MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA
UNIVERSITATEA TEHNICĂ A MOLDOVEI
FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

Laboratory work 1: SOLID principles

Elaborated: Cucos, Maria
st. gr. FAF-221

Chișinău, 2024

Task: Implement 2 SOLID letters in a simple project

I implemented the Single Responsibility Principle (SRP) and the Open/Closed Principle (OCP) in a python project that manages a cat shelter catalog.

Implementation:

Single Responsibility Principle:

- The `Cat` class is responsible only for holding cat information.

```
class Cat:
  def __init__(self, name, breed, id_number):
    self.name = name
    self.breed = breed
    self.id_number = id_number
```

- The `ShelterCatalog` class is responsible for managing the collection of cats.

```
class ShelterCatalog:
  def __init__(self):
    self.cats = []

  def add_cat(self, cat):
    self.cats.append(cat)

  def remove_cat(self, id_number):
    self.cats = [cat for cat in self.cats if cat.id_number != id_number]

  def get_cat(self, id_number):
    return next((cat for cat in self.cats if cat.id_number == id_number),
None)
```

Open/Closed Principle (OCP):

- The search functionality is implemented using the `Strategy` pattern. The `Strategy` pattern allows to define a family of algorithms, encapsulate each one, and make them interchangeable.
- The `SearchStrategy` abstract base class allows for easy extension of search capabilities without modifying existing code. It defines the common method that all concrete strategies must implement.

```
class SearchStrategy(ABC):
  @abstractmethod
  def search(self, catalog, query):
    pass
```

- The class that uses the strategy is the `CatalogSearch` class.

```
class CatalogSearch:
  def __init__(self, strategy):
    self.strategy = strategy

  def search(self, catalog, query):
    return self.strategy.search(catalog, query)
```

- Concrete Strategie are the actual implementations of the `Strategy` interface. In my project they are `NameSearch` and `BreedSearch`.

```
class NameSearch(SearchStrategy):
  def search(self, catalog, query):
        return  [cat  for  cat  in  catalog.cats  if  query.lower()  in
cat.name.lower()]

class BreedSearch(SearchStrategy):
  def search(self, catalog, query):
        return  [cat  for  cat  in  catalog.cats  if  query.lower()  in
cat.breed.lower()]
```

- New search strategies can be added without modifying existing code like `IDSearch, AdoptionStatusSearch` etc.

Conclusions:

This project uses two SOLID principles - Single Responsibility Principle (SRP) and Open/Closed Principle (OCP) - in a simple cat shelter management system. The project shows how these principles can lead to more maintainable and flexible code. It allows for easy extension without altering existing, working code. This structure would make it simpler to expand the system in the future, such as adding more animal types or implementing new features.