Минобрнауки России

Юго-Западный государственный университет

Факультет фундаментальной и прикладной информатики

Кафедра Программной инженерии

Лабораторная работа №3

По дисциплине: «Методология программной инженерии»

«Паттерн проектирования «Стратегия»

Выполнил: студент группы ПО-51м

Журбенко В.А.

Проверила: к.т.н., профессор

Белова Т. М.

Курск – 2025 г.

Вариант 12. Есть три типа героев – король, воин и маг. Герой ходит, либо бегает, но, возможно, в будущем он также сможет плавать. Реализовать программный продукт, демонстрирующий возможности каждого из этих типов героя.

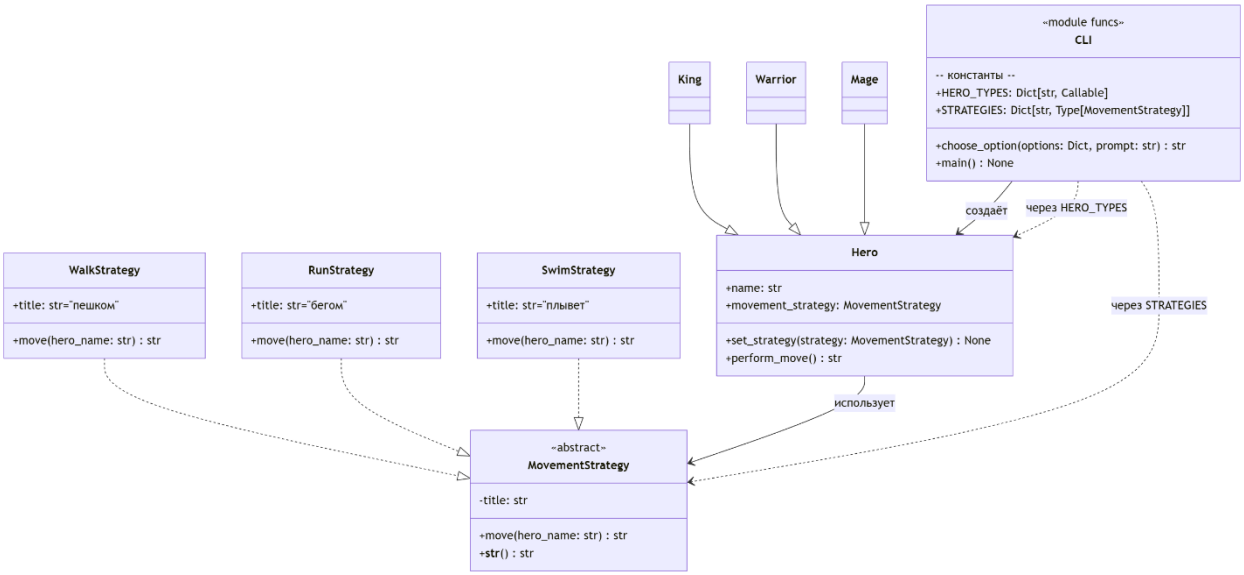

Рисунок 1 – UML-диаграмма

Код программы:

```
"""Console application demonstrating Strategy pattern for hero movement."""
from __future__ import annotations

from dataclasses import dataclass
from typing import Callable, Dict, Type


class MovementStrategy:
    """Base class for movement strategies."""

    title: str = "movement"

    def move(self, hero_name: str) -> str:
        """Return a description of how the hero moves."""
        raise NotImplementedError
```

```
    def __str__(self) -> str:  # pragma: no cover - trivial wrapper
        return self.title

class WalkStrategy(MovementStrategy):
    title = "пешком"

    def move(self, hero_name: str) -> str:
        return f"{hero_name} идет неспешным шагом."

class RunStrategy(MovementStrategy):
    title = "бегом"

    def move(self, hero_name: str) -> str:
        return f"{hero_name} стремительно бежит вперед."

class SwimStrategy(MovementStrategy):
    title = "плывет"

    def move(self, hero_name: str) -> str:
        return f"{hero_name} уверенно плывет по воде."

@dataclass
class Hero:
    """Base hero with ability to switch movement strategies."""

    name: str
    movement_strategy: MovementStrategy

    def set_strategy(self, strategy: MovementStrategy) -> None:
        """Replace the current movement strategy."""
        self.movement_strategy = strategy

    def perform_move(self) -> str:
        """Execute the current movement strategy."""
        return self.movement_strategy.move(self.name)
```

```
class King(Hero):
    pass

class Warrior(Hero):
    pass

class Mage(Hero):
    pass

HERO_TYPES: Dict[str, Callable[[MovementStrategy], Hero]] = {
    "король": lambda strategy: King("Король Артур", strategy),
    "воин": lambda strategy: Warrior("Воин Бьорн", strategy),
    "маг": lambda strategy: Mage("Маг Элоин", strategy),
}

STRATEGIES: Dict[str, Type[MovementStrategy]] = {
    "ходьба": WalkStrategy,
    "бег": RunStrategy,
    "плавание": SwimStrategy,
}

def choose_option(options: Dict[str, Callable], prompt: str) -> str:
    items = list(options.keys())
    while True:
        print(prompt)
        for index, title in enumerate(items, start=1):
            print(f"  {index}. {title}")
        selection = input("Введите номер варианта: ").strip()
        if not selection.isdigit():
            print("Введите номер из списка.\n")
            continue
        index = int(selection) - 1
        if not 0 <= index < len(items):
            print("Вариант вне диапазона. Попробуйте снова.\n")
            continue
```

```
        return items[index]


def main() -> None:
    print("Демонстрация паттерна 'Стратегия' для различных героев.\n")
    strategy_name = choose_option(
        STRATEGIES, "Выберите начальную стратегию передвижения:"
    )
    strategy = STRATEGIES[strategy_name]()

    hero_type_name = choose_option(HERO_TYPES, "Выберите тип героя:")
    hero = HERO_TYPES[hero_type_name](strategy)

    print(f"\n{hero.name} использует стратегию '{hero.movement_strategy}'.")
    print(hero.perform_move())

    while True:
        print("\nХотите сменить стратегию?")
        print("  1. Да")
        print("  2. Нет, завершить программу")
        choice = input("Введите номер варианта: ").strip()
        if choice == "1":
            new_strategy_name = choose_option(
                STRATEGIES, "Выберите новую стратегию передвижения:"
            )
            hero.set_strategy(STRATEGIES[new_strategy_name]())
            print(f"\n{hero.name} теперь {hero.movement_strategy}.")
            print(hero.perform_move())
        elif choice == "2":
            print("\nЗавершение работы. До свидания!")
            break
        else:
            print("Введите 1 или 2.\n")

if __name__ == "__main__":
    main()
```

Результат работы программы:

```
Выберите начальную стратегию передвижения:
  1. ходьба
  2. бег
  3. плавание
Введите номер варианта: Выберите тип героя:
  1. король
  2. воин
  3. маг
Введите номер варианта:
Воин Бьорн использует стратегию 'пешком'.
Воин Бьорн идет неспешным шагом.

Хотите сменить стратегию?
  1. Да
  2. Нет, завершить программу
Введите номер варианта: Выберите новую стратегию передвижения:
  1. ходьба
  2. бег
  3. плавание
Введите номер варианта:
Воин Бьорн теперь плывет.
Воин Бьорн уверенно плывет по воде.

Хотите сменить стратегию?
  1. Да
  2. Нет, завершить программу
Введите номер варианта:
Завершение работы. До свидания!
```