Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «ПИС»

Тема: “Доменная модель”

Выполнил:

Студент 3 курса

Группы ПО-7

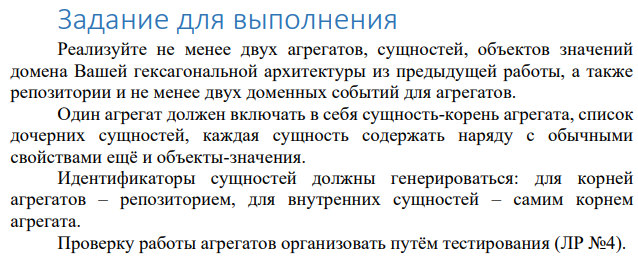
Комиссаров А.Е.

Проверил:

Михняев А.Л.

2023

**Цель работы:** познакомиться с тактическими шаблонами предметноориентированного проектирования.

****

**Ход работы.**

Описание:

**1.Список сущностей**:

- EntityInterface – абстрактная сущность

- User – клиент сайта

- Item – товар.

**2.Список объектов значений:**

- Adress – адрес клиента

- Provider – поставщик товара

**3.Список репозиториев:**

- UserRepository – интерфейс репозитория клиентов

- UserMySQLRepository – реализация для MySql репозитория клиента

- ItemRepository – интерфейс репозитория техники

- ItemMySQLRepository – реализация для MySql репозитория электрооборудования

- PurchaseRepository – интерфейс репозитория событий

- PurchaseMySQLRepository – реализация для MySql репозитория событий

**4.Список доменных событий:**

**-** PurchaseInterface – интерфейс события

- PurchaseDispatcher – диспетчер событий

- Authorization – авторизация клиента в системе

- UserByItem – покупка товара

**Агрегаторы:**

Корень агрегата User

Корень агрегата Item

Объект значений Adress

Объект значений Provider

**Агрегат User**

**Агрегат Item**

**Текст программы:**

    abstract class EntityInterface

    {

        protected $id;

        public function \_\_construct()

        {

            $this->id = uniqid();

        }

        public function getId(): string

        {

            return $this->id;

        }

    }

class Adress extends EntityInterface

{

    private $adress\_id;

    private $city;

    private $street;

    private $home;

    public function \_\_construct(string $city,string $adress\_id, string $street, string $home)

    {

       parent::\_\_construct();

       $this->adress\_id = $adress\_id;

       $this->city = $city;

       $this->street=$street;

       $this->home=$home;

    }

    public function getAdressId(): string

    {       return $this->adress\_id;

    }

    public function getStreet(): string

    {       return $this->street;

    }

    public function getCity(): string

    {       return $this->city;

    }

    public function getHome(): string

    {       return $this->home;

    }

    public function setAdressId(string $city\_id)

    {       $this->city\_id = $city\_id;

    }

    public function setStreet(string $street)

    {       $this->street = $street;

    }

    public function setHome(string $home)

    {       $this->home = $home;

    }

}

class User extends EntityInterface

{

    private $login;

    private $adress;

    private $user\_id;

    private $Purchase= [];

    public function \_\_construct(string $login, string $adress\_id,string $city, string $street, string $home, string $sub\_id)

    {       parent::\_\_construct();

       $this->login = $login;

       $this->adress=new Adress($city,$adress\_id, $street, $home);

       $this->user\_id=$sub\_id;

    }

      public function getLogin(): string

    {       return $this->login;

    }

    public function getAdressId(): string

    {       return $this->adress->getAdressId();

    }

    public function getCity(): string

    {       return $this->adress->getCity();

    }

    public function getStreet(): string

    {       return $this->adress->getStreet();

    }

    public function getHome(): string

    {        return $this->adress->getHome();

    }

    public function getUserId(): string

    {       return $this->user\_id;

    }

    public function setLogin(string $login)

    {        $this->login=$login;

    }

    public function setAdressId(string $adress\_id)

    {        $this->adress->setAdressId($adress\_id);

    }

    public function setStreet(string $street)

    {        $this->adress->setStreet($street);

    }

    public function setHome(string $home)

    {        $this->adress->setHome($home);

    }

    public function setUserId(string $user\_id)

    {        $this->user\_id = $user\_id;

    }

    public function authorize()

    {        $this->recordPurchase(new Authorization($this));

    }

    private function recordPurchase(PurchaseInterface $Purchase)

    {        $this->Purchase[] = $Purchase;

    }

    public function releasePurchase(): array

    {        $temp=$this->Purchase;

        $this->Purchase= [];

        return $temp;

    }

}

class Provider extends EntityInterface

{

    private $provider\_id;

    private $provider\_name;

    public function \_\_construct(string $provider\_id,string $provider\_name)

    {       parent::\_\_construct();

       $this->provider\_id = $provider\_id;

       $this->provider\_name = $provider\_name;

    }

       public function getProviderId(): string

    {       return $this->provider\_id;

    }

    public function getProviderName(): string

    {       return $this->provider\_name;

    }

    public function setProviderName(string $city\_name)

    {       $this->provider\_name = $provider\_name;

    }

}

class Item extends EntityInterface

{

    private $name;

    private $price;

    private $providers = [];

    private $good\_id;

    private $Purchase = [];

    private $inform;

    private $image;

    public function \_\_construct(string $name, string $price, string $good\_id,string $inform,string $image)

    {       parent::\_\_construct();

       $this->name = $name;

       $this->price = $price;

       $this->good\_id=$good\_id;

       $this->inform = $inform;

       $this->image = $image;

    }

       public function getName(): string

    {       return $this->name;

    }

    public function getPrice(): string

    {       return $this->price;

    }

    public function getGoodId(): string

    {       return $this->good\_id;

    }

    public function getInform(): string

    {       return $this->inform;

    }

    public function getImage(): string

    {       return $this->image;

    }

    public function setName(string $name)

    {        $this->name=$name;

    }

    public function setPrice(string $price)

    {        $this->price = $price;

    }

    public function setGoodId(string $good\_id)

    {        $this->good\_id = $good\_id;

    }

    public function setInform(string $inform)

    {        $this->inform = $inform;

    }

    public function setImage(string $image)

    {        $this->image = $image;

    }

    public function AddProvider(string $provider\_id,string $provider\_name)

    {        for($i=0; $i<count($this->providers); $i++)

            if($this->providers[$i]->getProviderId()===$provider\_id)

                return;

        $prov=new Provider($provider\_id,$provider\_name);

        $this->providers[] = $prov;

    }

    public function removeProvider(string $provider\_id)

    {        for($i=0; $i<count($this->providers); $i++)

            if($this->cities[$i]->getProviderId()===$provider\_id)

            {                unset($providers[$i]);

                return;

            }

    }

        public function UserBy(User $User)

    {        $this->recordPurchase(new UserByItem($User, $this));

    }

    private function recordPurchase(PurchaseInterface $Purchase)

    {        $this->Purchase[] = $Purchase;

    }

    public function releasePurchase(): array

    {        $temp=$this->Purchase;

        $this->Purchase= [];

        return $temp;

    }

}

interface UserRepository

    {

        public function getUserByLP(string $login, string $pass);

    }

class UserMySQLRepository implements UserRepository

{

    private $mysqli;

    public function \_\_construct()

    {

        $this->mysqli = new mysqli("localhost", "root", "", "pris");

    }

    public function getUserByLP(string $login, string $pass)

    {

        $stmt = $this->mysqli->prepare("

        SELECT B.id as id,login,city,street,home,A.id as adr FROM pris.users

as B Left outer join pris.addresses as A ON A.id\_user = B.id

where login=?  AND  password = ?");

        mysqli\_stmt\_bind\_param($stmt, "ss", $login, $pass);

        mysqli\_stmt\_execute($stmt);

        $login;$sub\_id; $adress\_id; $street; $home;

        mysqli\_stmt\_bind\_result($stmt,$sub\_id,$login,$city, $street, $home,$adress\_id);

        mysqli\_stmt\_fetch($stmt);

        if($sub\_id!="")

        {

            return new User($login, $adress\_id, $city,$street, $home, $sub\_id);

        }

        else

        {

            return null;

        }

    }

}

interface ItemRepository

{

    public function getItem();

    public function addItem(string $name, string $price,string $inform, string $image);

    public function removeItem(Item $Item);

    public function UserByItem(User $User, Item $Item);

    public function getUserItem(User $User);

    public function getItemById(string $id);

}

class ItemMySQLRepository implements ItemRepository

{

    private $mysqli;

    public function \_\_construct()

    {

        $this->mysqli = new mysqli("localhost", "root", "", "pris");

    }

    public function getItem()

    {

        $Item = [];

        $res = $this->mysqli->query("SELECT \* FROM pris.goods;");

        if($res->num\_rows==0)

            return null;

        while ($row = $res->fetch\_assoc())

        {

            $Item = new Item ($row['name'],$row['price'],$row['id'],$row['inform'],$row['image\_link']);

            $subres=$this->mysqli->query("SELECT \* FROM pris.good\_provider

             INNER JOIN pris.providers as P ON P.provider\_id = id\_provider where id\_good ='".$row['id']."'");

            while ($subrow = $subres->fetch\_assoc())

                $Item->addProvider($subrow['provider\_id'],$subrow['provider\_name']);

            $Item[] = $Item;

        }

        return $Item;

    }

    public function getUserItem(User $User)

    {

        $id\_User = $User->getUserId();

        $Item = [];

        $res = $this->mysqli->query("SELECT \* FROM user\_good where `id\_user`='".$id\_User."'");

        if($res->num\_rows==0)

            return null;

            $res = $this->mysqli->query("SELECT \* FROM user\_good where `id\_user`='".$id\_User."'");

            while ($row1 = mysqli\_fetch\_array($res))

                {

                    $result =  $this->mysqli->query("SELECT \* FROM goods where `id`=$row1[id\_good]");

                    while($row = mysqli\_fetch\_array($result))

                    {

                        $Item = new Item($row['name'],$row['price'],$row['id'],$row['inform'],$row['image\_link']);

                        $subres=$this->mysqli->query("SELECT \* FROM pris.good\_provider

                        INNER JOIN pris.providers as P ON P.provider\_id = id\_provider where id\_good ='".$row['id']."'");

                        while ($subrow = $subres->fetch\_assoc())

                        $Item->addProvider($subrow['provider\_id'],$subrow['provider\_name']);

                        $Item[] = $Item;

                    }

                }

        return $Item;

    }

    public function addItem(string $name, string $price,string $inform, string $image)

    {

        if(!is\_numeric($price))

            return 0;

        $stmt = $this->mysqli->prepare("INSERT INTO `pris`.`goods` (`name`, `price`, `inform`, `image\_link`) VALUES (?, ?, ?, ?);");

        mysqli\_stmt\_bind\_param($stmt, "sdss",$name, $price,$inform,$image);

        mysqli\_stmt\_execute($stmt);

        if(!$stmt)

            return 0;

        return 1;

    }

    public function removeItem(Item $Item)

    {

        $id=$Item->getGoodId();

        $stmt = $this->mysqli->prepare("DELETE FROM `pris`.`goods` WHERE (`id` = ?)");

        mysqli\_stmt\_bind\_param($stmt, "s", $id);

        mysqli\_stmt\_execute($stmt);

        $stmt = $this->mysqli->prepare("DELETE FROM pris.good\_provider where (`id\_good` = ?)");

        mysqli\_stmt\_bind\_param($stmt, "s", $id);

        mysqli\_stmt\_execute($stmt);

        $stmt = $this->mysqli->prepare("DELETE FROM `pris`.`user\_good` WHERE (`id\_good` = ?)");

        mysqli\_stmt\_bind\_param($stmt, "s", $id);

        mysqli\_stmt\_execute($stmt);

    }

    public function UserByItem(User $User, Item $Item)

    {

        $UserId=$User->getUserId();

        $ItemId=$Item->getGoodId();

        $stmt = $this->mysqli->prepare("INSERT INTO `pris`.`user\_good` (`id\_user`, `id\_good`) VALUES (?, ?)");

        mysqli\_stmt\_bind\_param($stmt, "ss", $UserId, $ItemId);

        mysqli\_stmt\_execute($stmt);

        if(!$stmt)

            return 0;

        return 1;

    }

    public function getItemById($id)

    {

       $Item;

        $res = $this->mysqli->query("SELECT \* FROM pris.goods;");

        if($res->num\_rows==0)

            return null;

        while ($row = $res->fetch\_assoc())

        {

            $Item = new Item ($row['name'],$row['price'],$row['id'],$row['inform'],$row['image\_link']);

            if($Item->getGoodId()==$id)

            return $Item;

        }

        $Item=0;

        return $Item;

    }

}

interface PurchaseRepository

    {

        public function writePurchase(string $time, string $message);

    }

class PurchaseRepositoryMySQL implements PurchaseRepository

{

    private $mysqli;

    public function \_\_construct()

    {

        $this->mysqli = new mysqli("localhost", "root", "", "pris");

    }

    public function writePurchase(string $time, string $message)

    {

        $stmt = $this->mysqli->prepare("INSERT INTO pris.Purchase (time, message) VALUES(?,?)");

        mysqli\_stmt\_bind\_param($stmt, "ss", $time, $message);

        mysqli\_stmt\_execute($stmt);

    }

}

interface PurchaseInterface

    {

        public function getOccurredOn();

        public function getMessage();

    }

class Authorization implements PurchaseInterface

    {

        private $User;

        private $occurredOn;

        public function \_\_construct(User $User)

        {

            $this->User = $User;

            $this->occurredOn = new \DateTimeImmutable();

        }

        public function getOccurredOn()

        {

            return $this->occurredOn;

        }

        public function getMessage()

        {

            return "Клиент :".$this->User->getLogin()." авторизовался в системе";

        }

    }

class UserByItem implements PurchaseInterface

    {

        private $User;

        private $occurredOn;

        public function \_\_construct(User $User)

        {

            $this->User = $User;

            $this->occurredOn = new \DateTimeImmutable();

        }

        public function getOccurredOn()

        {

            return $this->occurredOn;

        }

        public function getMessage()

        {

            return "Клиент :".$this->User->getLogin()." авторизовался в системе";

        }

    }

class PurchaseDispatcher

{

    public function dispatch(array $Purchase)

    {

        $PurchaseRepository=new PurchaseRepositoryMySQL();

        for($i=0; $i<count($Purchase); $i++)

        {

            $PurchaseRepository->writePurchase($Purchase[$i]->getOccurredOn()->format('Y-m-d H:i:s'), $Purchase[$i]->getMessage());

        }

    }

}

**Вывод:** я познакомился с тактическими шаблонами предметноориентированного проектирования.