Минобрнауки России

Юго-Западный государственный университет

Факультет фундаментальной и прикладной информатики

Кафедра Программной инженерии

Лабораторная работа №1

По дисциплине: «Методология программной инженерии»

«Паттерн проектирования «Абстрактная фабрика»»

Выполнил: студент группы ПО-51м

Журбенко В.А.

Проверила: к.т.н., профессор

Белова Т. М.

Курск – 2025 г.

Вариант 12. У каждого государства есть символика в виде флага и гимна и есть столица. Использовав паттерн проектирования «Абстрактная фабрика», реализовать программный продукт, демонстрирующий символику и столицу для каждого из следующих государств: Российская Федерация, Китай, Индия.

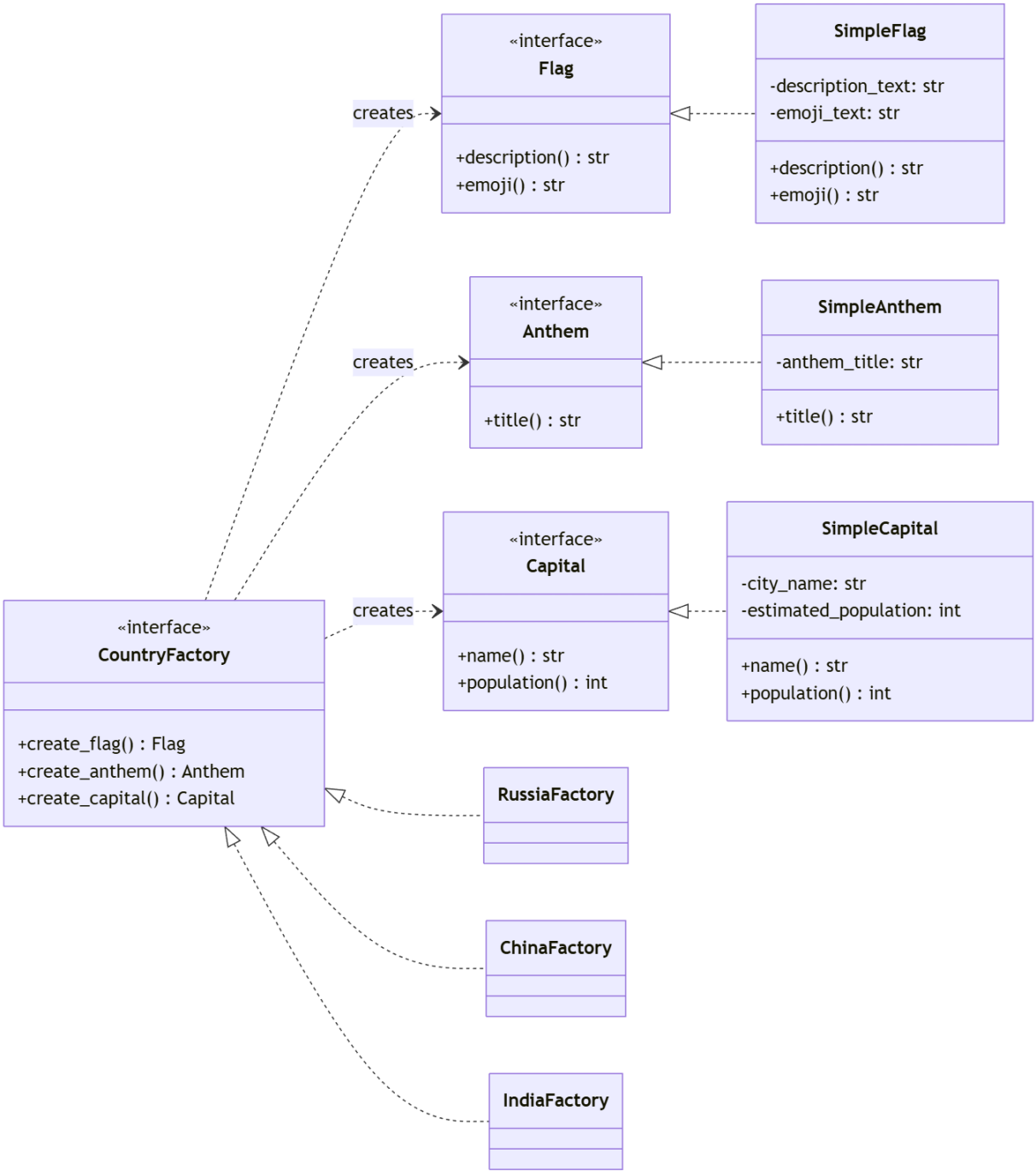

Рисунок 1 – UML-диаграмма

Код программы:

```
from __future__ import annotations
```

```
from abc import ABC, abstractmethod
from dataclasses import dataclass


# --- Abstract product definitions ---------------------------------------
-------


class Flag(ABC):
    """Represents a country flag."""

    @abstractmethod
    def description(self) -> str:
        """Return a human friendly description of the flag."""

    @abstractmethod
    def emoji(self) -> str:
        """Return an emoji representation of the flag."""


class Anthem(ABC):
    """Represents a country anthem."""

    @abstractmethod
    def title(self) -> str:
        """Return the official title of the anthem."""


class Capital(ABC):
    """Represents a country capital city."""

    @abstractmethod
    def name(self) -> str:
        """Return the capital's name."""

    @abstractmethod
    def population(self) -> int:
        """Return the estimated population of the capital."""


# --- Abstract factory definition ----------------------------------------
-------


class CountryFactory(ABC):
    """Abstract factory for creating country specific objects."""

    @abstractmethod
    def create_flag(self) -> Flag:
        """Create the flag for the country."""

    @abstractmethod
    def create_anthem(self) -> Anthem:
        """Create the anthem for the country."""

    @abstractmethod
    def create_capital(self) -> Capital:
        """Create the capital for the country."""
```

```
# --- Concrete product implementations ------------------------------------
------


@dataclass(frozen=True)
class SimpleFlag(Flag):
    description_text: str
    emoji_text: str

    def description(self) -> str:
        return self.description_text

    def emoji(self) -> str:
        return self.emoji_text


@dataclass(frozen=True)
class SimpleAnthem(Anthem):
    anthem_title: str

    def title(self) -> str:
        return self.anthem_title


@dataclass(frozen=True)
class SimpleCapital(Capital):
    city_name: str
    estimated_population: int

    def name(self) -> str:
        return self.city_name

    def population(self) -> int:
        return self.estimated_population


# --- Concrete factories --------------------------------------------------
-------


class RussiaFactory(CountryFactory):
    def create_flag(self) -> Flag:
        return SimpleFlag(
            description_text="Три горизонтальные полосы: белая, синяя,
красная.",
            emoji_text="RU",
        )

    def create_anthem(self) -> Anthem:
        return SimpleAnthem("Государственный гимн Российской Федерации")

    def create_capital(self) -> Capital:
        return SimpleCapital("Москва", 12680000)


class ChinaFactory(CountryFactory):
    def create_flag(self) -> Flag:
        return SimpleFlag(
```

```
            description_text=(
                "Красное полотнище с одной большой и четырьмя меньшими"
                " звездами."
            ),
            emoji_text="CN",
        )

    def create_anthem(self) -> Anthem:
        return SimpleAnthem("Марш добровольцев")

    def create_capital(self) -> Capital:
        return SimpleCapital("Пекин", 21890000)


class IndiaFactory(CountryFactory):
    def create_flag(self) -> Flag:
        return SimpleFlag(
            description_text=(
                "Три горизонтальные полосы: шафрановая, белая с синим
колесом"
                " Ашоки, зеленая."
            ),
            emoji_text="IN",
        )

    def create_anthem(self) -> Anthem:
        return SimpleAnthem("Джана-гана-мана")

    def create_capital(self) -> Capital:
        return SimpleCapital("Нью-Дели", 16790000)


# --- Application logic ----------------------------------------------------
-------


COUNTRY_FACTORIES: dict[str, CountryFactory] = {
    "Россия": RussiaFactory(),
    "Китай": ChinaFactory(),
    "Индия": IndiaFactory(),
}


def get_country_choice() -> str:
    """Prompt the user to choose a country from the available factories."""

    options = list(COUNTRY_FACTORIES.keys())
    options_display = ", ".join(options)
    while True:
        choice = input(f"Выберите страну ({options_display}): ").strip()
        if choice in COUNTRY_FACTORIES:
            return choice
        print("Неизвестная страна. Попробуйте снова.")


def show_country_info(factory: CountryFactory) -> None:
    """Print information about the country's symbols."""

    flag = factory.create_flag()
```

```
    anthem = factory.create_anthem()
    capital = factory.create_capital()

    print("Символика страны:")
    print(f"Флаг: {flag.description()}")
    print(f"Гимн: {anthem.title()}")
    print(
        "Столица: "
        f"{capital.name()} (население ≈ {capital.population():,}
человек)".replace(",", " ")
    )
    print("\n")


if __name__ == "__main__":
    for name in COUNTRY_FACTORIES:
        print(name)
        factory = COUNTRY_FACTORIES[name]
        show_country_info(factory)
```

Результат работы программы:

```
Россия
Символика страны:
Флаг: Три горизонтальные полосы: белая, синяя, красная.
Гимн: Государственный гимн Российской Федерации
Столица: Москва (население ≈ 12 680 000 человек)


Китай
Символика страны:
Флаг: Красное полотнище с одной большой и четырьмя меньшими звездами.
Гимн: Марш добровольцев
Столица: Пекин (население ≈ 21 890 000 человек)


Индия
Символика страны:
Флаг: Три горизонтальные полосы: шафрановая, белая с синим колесом Ашоки, зеленая.
Гимн: Джана-гана-мана
Столица: Нью-Дели (население ≈ 16 790 000 человек)
```