Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Отчёт
По лабораторной работе №5
По дисциплине ПрИС

Выполнил
Студент группы ПО-3
3-го курса
Кулинкович И. Т.

Проверил
Лаврущик А. И.

Брест 2021

Лабораторная работа №5

Доменная модель

ВАРИАНТ 17

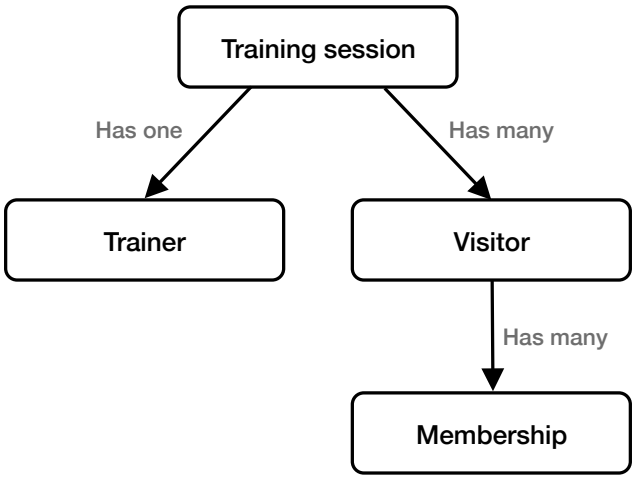
Цель работы. Познакомиться с тактическими шаблонами предметно-ориентированного проектирования.

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).

Предметная область. Управление фитнес-центром.

Ход работы

Применим сущности из предыдущих лабораторных работ в соответствии с предметной областью. Предыдущий подход использования ключей с идентификаторами удаляем и переделываем с использованием механизма агрегатов. Добавляем еще одну модель Membership и привязываем ее к посетителям. Получаем следующую структуру моделей с агрегатами:



Программная реализация

Membership.php

```php
<?php

namespace App\Entity;

use App\Repository\MembershipRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=MembershipRepository::class)
 */
class Membership
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
```

```
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="integer")
     */
    private $duration;

    /**
     * @ORM\Column(type="integer")
     */
    private $cost;

    /**
     * @ORM\OneToMany(targetEntity=Visitor::class, mappedBy="membership")
     */
    private $member;

    public function __construct()
    {
        $this->member = new ArrayCollection();
    }

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

    public function getDuration(): ?int
    {
        return $this->duration;
    }

    public function setDuration(int $duration): self
    {
        $this->duration = $duration;

        return $this;
    }

    public function getCost(): ?int
    {
        return $this->cost;
    }

    public function setCost(int $cost): self
    {
        $this->cost = $cost;

        return $this;
    }

    /**
     * @return Collection|Visitor[]
     */
    public function getMember(): Collection
    {
        return $this->member;
    }
```

```
    public function addMember(Visitor $member): self
    {
        if (!$this->member->contains($member)) {
            $this->member[] = $member;
            $member->setMembership($this);
        }

        return $this;
    }

    public function removeMember(Visitor $member): self
    {
        if ($this->member->removeElement($member)) {
            // set the owning side to null (unless already changed)
            if ($member->getMembership() === $this) {
                $member->setMembership(null);
            }
        }

        return $this;
    }
}
```

Trainer.php

```
<?php

namespace App\Entity;

use App\Repository\TrainerRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=TrainerRepository::class)
 */
class Trainer
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=100)
     */
    private $name;

    /**
     * @ORM\Column(type="integer")
     */
    private $years_experience;

    /**
     * @ORM\OneToMany(targetEntity=TrainingSession::class,
mappedBy="trainer")
     */
```

```
    private $trainingSessions;

    public function __construct()
    {
        $this->trainingSessions = new ArrayCollection();
    }

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

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

    public function setName(string $name): self
    {
        $this->name = $name;

        return $this;
    }

    public function getYearsExperience(): ?int
    {
        return $this->years_experience;
    }

    public function setYearsExperience(int $years_experience): self
    {
        $this->years_experience = $years_experience;

        return $this;
    }

    /**
     * @return Collection|TrainingSession[]
     */
    public function getTrainingSessions(): Collection
    {
        return $this->trainingSessions;
    }

    public function addTrainingSession(TrainingSession $trainingSession):
self
    {
        if (!$this->trainingSessions->contains($trainingSession)) {
            $this->trainingSessions[] = $trainingSession;
            $trainingSession->setTrainer($this);
        }

        return $this;
    }

    public function removeTrainingSession(TrainingSession $trainingSession):
self
    {
        if ($this->trainingSessions->removeElement($trainingSession)) {
            // set the owning side to null (unless already changed)
```

```
            if ($trainingSession->getTrainer() === $this) {
                $trainingSession->setTrainer(null);
            }
        }

        return $this;
    }
}
```

TrainingSession.php

```
<?php

namespace App\Entity;

use App\Repository\TrainingSessionRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=TrainingSessionRepository::class)
 */
class TrainingSession
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="date")
     */
    private $date;

    /**
     * @ORM\ManyToOne(targetEntity=Trainer::class,
inversedBy="trainingSessions")
     */
    private $trainer;

    /**
     * @ORM\ManyToMany(targetEntity=Visitor::class,
inversedBy="trainingSessions")
     */
    private $visitor;

    public function __construct()
    {
        $this->visitor = new ArrayCollection();
    }

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

    public function getDate(): ?\DateTimeInterface
    {
```

```
        return $this->date;
    }

    public function setDate(\DateTimeInterface $date): self
    {
        $this->date = $date;

        return $this;
    }

    public function getTrainer(): ?Trainer
    {
        return $this->trainer;
    }

    public function setTrainer(?Trainer $trainer): self
    {
        $this->trainer = $trainer;

        return $this;
    }

    /**
     * @return Collection|Visitor[]
     */
    public function getVisitor(): Collection
    {
        return $this->visitor;
    }

    public function addVisitor(Visitor $visitor): self
    {
        if (!$this->visitor->contains($visitor)) {
            $this->visitor[] = $visitor;
        }

        return $this;
    }

    public function removeVisitor(Visitor $visitor): self
    {
        $this->visitor->removeElement($visitor);

        return $this;
    }
}
```

Visitor.php

```
<?php

namespace App\Entity;

use App\Repository\VisitorRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=VisitorRepository::class)
 */
```

```php
class Visitor
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=100)
     */
    private $name;

    /**
     * @ORM\Column(type="integer")
     */
    private $weight;

    /**
     * @ORM\Column(type="integer")
     */
    private $height;

    /**
     * @ORM\ManyToMany(targetEntity=TrainingSession::class,
mappedBy="visitor")
     */
    private $trainingSessions;

    /**
     * @ORM\ManyToOne(targetEntity=Membership::class, inversedBy="member")
     */
    private $membership;

    public function __construct()
    {
        $this->trainingSessions = new ArrayCollection();
    }

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

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

    public function setName(string $name): self
    {
        $this->name = $name;

        return $this;
    }

    public function getWeight(): ?int
    {
        return $this->weight;
```

```
    }

    public function setWeight(int $weight): self
    {
        $this->weight = $weight;

        return $this;
    }

    public function getHeight(): ?int
    {
        return $this->height;
    }

    public function setHeight(int $height): self
    {
        $this->height = $height;

        return $this;
    }

    /**
     * @return Collection|TrainingSession[]
     */
    public function getTrainingSessions(): Collection
    {
        return $this->trainingSessions;
    }

    public function addTrainingSession(TrainingSession $trainingSession):
self
    {
        if (!$this->trainingSessions->contains($trainingSession)) {
            $this->trainingSessions[] = $trainingSession;
            $trainingSession->addVisitor($this);
        }

        return $this;
    }

    public function removeTrainingSession(TrainingSession $trainingSession):
self
    {
        if ($this->trainingSessions->removeElement($trainingSession)) {
            $trainingSession->removeVisitor($this);
        }

        return $this;
    }

    public function getMembership(): ?Membership
    {
        return $this->membership;
    }

    public function setMembership(?Membership $membership): self
    {
        $this->membership = $membership;

        return $this;
```

```
    }
}
```

Репозитории работают идентично тем, что реализованы в прошлых лабораторных.

Тестирование агрегатных сущностей

MembershipTest.php

```php
<?php namespace App\Tests;

use PHPUnit\Framework\TestCase;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class MembershipTest extends KernelTestCase
{
    private $entityManager;

    protected function setUp(): void
    {
        $kernel = self::bootKernel();

        $this->entityManager = $kernel->getContainer()
            ->get('doctrine')
            ->getManager();
    }

    public function testInsertAndAssignMember()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\Membership::class);
        $visitor_repository = $this->entityManager
            ->getRepository(\App\Entity\Visitor::class);

        $new_visitor = $visitor_repository->addVisitor("John", 100, 50);

        $membership = $repository->addMembership(100, 60);

        $this->assertEquals(count($membership->getMember()), 0);

        $membership->addMember($new_visitor);

        $this->assertEquals(count($membership->getMember()), 1);
    }

    public function testFindAll()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\Membership::class);

        $our_count = $repository->getMembershipCount();
        $get_all_count = count($repository->findAll());
        $this->assertEquals($our_count, $get_all_count);
    }
}
```

TrainingSessionTest.php

```php
<?php namespace App\Tests;

use PHPUnit\Framework\TestCase;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;
```

```php
class TrainingSessionTest extends KernelTestCase
{
    private $entityManager;

    protected function setUp(): void
    {
        $kernel = self::bootKernel();

        $this->entityManager = $kernel->getContainer()
            ->get('doctrine')
            ->getManager();
    }

    public function testInsert()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\TrainingSession::class);
        $trainer_repository = $this->entityManager
            ->getRepository(\App\Entity\Trainer::class);
        $visitor_repository = $this->entityManager
            ->getRepository(\App\Entity\Visitor::class);

        $new_trainer = $trainer_repository->addTrainer("test trainer", 10);
        $new_visitor = $visitor_repository->addVisitor("test visitor", 65,
180);

        $original_count = $repository->getTrainingSessionCount();

        $repository->addTrainingSession(new \DateTime(), $new_trainer,
$new_visitor);
        $new_count = $repository->getTrainingSessionCount();
        $this->assertEquals($original_count + 1, $new_count);
    }

    public function testFindAll()
    {
        $repository = $this->entityManager
            ->getRepository(\App\Entity\TrainingSession::class);

        $our_count = $repository->getTrainingSessionCount();
        $get_all_count = count($repository->findAll());
        $this->assertEquals($our_count, $get_all_count);
    }
}
```

```
ilyakulinkovich@Ilyas-MacBook-Pro fitness_centre % php bin/phpunit
PHPUnit 8.5.14 by Sebastian Bergmann and contributors.

Testing Project Test Suite
........                                                            8 / 8 (100%)

Time: 1.22 seconds, Memory: 22.00 MB

OK (8 tests, 9 assertions)
```

Вывод

В данной лабораторной работе я познакомился с тактическими шаблонами предметно-ориентированного проектирования.