Применение принципов SOLID

1. Single Responsibility Principle (SRP)

- `Animal` - отвечает только за базовую информацию о животном
- `VetClinic` - отвечает только за медицинские проверки
- `Zoo` - отвечает только за управление зоопарком
- `DIContainer` - отвечает только за управление зависимостями

2. Open/Closed Principle (OCP)

- `Herbo` и `Predator` - открыты для расширения, закрыты для модификации
- `Thing` - базовый класс для вещей, легко расширяется новыми типами
- **Интерфейсы** - позволяют добавлять новые реализации без изменения существующего кода

3. Liskov Substitution Principle (LSP)

- `Monkey`, `Rabbit` могут заменять `Herbo`
- `Tiger`, `Wolf` могут заменять `Predator`
- `Table`, `Computer` могут заменять `Thing`

4. Interface Segregation Principle (ISP)

- `IAlive` - для живых сущностей
- `IInventory` - для инвентаря
- `IVetService` - ветклиника
- `IZooService` - для управления зоопарком

5. Dependency Inversion Principle (DIP)

- `Zoo` зависит от `IVetService`, а не от конкретной реализации
- `DIContainer` управляет всеми зависимостями
- Все сервисы работают через интерфейсы

Структура проекта

```
src/
├── Main.java                    # Точка входа в приложение
├── interfaces/                  # Интерфейсы
│   ├── IAlive.java             # Интерфейс для живых сущностей
│   └── IInventory.java         # Интерфейс для инвентаризации
├── animals/                     # Иерархия животных
│   ├── Animal.java             # Базовый класс животных
│   ├── Herbo.java              # Травоядные животные
```

```
│   ├── Predator.java           # Хищные животные
│   ├── Monkey.java             # Обезьяна
│   ├── Rabbit.java             # Кролик
│   ├── Tiger.java              # Тигр
│   └── Wolf.java               # Волк
├── things/                     # Иерархия вещей
│   ├── Thing.java              # Базовый класс вещей
│   ├── Table.java              # Стол
│   └── Computer.java           # Компьютер
├── services/                   # Сервисы
│   ├── IVetService.java # Интерфейс ветеринарных услуг
│   ├── IZooService.java        # Интерфейс управления зоопарком
│   ├── VetClinic.java  # Ветеринарная клиника
│   └── Zoo.java                # Зоопарк
├── di/                         # DI-контейнер
│   ├── DIContainer.java        # Контейнер зависимостей
│   └── DIConfiguration.java    # Конфигурация зависимостей
└── console/                    # Консольное приложение
    └── ZooConsoleApp.java      # Интерактивное приложение

test/                           # Юнит-тесты
└── java/
    ├── animals/                # Тесты животных
    ├── things/                 # Тесты вещей
    └── services/               # Тесты сервисов
```

Функциональность

Основные возможности

1. **Управление животными**
    - Добавление новых животных с обязательным медосмотром
    - Автоматическая проверка здоровья ветеринарной клиникой
    - Подсчет общего потребления еды всеми животными
2. **Контактный зоопарк**
    - Автоматическое определение животных для контактного зоопарка
    - Только травоядные с уровнем доброты > 5 и прошедшие медосмотр
3. **Инвентаризация**
    - Добавление вещей в инвентарь зоопарка
    - Проверка на дублирование инвентарных номеров
4. **Отчетность**
    - Отчеты по животным
    - Отчеты по вещам
    - Полный отчет о состоянии зоопарка

- Статистика зоопарка

DI-контейнер

Система использует простой DI-контейнер с внедрением зависимостей через конструктор:

```
container.registerSingleton(IVeterinaryService.class, new VetClinic());
container.register(IZooService.class, () -> {
    IVetService veterinaryService = container.get(IVeterinaryService.class);
    return new Zoo("Московский зоопарк", veterinaryService);
});

IZooService zoo = container.get(IZooService.class);
```

Тестирование

Проект покрыт юнит-тестами на **70-75%**:

- **8 тестовых файлов**
- **58 тестовых методов**
- Покрытие всех основных классов
- Использование JUnit 5

После запуска доступно меню:

```
=== ГЛАВНОЕ МЕНЮ ===
1. Добавить животное
2. Добавить вещь
3. Показать отчет по животным
4. Показать отчет по вещам
5. Показать животных для контактного зоопарка
6. Показать общее потребление еды
7. Показать полный отчет
8. Статистика зоопарка
0. Выход
```

Пример использования

1. **Добавьте животное**: Выберите тип, введите имя, количество еды и номер
2. **Пройдите медосмотр**: Система автоматически проверит здоровье
3. **Добавьте вещь**: Выберите тип стола или компьютера
4. **Просмотрите отчеты**: Получите полную информацию о зоопарке