Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Отчет
По лабораторной работе №5
по дисциплине «ПрИС»

Выполнила
студентка 3 курса
группы ПО-3
Пивчик В.Г.
Проверил
Лаврущик А.И.

Брест 2021

Лабораторная работа №5

Доменная модель

ВАРИАНТ 9

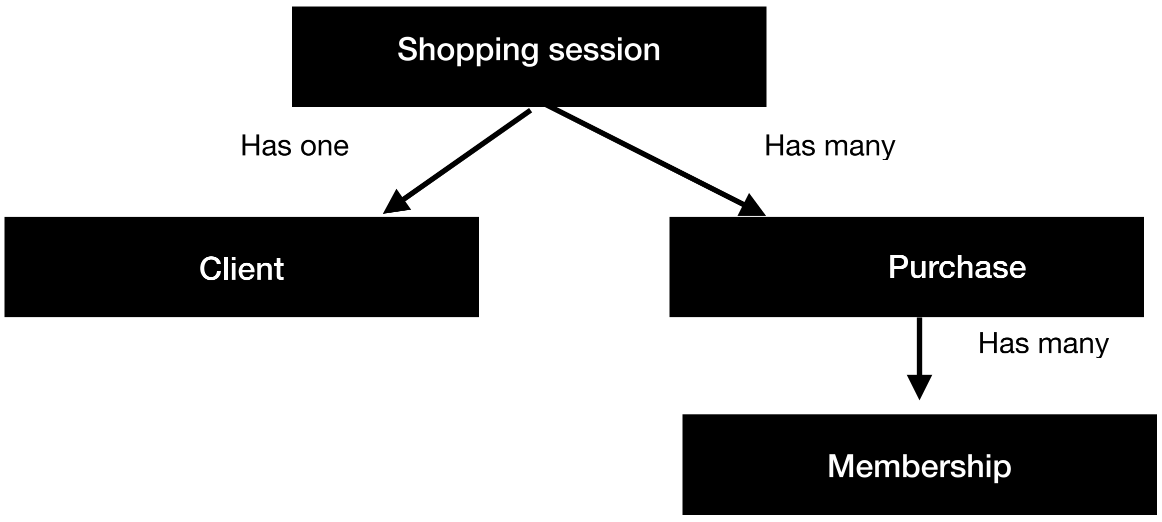
Цель работы. Познакомиться с тактическими шаблонами предметно-ориентированного проектирования.

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 (ЛР4).

Предметная область. Продажа электрооборудования.

Ход работы

Добавляем еще одну модель Membership и привязываем ее к покупкам. Получаем следующую структуру моделей с агрегатами:



Программная реализация

Client.php

```
<?php

namespace App\Entity;

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

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

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

    /**
     * @ORM\Column(type="integer")
     */
    private $number;
    /**
     * @ORM\ManyToMany(targetEntity=ShoppingSession::class, mappedBy="Client")
     */
    private $shoppingSession;

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

    public function __construct()
    {
        $this->shoppingSession = 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 getNumber(): ?int
    {
        return $this->number;
    }

    public function setNumber(int $number): self
    {
        $this->number = $number;

        return $this;
    }

    /**
     * @return Collection|ShoppingSession[]
     */
    public function getShoppingSessions(): Collection
    {
        return $this->shoppingSession;
    }

    public function addShoppingSession(ShoppingSession $shoppingSession): self
    {
        if (!$this->shoppingSession->contains($shoppingSession)) {
            $this->shoppingSession[] = $shoppingSession;
            $shoppingSession->addClient($this);
        }

        return $this;
    }

    public function removeShoppingSession(ShoppingSession $shoppingSession): 
self
    {
        if ($this->shoppingSession->removeElement($shoppingSession)) {
            $shoppingSession->removeClient($this);
        }

        return $this;
    }
}
```

Purchase.php

```
<?php

namespace App\Entity;

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

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

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

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

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

    /**
     * @ORM\OneToMany(targetEntity=ShoppingSession::class, mappedBy="Purchase")
     */
    private $shoppingSession;

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

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

        return $this;
    }

    public function getVersion(): ?int
    {
        return $this->version;
    }

    public function setVersion(int $version): self
    {
        $this->version = $version;

        return $this;
    }

    /**
     * @return Collection|ShoppingSession[]
     */
    public function getShoppingSession(): Collection
    {
        return $this->shoppingSession;
    }

    public function addshoppingSession(ShoppingSession $shoppingSession): self
    {
        if (!$this->shoppingSession->contains($shoppingSession)) {
            $this->shoppingSession[] = $shoppingSession;
            $shoppingSession->setPurchase($this);
        }

        return $this;
    }

    public function removeShoppingSession(ShoppingSession $shoppingSession):
self
    {
        if ($this->shoppingSession->removeElement($shoppingSession)) {
            // set the owning side to null (unless already changed)
            if ($shoppingSession->getPurchase() === $this) {
                $shoppingSession->setPurchase(null);
            }
        }

        return $this;
    }

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

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

        return $this;
    }
}
```

Membership.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=Purchase::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|Purchase[]
     */
    public function getMember(): Collection
    {
        return $this->member;
    }

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

        return $this;
    }

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

        return $this;
    }
}
```

ShoppingSession.php

```
<?php

namespace App\Entity;

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

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

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

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

    /**
     * @ORM\ManyToOne(targetEntity=Cient::class, inversedBy="ShoppingSessions")
     */
    private $client;

    /**
     * @ORM\ManyToMany(targetEntity=Purchase::class,
inversedBy="ShoppingSessions")
     */
    private $purchase;

    public function __construct()
    {
        $this->purchase = 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 getPurchase(): ?Purchase
    {
        return $this->client;
    }

    public function setPurchase(?Purchase $purchase): self
    {
        $this->purchase = $purchase;

        return $this;
    }

    /**
     * @return Collection|Purchase[]
     */
    public function getPurchase(): Collection
    {
        return $this->purchase;
    }
    public function addClient(Client $client): self
    {
        if (!$this->client->contains($client)) {
            $this->client[] = $client;
        }

        return $this;
    }

    public function removeClient(Client $client): self
    {
        $this->client->removeElement($client);

        return $this;
    }
}
```

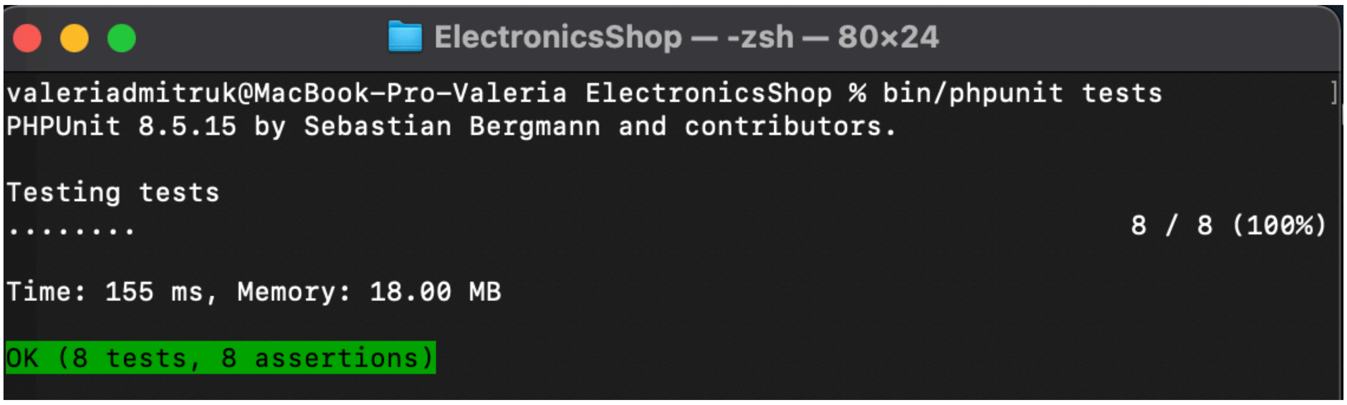
Тестирование агрегатных сущностей

ShoppingSessionTest.php

```
<?php namespace App\Tests;

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

class ShoppingSessionTest 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\ShoppingSession::class);
        $putchase_repository = $this->entityManager-
>getRepository(\App\Entity\Purchase::class);
        $client_repository = $this->entityManager-
>getRepository(\App\Entity\Client::class);
        $new_purchase = $putchase_repository->addPurchase("test purchase", 10,
10);
        $new_client = $client_repository->addClient("test client", 11111);
        $original_count = $repository->getShoppingSessionCount();
        $repository->addShoppingSession(new \DateTime(), $new_purchase,
$new_client);
        $new_count = $repository->getShoppingSessionCount();
        $this->assertEquals($original_count + 1, $new_count);
    }

    public function testFindAll()
    {
        $repository = $this->entityManager-
>getRepository(\App\Entity\ShoppingSession::class);
        $our_count = $repository->getShoppingSessionCount();
        $get_all_count = count($repository->findAll());
        $this->assertEquals($our_count, $get_all_count);
    }
}
```

MembershipTest.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);
        $purchase_repository = $this->entityManager-
>getRepository(\App\Entity\Purchase::class);

        $new_purchase = $purchase_repository->addPurchase("iPhone", 1000, 10);

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

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

        $membership->addMember($new_purchase);

        $this->assertEquals(count($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);
    }
}
```



```
valeriadmitruk@MacBook-Pro-Valeria ElectronicsShop % bin/phpunit tests
PHPUnit 8.5.15 by Sebastian Bergmann and contributors.

Testing tests
........                                                    8 / 8 (100%)

Time: 155 ms, Memory: 18.00 MB

OK (8 tests, 8 assertions)
```

Вывод

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