Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «ПИС»

за 5-й семестр

Выполнил:
студент 3 курса
группы ПО-3 (1)
Лущ М. Г.

Проверил:
Лаврущик А.И.

Брест, 2021

Вариант 4

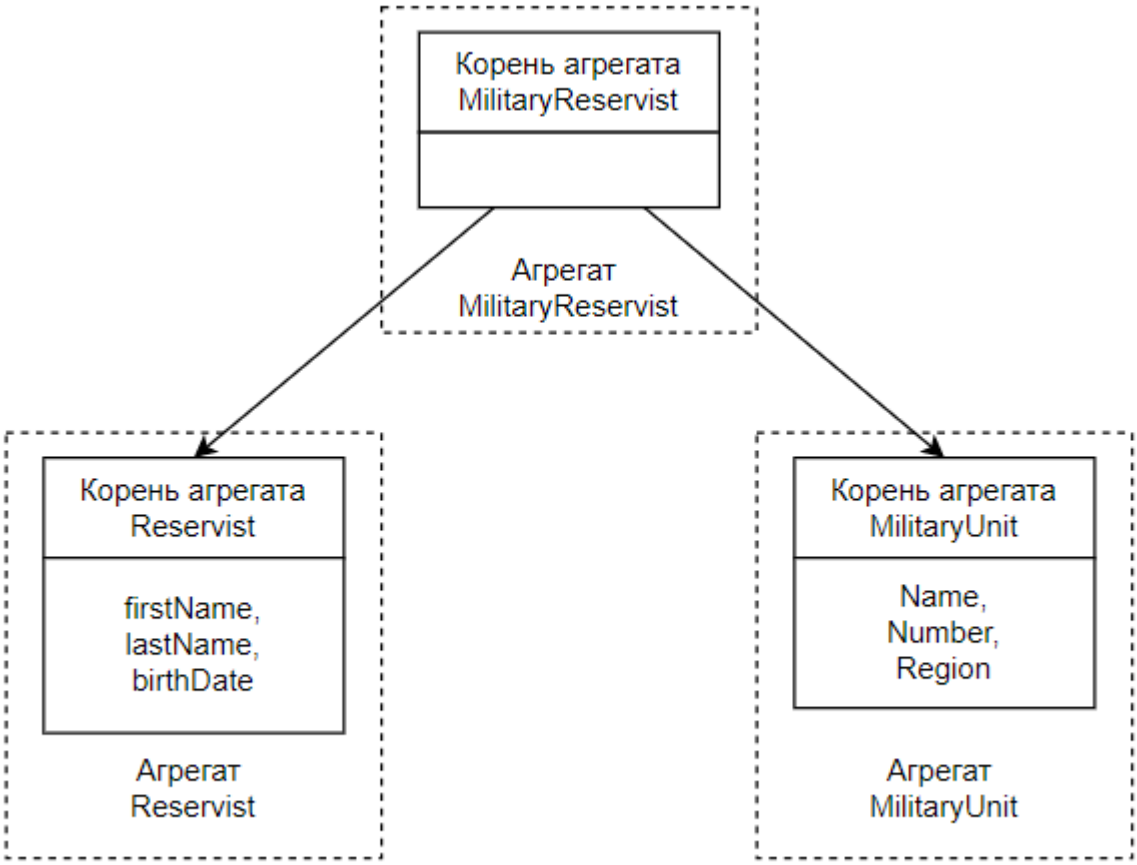
Цель работы: Познакомиться с тактическими шаблонами предметно ориентированного проектирования

Задание для выполнения:

Реализуйте не менее двух агрегатов, сущностей, объектов значений домена Вашей гексагональной архитектуры из предыдущей работы, а также репозитории и не менее двух доменных событий для агрегатов. Один агрегат должен включать в себя сущность-корень агрегата, список дочерних сущностей, каждая сущность содержать наряду с обычными свойствами ещё и объекты-значения. Идентификаторы сущностей должны генерироваться: для корней агрегатов – репозиторием, для внутренних сущностей – самим корнем агрегата. Проверку работы агрегатов организовать путём тестирования (ЛР №4).

Тема: Учёт военнообязанных.

Ход работы



Ревлизация:

MilitaryReservist.php:

```php
<?php

class MilitaryReservist
{
    private int $id;
    private int $militaryUnitId;
    private MilitaryUnit $militaryUnit;
    private int $reservistId;
    private Reservist $reservist;

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }

    /**
     * @param int $id
     */
    public function setId(int $id): void
    {
        $this->id = $id;
    }

    /**
     * @return int
     */
    public function getMilitaryUnitId(): int
    {
        return $this->militaryUnitId;
    }

    /**
     * @param int $militaryUnitId
     */
    public function setMilitaryUnitId(int $militaryUnitId): void
    {
        $this->militaryUnitId = $militaryUnitId;
    }

    /**
     * @return int
     */
    public function getReservistId(): int
    {
        return $this->reservistId;
    }
```

```
    /**
     * @param int $reservistId
     */
    public function setReservistId(int $reservistId): void
    {
        $this->reservistId = $reservistId;
    }

    /**
     * @return MilitaryUnit
     */
    public function getMilitaryUnit(): MilitaryUnit
    {
        return $this->militaryUnit;
    }

    /**
     * @param MilitaryUnit $militaryUnit
     */
    public function setMilitaryUnit(MilitaryUnit $militaryUnit): void
    {
        $this->militaryUnit = $militaryUnit;
    }

    /**
     * @return Reservist
     */
    public function getReservist(): Reservist
    {
        return $this->reservist;
    }

    /**
     * @param Reservist $reservist
     */
    public function setReservist(Reservist $reservist): void
    {
        $this->reservist = $reservist;
    }
}
```

MilitaryUnit.php:

```
<?php

class MilitaryUnit
{
    private int $id;
    private string $name;
    private int $number;
```

```php
    private string $region;

    private array $reservists;

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }

    /**
     * @param int $id
     */
    public function setId(int $id): void
    {
        $this->id = $id;
    }

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }

    /**
     * @param string $name
     */
    public function setName(string $name): void
    {
        $this->name = $name;
    }

    /**
     * @return int
     */
    public function getNumber(): int
    {
        return $this->number;
    }

    /**
     * @param int $number
     */
    public function setNumber(int $number): void
    {
        $this->number = $number;
```

```
    }

    /**
     * @return string
     */
    public function getRegion(): string
    {
        return $this->region;
    }

    /**
     * @param string $region
     */
    public function setRegion(string $region): void
    {
        $this->region = $region;
    }

    /**
     * @param array $reservists
     */
    public function setReservists(array $reservists): void
    {
        $this->reservists = $reservists;
    }

    /**
     * @return array
     */
    public function getReservists(): array
    {
        return $this->reservists;
    }
}
```

Reservist.php:

```
<?php

class Reservist
{
    private int $id;
    private string $firstName;
    private string $lastName;
    private DateTime $birthDate;

    private array $militaryUnit;

    /**
     * @return string
     */
```

```php
    public function getLastName(): string
    {
        return $this->lastName;
    }

    /**
     * @param string $lastName
     */
    public function setLastName(string $lastName): void
    {
        $this->lastName = $lastName;
    }

    /**
     * @return string
     */
    public function getFirstName(): string
    {
        return $this->firstName;
    }

    /**
     * @param string $firstName
     */
    public function setFirstName(string $firstName): void
    {
        $this->firstName = $firstName;
    }

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }

    /**
     * @param int $id
     */
    public function setId(int $id): void
    {
        $this->id = $id;
    }

    /**
     * @return DateTime
     */
    public function getBirthDate(): DateTime
    {
```

```
        return $this->birthDate;
    }

    /**
     * @param DateTime $birthDate
     */
    public function setBirthDate(DateTime $birthDate): void
    {
        $this->birthDate = $birthDate;
    }

    /**
     * @return array
     */
    public function getMilitaryUnit(): array
    {
        return $this->militaryUnit;
    }

    /**
     * @param array $militaryUnit
     */
    public function setMilitaryUnit(array $militaryUnit): void
    {
        $this->militaryUnit = $militaryUnit;
    }
}
```

Репозитории:

```
<?php
include("Interfaces/IMilitaryReservistRepository.php");

class MilitaryReservistRepository implements IMilitaryReservistRepository
{
    private PDO $connection;

    public function __construct(PDO $connection)
    {
        $this->connection = $connection;
    }

    public function getAll(): array
    {
        $query = 'SELECT * FROM "MilitaryReservist"';
        $stmt = $this->connection->prepare($query);
        $stmt->execute();
        return $stmt->fetchAll();
    }
```

```
    public function create(MilitaryReservist $entity)
    {
        $query = 'INSERT INTO "MilitaryReservist"
                    ("ReservistId", "MilitaryUnitId")
                VALUES
                    (?, ?)';

        $stmt = $this->connection->prepare($query);
        $stmt->execute([$entity->getReservistId(), $entity->getMilitaryUnitId()]);
        return $stmt->fetch();
    }

    public function getByIdOrNull($id): MilitaryReservist
    {
        // TODO: Implement getByIdOrNull() method.
    }

    public function update(MilitaryReservist $entity)
    {
        // TODO: Implement update() method.
    }

    public function delete($id)
    {
        // TODO: Implement delete() method.
    }
}
<?php
include("Interfaces/IMilitaryUnitRepository.php");

class MilitaryUnitRepository implements IMilitaryUnitRepository
{
    private PDO $connection;

    public function __construct(PDO $connection)
    {
        $this->connection = $connection;
    }

    public function getAll(): array
    {
        $query = 'SELECT * FROM "MilitaryUnit"';
        $stmt = $this->connection->prepare($query);
        $stmt->execute();
        return $stmt->fetchAll();
    }

    public function create(MilitaryUnit $entity)
    {
        $query = 'INSERT INTO "MilitaryUnit"
```

```
                    ("Name", "Number", "Region")
                VALUES
                    (?, ?, ?)';

        $stmt = $this->connection->prepare($query);
        $stmt->execute([$entity->getName(), $entity->getNumber(), $entity->getRegion()]);

        return $stmt->fetch();
    }

    public function getByIdOrNull($id): MilitaryUnit
    {
        // TODO: Implement getByIdOrNull() method.
    }

    public function update(MilitaryUnit $entity)
    {
        // TODO: Implement update() method.
    }

    public function delete($id)
    {
        // TODO: Implement delete() method.
    }
}
<?php
include("Interfaces/IReservistRepository.php");

class ReservistRepository implements IReservistRepository
{
    private PDO $connection;

    public function __construct(PDO $connection)
    {
        $this->connection = $connection;
    }

    public function getAll(): array
    {
        $query = 'SELECT * FROM "Reservist"';
        $stmt = $this->connection->prepare($query);
        $stmt->execute();
        return $stmt->fetchAll();
    }

    public function create(Reservist $entity)
    {
        $query = 'INSERT INTO "Reservist"
                    ("FirstName", "LastName", "BirthDate")
                VALUES
```

```
                    (?, ?, ?)';

        $stmt = $this->connection->prepare($query);
        $stmt->execute([$entity->getFirstName(), $entity->getLastName(), $entity-
>getBirthDate()->format('Y-m-d H:i:s')]);
        return $stmt->fetch();
    }

    public function getByIdOrNull($id): Reservist
    {
        // TODO: Implement getByIdOrNull() method.
    }

    public function update(Reservist $entity)
    {
        // TODO: Implement update() method.
    }

    public function delete($id)
    {
        // TODO: Implement delete() method.
    }
}
```

Вывод: Я ознакомился с агрегатами, объектами значений, сущностями и репозиториями. Реализовал парочку из них на практике