Практическая 27-31

Тема: Работа с кодом на языке программирования PHP. Анализ кода. На основе кода построение: диаграммы последовательности, диаграммы состояния, use-case диаграммы, бизнес-модели в нотации IDEF3 и диаграммы потоков в нотации DFD.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Диаграмма прецедентов или диаграмма вариантов использования (англ. use case diagram) в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Диаграмма состояний -она показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

Диаграмма последовательности используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования! Диаграммы последовательностей обычно содержат **объекты**, которые **взаимодействуют в рамках сценария**, **сообщения**, которыми они обмениваются, и **возвращаемые результаты**, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

В отличие от IDEF0, представляющего моделируемую систему как совокупность видов деятельности, IDEF3 представляет собой технику моделирования деятельности как последовательности событий, а также участвующих в этих событиях объектов. Модели IDEF3 могут использоваться для детализации функциональных блоков IDEF0, они более низкого уровня . IDEF3 удобен для подробного моделирования деятельности отдельных подразделений, сотрудников, описания техпроцессов и т.д.

DFD (data flow diagram) — диаграмма потоков данных, один из основных инструментов структурного анализа и проектирования информационных систем, существовавших еще до широкого распространения UML DFD диаграммы в отличии от других нотаций позволяют визуально показать все процессы с точки зрения данных. Это может быть полезно:

- при разработке информационной системы;
- при интеграции системы;
- при миграции данных и функционала с одной системы на другую;
- в проектах, связанных с Data Management;
- в процессе построения аналитического хранилища, BI-решения.

Диаграмма позволяет визуализировать как движение данных между объектами системы, так и преобразования данных, которые могут применяться на разных шагах процесса.

ХОД РАБОТЫ

Ссылка на где можно онлайн запустить код и редактировать! PHP Sandbox - Execute PHP code online through your browser (onlinephp.io)

Код написанный в процедурном стиле!

```
<?php
$searchByAuthor = 'Marina2';//входящий поиск по автору 1

//проверка входящего запроса поиска 1.1
if ($searchByAuthor === '') {//если строка поиска пустая тогда вывести ошибку
    echo 'Ошибка, вы не ввели автора!';//вывод ошибки на экран 1.3
    return;//завершение выполнения поиска
}

//Начало имитации получения книг из базы данных 1.1.1
$databaseBooks = [];//инициализация массива
```

```
//добавление в массив книг по автору, где ключ - это автор, а значение книга. Связь 1 ко многим
$databaseBooks['Marina1'][] = 'Name Book 1';//Marina1 - автор, Name Book 1 - название книги
$databaseBooks['Marina2'][] = 'Name Book 2';
$databaseBooks['Marina2'][] = 'Name Book 3';
//Конец имитации получения кнги из базы данных

//Начало поиска автора 1.2
$booksByAuthor = [];//результат поиска книг по автора
foreach ($databaseBooks as $author => $book) {//поиск по нашей базе данной
    if ($author === $searchByAuthor) {//если поиск по автору совпадает с автором из базы данных то:
        $booksByAuthor[$author] = $book;
    }
}
//Конец поиска автора

//Начало создания вывода результата поиска 1.2.1
$searchResult = '';//инициализация строки результата поиска
foreach ($booksByAuthor as $author => $book) {//поиск по нашей базе данной
    $searchResult .= $author . ':' . implode(',', $book) . "\n"; //добавить в результат Автора и его книги
}
//Конец создания вывода результата поиска

echo $searchResult;//вывод результата поиска 1.2.1.1
```

Код написанный в стиле ООП!

```
<?php

$searchByAuthor = 'Marina3';

$databaseBook = getDatabaseBook();
$validationService = new ValidationService();
$mapperLibrary = new MapperLibrary();

$customerRequest = new CustomerRequest($searchByAuthor);
if (!$validationService->valid($customerRequest)) {
    echo 'Ошибка, вы не ввели автора!' . PHP_EOL;
    return;
}

$library = $mapperLibrary->getLibrary($databaseBook);
$searchBookService = new SearchBookService($library);
$bookListByAuthor = $searchBookService->searchByAuthor($customerRequest->author);

$bookFormatter = new BookFormatter();

echo $bookFormatter->format($bookListByAuthor);


class SearchBookService
{
    public Library $library;

    /**
     * @param Library $library
     */
    public function __construct(Library $library)
    {
        $this->library = $library;
```

```
    }

    /**
     * @param string $searchByAuthor
     * @return array<int, Book>
     */
    public function searchByAuthor(string $searchByAuthor): array
    {
        $resultSearchBooksByAuthor = [];
        foreach ($this->library->getBooks() as $book)
        {
            if ($book->getAuthor() == $searchByAuthor) {
                $resultSearchBooksByAuthor[] = $book;
            }
        }

        return $resultSearchBooksByAuthor;
    }
}

class BookFormatter
{
    /**
     * @param array<int, Book> $books
     * @return void
     */
    public function format(array $books): string
    {
        $output = '';

        foreach ($books as $book)
        {
            $output .= $book->getAuthor() . ':' . $book->getName() . PHP_EOL;
        }

        return $output;
    }
}

class CustomerRequest
{
    public string $author;

    /**
     * @param string $author
     */
    public function __construct(string $author)
    {
        $this->author = $author;
    }

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

class ValidationService
{
    public function valid(CustomerRequest $customerRequest): bool
    {
        if ($customerRequest->getAuthor() === '') {
            return false;
        }

        return true;
    }
}

class MapperLibrary
{
    private array $books;

    public function getLibrary(array $books): Library
```

```
    {
        $booksObj = [];
        foreach ($books as $book) {
            $booksObj[] = new Book($book['author'], $book['nameBook']);
        }

        return new Library($booksObj);
    }
}

class Library
{
    /** @var array<int, Book> */
    private array $books;

    /**
     * @param Book[] $books
     */
    public function __construct(array $books)
    {
        $this->books = $books;
    }

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

class Book
{
    private string $author;
    private string $name;

    /**
     * @param string $author
     * @param string $name
     */
    public function __construct(string $author, string $name)
    {
        $this->author = $author;
        $this->name = $name;
    }

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

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

function getDatabaseBook(): array
{
    $databaseBook = [];
    $databaseBook[] = ['author' => 'Marina1', 'nameBook' => 'Name Book 1'];
    $databaseBook[] = ['author' => 'Marina2', 'nameBook' => 'Name Book 2'];
    $databaseBook[] = ['author' => 'Marina3', 'nameBook' => 'Name Book 3'];
    $databaseBook[] = ['author' => 'Marina3', 'nameBook' => 'Name Book 3_2'];
    $databaseBook[] = ['author' => 'Marina3', 'nameBook' => 'Name Book 3_3'];
    $databaseBook[] = ['author' => 'Marina4', 'nameBook' => 'Name Book 4'];
    $databaseBook[] = ['author' => 'Marina5', 'nameBook' => 'Name Book 5'];
    $databaseBook[] = ['author' => 'Marina6', 'nameBook' => 'Name Book 6'];

    return $databaseBook;
```

}

Задание1

1. выберите один из вариантов предложенного кода, идея кода одинаковая, только написана в разных стилях!
2. Откройте онлайн редактор, вставьте код, запустите его и проверите результат.
3. Сделайте анализ кода и по коду напишите техническое задание на разработку, чтобы оно соответствовало написанному коду.
4. Результат задания должно быть написанное ТЗ

Задание2

По коду и ТЗ составьте диаграмму последовательности.

Задание3

По коду и ТЗ составьте диаграмму состояния

Задание4

По коду и ТЗ составьте use-case диаграмму

Задание5

По коду и ТЗ составьте бизнес-модель в нотации IDEF3

Задание6

По коду и ТЗ составьте диаграмму потоков в нотации DFD.

P.S. для тех кто не смог составить самостоятельно, но пролистал методичку до конца)!
ТЗ:
есть два сценария:
1. Основной
2. Альтернативный
1.Основной: Клиент вводит ФИО автора и нажимает кнопку поиска. После чего система проверяет параметры ввода пользователя, если валидация пройдена- система ищет Автора по каталогам, когда поиск окончен - система выдает результат
2. Если клиент не ввел имя автора, но нажал на кнопку поиска, то система выводит сообщение об ошибке.

Контрольные вопросы:

1. Для чего нужна диаграмма последовательности?
2. Для чего нужна диаграмма состояний?
3. Для чего нужна диаграмма use-case?
4. Что показывает нотация бизнес-модели IDEF3?
5. Зачем нужна нотация DFD?
6. Чем отличается код ООП и код процедурный?

Результатом каждого задания должна быть схема, кроме задания 1.

Содержание отчета:

1. Тема, цель практической работы
2. Поэтапное описание выполнения практической работы
3. Скриншоты или результат практической

Краткие ответы на контрольные вопросы