

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

Отчет по лабораторной работе №5
Дисциплина “ПИС”

Выполнила:
студентка, гр. ПО-3
Гаврилюк Р. И.

Проверил:
Лаврущик А.

Лабораторная работа №5

Доменная модель

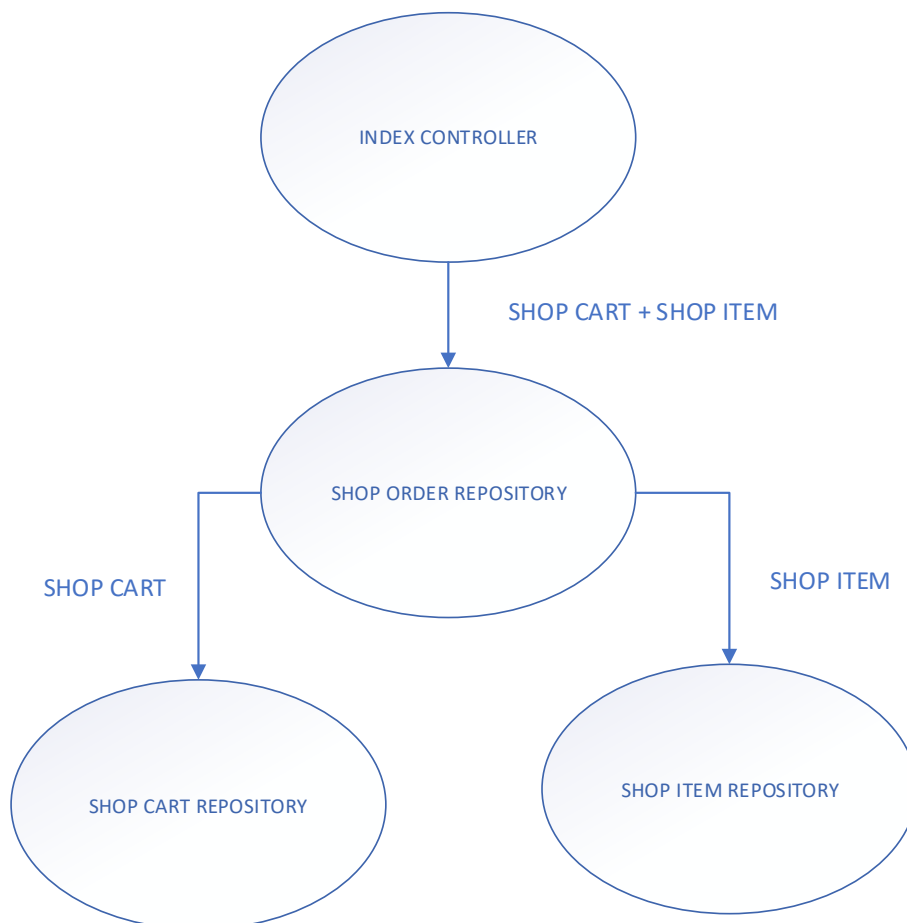
Вариант 21

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.

Предметная область: продажа игрушек.

Применим из предыдущих лабораторных работ в соответствии с предметной областью. Предыдущий подход с использованием ключей с идентификаторами удаляем и переделываем с использованием механизма агрегатов.



Программная реализация:

ShopCart.php

```
<?php

namespace App\Entity;

use App\Repository\ShopCartRepository;
use Doctrine\ORM\Mapping as ORM;

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

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $sessionId;

    /**
     * @ORM\ManyToOne(targetEntity=ShopItems::class, inversedBy="shopCarts")
     * @ORM\JoinColumn(nullable=false)
     */
    private $shopItem;

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

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

    public function getSessionId(): ?string
    {
        return $this->sessionId;
    }

    public function setSessionId(string $sessionId): self
    {
        $this->sessionId = $sessionId;

        return $this;
    }
```

```

    }

    public function getShopItem(): ?ShopItems
    {
        return $this->shopItem;
    }

    public function setShopItem(?ShopItems $shopItem): self
    {
        $this->shopItem = $shopItem;

        return $this;
    }

    public function getCount(): ?int
    {
        return $this->count;
    }

    public function setCount(int $count): self
    {
        $this->count = $count;

        return $this;
    }
}

```

ShopItems.php

```

<?php

namespace App\Entity;

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

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

    /**
     * @ORM\Column(type="string", length=255)
     */

```

```
private $price;

/**
 * @ORM\Column(type="string", length=255)
 */
private $title;

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

/**
 * @ORM\OneToMany(targetEntity=ShopCart::class, mappedBy="shopItem", orphanRemoval=true)
 */
private $shopCarts;

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

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

public function getPrice(): ?string
{
    return $this->price;
}

public function setPrice(string $price): self
{
    $this->price = $price;

    return $this;
}

public function getTitle(): ?string
{
    return $this->title;
}

public function setTitle(string $title): self
{
    $this->title = $title;

    return $this;
}

public function getDescription(): ?string
{

```

```

        return $this->description;
    }

    public function setDescription(string $description): self
    {
        $this->description = $description;

        return $this;
    }

    /**
     * @return Collection|ShopCart[]
     */
    public function getShopCarts(): Collection
    {
        return $this->shopCarts;
    }

    public function addShopCart(ShopCart $shopCart): self
    {
        if (!$this->shopCarts->contains($shopCart)) {
            $this->shopCarts[] = $shopCart;
            $shopCart->setShopItem($this);
        }

        return $this;
    }

    public function removeShopCart(ShopCart $shopCart): self
    {
        if ($this->shopCarts->contains($shopCart)) {
            $this->shopCarts->removeElement($shopCart);
            // set the owning side to null (unless already changed)
            if ($shopCart->getShopItem() === $this) {
                $shopCart->setShopItem(null);
            }
        }

        return $this;
    }
}

```

ShopOrder.php

```

<?php

namespace App\Entity;

use App\Repository\ShopOrderRepository;
use Doctrine\ORM\Mapping as ORM;

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

```

```

*/
class ShopOrder
{
    public const STATUS_NEW_ORDER = 1;

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

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $sessionId;

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

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $userName;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $userEmail;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $userPhone;

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

    public function getSessionId(): ?string
    {
        return $this->sessionId;
    }

    public function setSessionId(string $sessionId): self
    {
        $this->sessionId = $sessionId;

        return $this;
    }
}

```

```

public function getStatus(): ?int
{
    return $this->status;
}

public function setStatus(int $status): self
{
    $this->status = $status;

    return $this;
}

public function getUserName(): ?string
{
    return $this->userName;
}

public function setUserName(string $userName): self
{
    $this->userName = $userName;

    return $this;
}

public function getUserEmail(): ?string
{
    return $this->userEmail;
}

public function setUserEmail(string $userEmail): self
{
    $this->userEmail = $userEmail;

    return $this;
}

public function getUserPhone(): ?string
{
    return $this->userPhone;
}

public function setUserPhone(string $userPhone): self
{
    $this->userPhone = $userPhone;

    return $this;
}
}

```

IndexController.php

```
<?php
```

```

namespace App\Controller;

use App\Entity\ShopCart;
use App\Entity\ShopItems;
use App\Entity\ShopOrder;
use App\Form\OrderFormType;
use App\Repository\ShopCartRepository;
use App\Repository\ShopItemsRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Session\SessionInterface;
use Symfony\Component\Routing\Annotation\Route;

class IndexController extends AbstractController
{
    private SessionInterface $session;

    /**
     * IndexController constructor.
     * @param SessionInterface $session
     */
    public function __construct(SessionInterface $session)
    {
        $this->session = $session;
        $this->session->start();
    }

    /**
     * @Route("/", name="index")
     */
    public function index(): Response
    {
        return $this->render(
            'index/index.html.twig',
            [
                'title' => 'Главная страница',
            ]
        );
    }

    /**
     * @Route("/shop/list", name="shopList")
     * @param ShopItemsRepository $itemsRepository
     * @return Response
     */
    public function shopList(ShopItemsRepository $itemsRepository): Response
    {
        $items = $itemsRepository->findAll();

        return $this->render(
            'index/shopList.html.twig',

```

```

        [
            'title' => 'SHOP LIST',
            'items' => $items,
        ]
    );
}

/**
 * @Route("/shop/cart/add/{id<\d+>}", name="shopCartAdd")
 *
 * @param ShopItems $shopItems
 * @param EntityManagerInterface $em
 * @return Response
 */
public function shopCartAdd(ShopItems $shopItems, EntityManagerInterface $em): Response
{
    $sessionId = $this->session->getId();

    $shopCart = (new ShopCart())
        ->setShopItem($shopItems)
        ->setCount(1)
        ->setSessionId($sessionId);

    $em->persist($shopCart);
    $em->flush();

    return $this->redirectToRoute('shopItem', ['id' => $shopItems->getId()]);
}

/**
 * @Route("/shop/item/{id<\d+>}", name="shopItem")
 *
 * @param ShopItems $shopItems
 * @return Response
 */
public function shopItem(ShopItems $shopItems): Response
{
    return $this->render(
        'index/shopItem.html.twig',
        [
            'title' => $shopItems->getTitle(),
            'description' => $shopItems->getDescription(),
            'price' => $shopItems->getPrice(),
            'id' => $shopItems->getId(),
        ]
    );
}

/**
 * @Route("/shop/cart", name="shopCart")
 *
 * @param ShopCartRepository $cartRepository
 * @return Response

```

```

*/
public function shopCart(ShopCartRepository $cartRepository): Response
{
    $session = $this->session->getId();
    $items = $cartRepository->findBy(['sessionId' => $session]);

    return $this->render(
        'index/shopCart.html.twig',
        [
            'title' => 'Корзина',
            'items' => $items,
        ]
    );
}

/**
 * @Route("/shop/order", name="shopOrder")
 *
 * @param Request $request
 * @return Response
 */
public function shopOrder(Request $request, EntityManagerInterface $em): Response
{
    // just setup a fresh $task object (remove the example data)
    $shopOrder = new ShopOrder();

    $form = $this->createForm(OrderFormType::class, $shopOrder);

    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()) {
        $shopOrder = $form->getData();

        if ($shopOrder instanceof ShopOrder) {
            $sessionId = $this->session->getId();
            $shopOrder->setStatus(ShopOrder::STATUS_NEW_ORDER);
            $shopOrder->setSessionId($sessionId);
            $em->persist($shopOrder);
            $em->flush();
            //session_regenerate_id
            $this->session->migrate();
        }

        return $this->redirectToRoute('index');
    }

    return $this->render(
        'index/shopOrder.html.twig',
        [
            'title' => 'Оформление заказа',
            'form' => $form->createView(),
        ]
    );
}

```

```
}  
}
```

Вывод: познакомилась с тактическими шаблонами предметно-ориентированного проектирования.