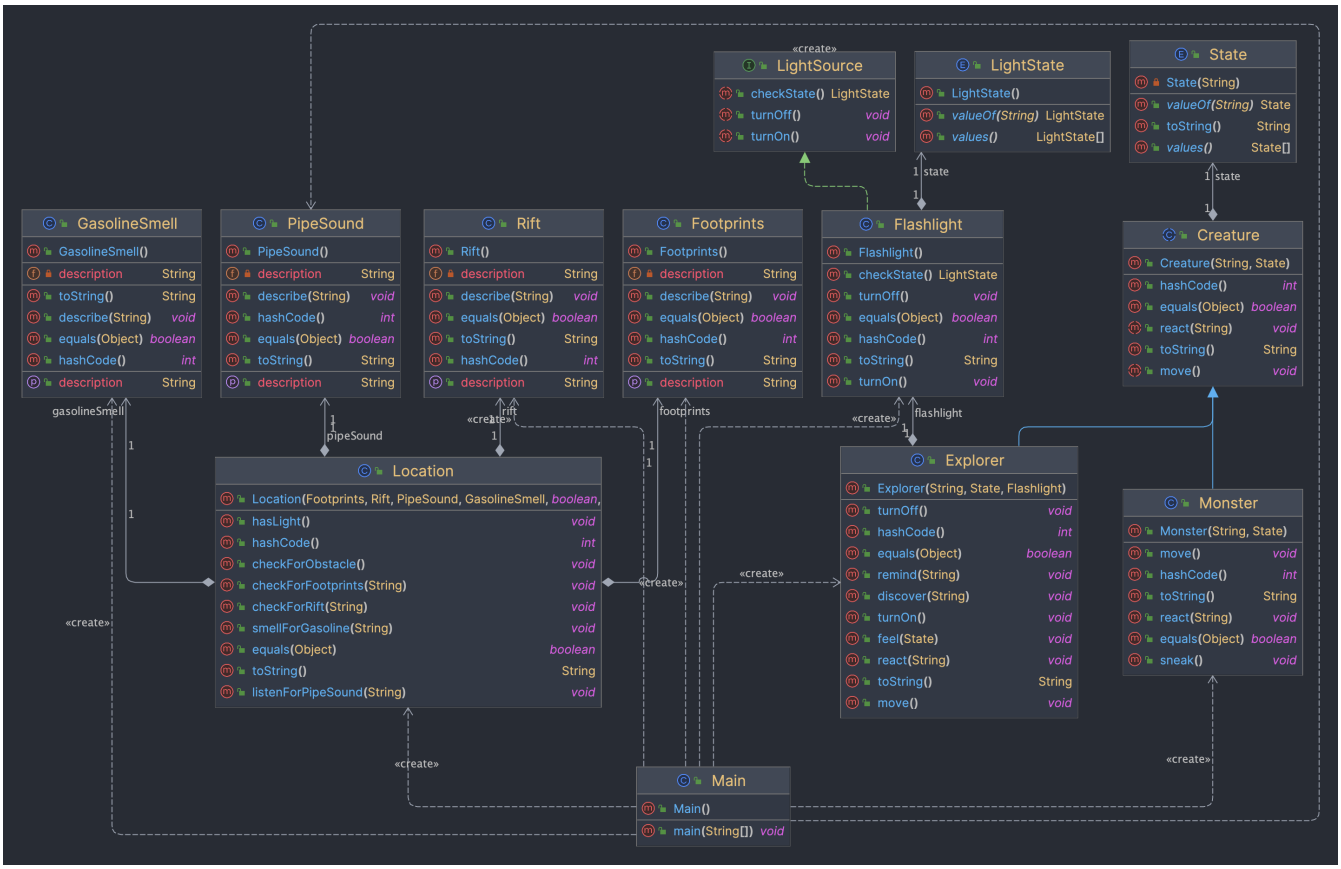
Описание работы

Наше дальнейшее поведение пусть объясняют психологи. Мы понимали, что на это темное, как ночь, кладбище канувших в вечность времен прокралось нечто, подобное монстрам с базы Лейка, и потому не сомневались: впереди нас ждет встреча с неведомым. И все же продолжили путь -- то ли из чистого любопытства, то ли из-за сумятицы в головах или под влиянием самогипноза, а может, нас влекло вперед беспокойство за судьбу Гедни. Денфорт напомнил мне шепотом о подозрительных следах на улице города и прибавил, что он также слышал слабые трубные звуки -- очень важное свидетельство в свете оставленного Лейком отчета о вскрытии неизвестных тварей. Эти звуки, впрочем, могли сойти за эхо, гулко разносившееся по пещерам, изрешетившим горные вершины; похожие звуки доносились и откуда-то снизу, из таинственных недр. Я тоже прошептал ему на ухо свою версию, напомнив, в каком страшном виде предстал перед нашими взорами лагерь Лейка и сколько всего там исчезло: одинокий безумец мог совершить невозможное -- перевалить через эти гигантские хребты и, подобно нам, войти в каменный первобытный лабиринт... Но, поверяя друг другу свои догадки, мы не приходили к единому мнению. Стоя на месте, мы в целях экономии потушили фонарик и только тогда заметили, что в темноте чуть брезжится -- это сверху просачивался свет. Непроизвольно двинулись дальше, включая теперь фонарик лишь изредка, чтобы убедиться, что идем в нужном направлении. Неприятный осадок от недавних следов не покидал нас, тем более что запах бензина становился все сильнее. Разруха усиливалась, мы спотыкались на каждом шагу, и вскоре поняли, что впереди тупик. Наши пессимистические прогнозы оправдались, и виной была та глубокая расщелина, которую мы заметили еще с воздуха. Проход к туннелю был завален -- даже к выходу не пробраться.

Отчет

[Код программы]

UML:



Результат работы программы:

```
Неизвестное создание крадется, почти не издавая звуков.
Денфорт ощущает себя в опасности.
Исследователь ощущает себя в опасности.
Денфорт осторожно двигается вперёд.
Исследователь осторожно двигается вперёд.
Денфорт чувствует любопытство.
Исследователь чувствует любопытство.
Денфорт напоминает о следах
Есть подозрительные следы.
Денфорт напоминает о звуках
Слышен слабые звук трубы.
Денфорт догадывается, что странные звуки могут быть признаком опасности.
Исследователь предполагает, что покинутый лагерь может быть с этим связан.
Денфорт выключил фонарик.
Исследователь выключил фонарик.
Есть свет.
Денфорт обнаружил свет сверху
Денфорт осторожно двигается вперёд.
Исследователь осторожно двигается вперёд.
Чувствуется сильный запах бензина.
Денфорт неприятно себя чувствует.
Исследователь неприятно себя чувствует.
Исследователь обнаружил тупик впереди
Спереди глубокая расщелина.
Денфорт испуган.
Исследователь испуган.
```

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 объектно-ориентированного 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.