Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3-4

Вариант №(37557, 37557.4)

Выполнил

Макогон Ярослав Вадимович

Номер группы: P3118

Проверил

Ермаков М.К.

г. Санкт-Петербург 2024г.

Содержание

Задание

В соответствии с выданным вариантом на основе предложенного текстового отрывка из литературного произведения создать объектную модель реального или воображаемого мира, описываемого данным текстом. Должны быть выделены основные персонажи и предметы со свойственным им состоянием и поведением. На основе модели написать программу на языке Java.

Описание предметной области, по которой должна быть построена объектная модель:

Урашима не мог вымолвить ни слова, пораженный ее неземной красотой. Одним лишь желанием было следовать за ней повсюду. - Да, - только и мог вымолвить он и, подав ей руку, последовал за ней на морское дно. Хрустальная рыбка с золотыми плавниками сопровождала их. Еще до захода солнца они достигли подводного дворца. Он был сделан из кораллов и жемчуга и сверкал так, что было больно глазам. Драконы с нежно-бархатной кожей охраняли вход во дворец. В тишине и роскоши дворца прожил Урашима четыре года вместе с красавицей принцессой. Каждый день море искрилось и сияло в лучах солнца. Они были счастливы, пока однажды Урашима не повстречал маленькую черепаху. Она напомнила ему тот день, когда он ушел к морю. Он вспомнил о своей деревне и своей семье. Принцесса знала, что однажды он вспомнит и затоскует по дому.

Этапы выполнения работы:

1. Получить вариант
2. Нарисовать UML-диаграмму, представляющую классы и интерфейсы объектной модели и их взаимосвязи;
3. Придумать сценарий, содержащий действия персонажей, аналогичные приведенным в исходном тексте;
4. Согласовать диаграмму классов и сценарий с преподавателем;
5. Написать программу на языке Java, реализующую разработанные объектную модель и сценарий взаимодействия и изменения состояния объектов. При запуске программа должна проигрывать сценарий и выводить в стандартный вывод текст, отражающий изменение состояния объектов, приблизительно напоминающий исходный текст полученного отрывка.
6. Продемонстрировать выполнение программы на сервере `helios`.
7. Ответить на контрольные вопросы и выполнить дополнительное задание.

Текст, выводящийся в результате выполнения программы не обязан дословно повторять текст, полученный в исходном задании. Также не обязательно реализовывать грамматическое согласование форм и падежей слов выводимого текста.

Стоит отметить, что цель разработки объектной модели состоит не в выводе текста, а в эмуляции объектов предметной области, а именно их состояния (поля) и

поведения (методы). Методы в разработанных классах должны изменять состояние объектов, а выводимый текст должен являться побочным эффектом, отражающим эти изменения.

Требования к объектной модели, сценарию и программе:

1. В модели должны быть представлены основные персонажи и предметы, описанные в исходном тексте. Они должны иметь необходимые атрибуты и характеристики (состояние) и уметь выполнять свойственные им действия (поведение), а также должны образовывать корректную иерархию наследования классов.
2. Объектная модель должна реализовывать основные принципе ООП - инкапсуляцию, наследование и полиморфизм. Модель должна соответствовать принципам SOLID, быть расширяемой без глобального изменения структуры модели.
3. Сценарий должен быть вариативным, то есть при изменении начальных характеристик персонажей, предметов или окружающей среды, их действия могут изменяться и отклоняться от базового сценария, приведенного в исходном тексте. Кроме того, сценарий должен поддерживать элементы случайности (при генерации персонажей, при задании исходного состояния, при выполнении методов).
4. Объектная модель должна содержать как минимум один корректно использованный элемент каждого типа из списка:
    - абстрактный класс как минимум с одним абстрактным методом;
    - интерфейс;
    - перечисление (enum);
    - запись (record);
    - массив или ArrayList для хранения однотипных объектов;
    - проверяемое исключение.
5. В созданных классах основных персонажей и предметов должны быть корректно переопределены методы `equals()`, `hashCode()` и `toString()`. Для классов-исключений необходимо переопределить метод `getMessage()`.
6. Созданные в программе классы-исключения должны быть использованы и обработаны. Кроме того, должно быть использовано и обработано хотя бы одно unchecked исключение (можно свое, можно из стандартной библиотеки).
7. При необходимости можно добавить внутренние, локальные и анонимные классы.

Программа

Проект на GitHub: https://github.com/Not-N0w/lab_34

Диаграмма классов реализованной объектной модели

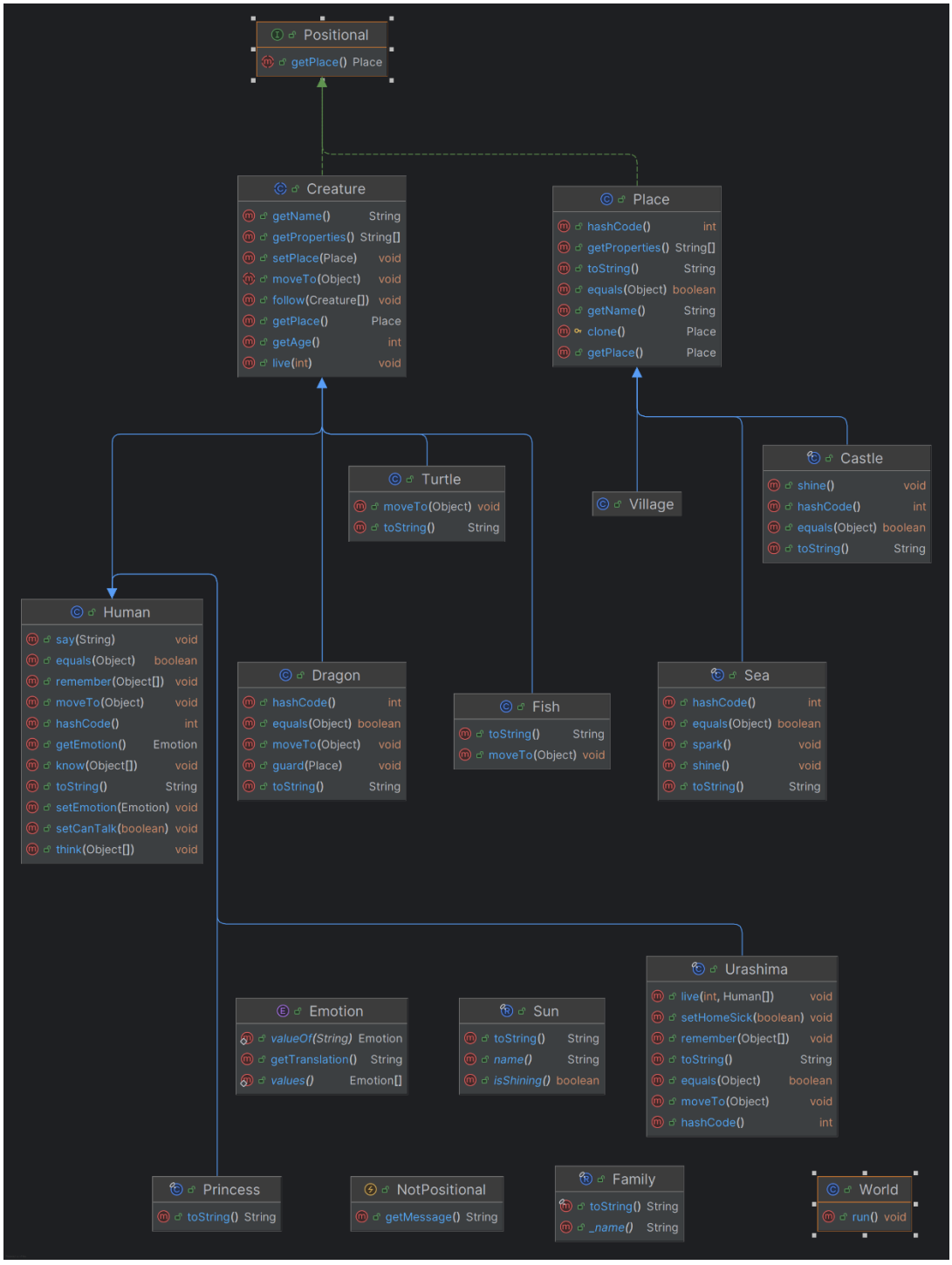

Рисунок 1. Диаграмма классов

Результат работы программы

Урашима {

properties: []

place: Урашима

age: 36

emotion: null

canTalk: True

isHomeSick: False

}

Принцесса {

properties: ['красавица']

place: Принцесса

age: 39

}

-> Урашима {

properties: []

place: Урашима

age: 36

emotion: null

canTalk: False

isHomeSick: False

}

Урашима сказал: 'Да.'

Рыбка {

properties: ['хрустальная', 'с золотыми плавниками']

place: Рыбка

age: 1

}

Принцесса подошел к Дворец

Урашима последовал за Принцесса и очутился у Дворец.

Рыбка последовал за Урашима Принцесса и очутился у Дворец.

Дворец {

properties: ['подводный']

materials: ['из кораллов', 'из жемчуга']

isShining: false

}

Дворец сверкал

-> Дворец {

properties: ['подводный']

materials: ['из кораллов', 'из жемчуга']

isShining: true

}

Дракон1 {

properties: ['с нежно-бархатной кожей']

place: Дворец

guarding: null

age: 76

}

Дракон1 охраняет Дворец

-> Дракон1 {

properties: ['с нежно-бархатной кожей']

place: Дворец

guarding: Дворец

age: 76

}

Дракон2 {

properties: ['с нежно-бархатной кожей']

place: Дворец

guarding: null

age: 85

}

Дракон2 охраняет Дворец

-> Дракон2 {

properties: ['с нежно-бархатной кожей']

place: Дворец

guarding: Дворец

age: 85

}

Урашима {

properties: []

place: Дворец

age: 36

emotion: null

canTalk: True

isHomeSick: False

}

Принцесса {

properties: ['красавица']

place: Дворец

age: 39

}

Урашима прожил счастливо 4 лет с Принцесса

-> Урашима {

properties: []

place: Дворец

age: 40

emotion: Happy

canTalk: True

isHomeSick: False

}

-> Принцесса {

properties: ['красавица']

place: Дворец

age: 39

}

Море {

isShining: false

isSparking: false

}

Черепаха {

properties: ['маленькая']

place: Черепаха

age: 3

}

Урашима подошел к Черепаха

Урашима вспомнил о:

Семья {}

Семья {}

Деревня {}

Принцесса знал о:

Урашима {

properties: []

place: Черепаха

age: 40

emotion: Sad

canTalk: True

isHomeSick: True

}

Выводы

Была написана программа, выполняющая поставленное задание. По ходу написания были изучены базовые концепции языка Java, такие как:

1. Принципы объектно-ориентированного программирования SOLID и STUPID.
2. Класс Object. Реализация его методов по умолчанию.
3. Простое и множественное наследование. Особенности реализации наследования в Java.
4. Понятие абстрактного класса. Модификатор abstract.
5. Понятие интерфейса. Реализация интерфейсов в Java. Отличие интерфейсов от абстрактных классов.
6. Модификаторы default, static и private для методов интерфейса.
7. Перечисляемый тип данных (enum) в Java. Особенности реализации и использования.
8. Тип запись (record) в Java. Особенности использования.
9. Методы и поля с модификаторами static и final.
10. Перегрузка и переопределение методов.
11. Обработка исключительных ситуаций, три типа исключений.
12. Стандартный массив и динамический массив (ArrayList). Основные различия.
13. Вложенные, локальные и анонимные классы.