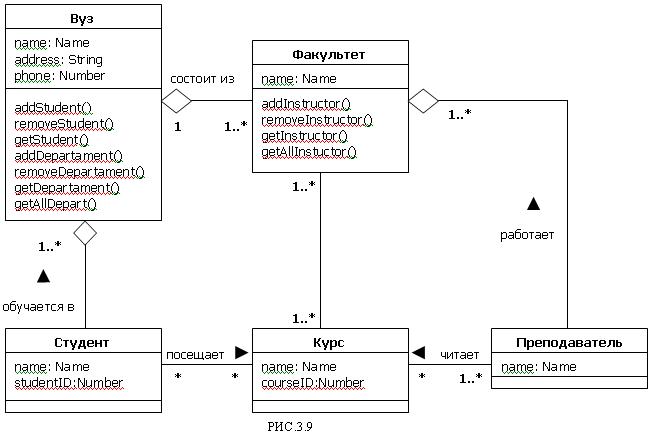
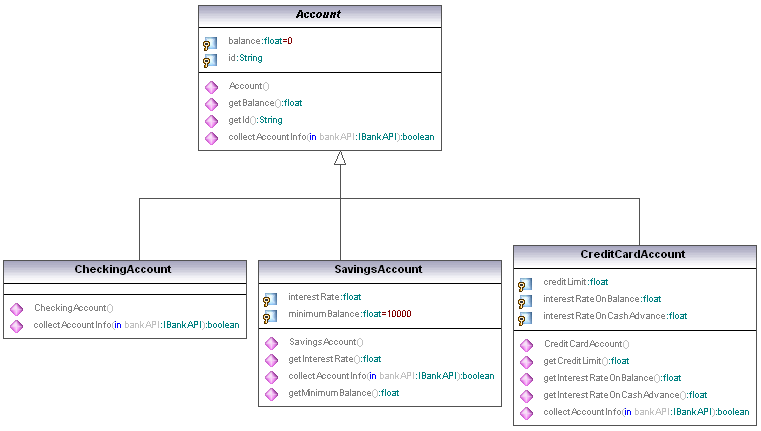
***Практическая работа № 12***

**Создание UML диаграмм классов**

1. Создайте диаграмму классов для своей задачи, см примеры и пояснения ниже.

Пример 1

Пример 2

1. Создайте все эти виды диаграмм еще и для любой задачи из списка практической работы № 2

**Диаграммы классов**

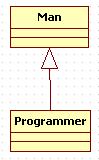
**Диаграмма классов** является типом **диаграммы статической структуры**. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, и также взаимосвязи этих классов. И так, начнем с примеров:

**Генерализация (обобщение)**

Генерализация показывает, что один из двух связанных классов (*подтип*), является более частной формой другого (*супертип*), который называется **обобщением** первого.

Графически генерализация представляется **линией с пустым треугольником у супертипа**.

Диаграмма:



На PHP это будет выглядеть, следующим образом:

<?php

class Man

{

}

class Programmer extends Man

{

}

?>

И введем пару определений:

**Связь** – простая взаимосвязь между объектами. Она представляется **линией соединяющей два или более объектных блока**. Она встречается на диаграммах классов или объектов. Связь – частный случай ассоциации.

**Ассоциация** – это семейство связей двух и более классов. Всего существует пять типов ассоциации. Но наиболее распространены два: **двунаправленная** и **однонаправленная**  
ассоциации.

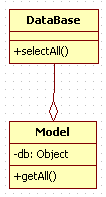
## Агрегация

Агрегация – **«has a»** (быть частью) случай ассоциации. Агрегация применяется когда один класс должен быть контейнером других классов. Причем время существования содержащихся классов никак не зависит от времени существования класса контейнера.

Т.е. в нашем примере в качестве контейнера выступает объект класса **Model**. И если в ходе выполнения программы он будет уничтожен, это никак не повлияет на объект класса **DataBase**.

Графически агрегация представляется **пустым ромбиком** на блоке класса и **линией, идущей от этого ромбика к содержащемуся классу**.

Диаграмма:



Код:

<?php

class DataBase

{

public function selectAll()

{

}

}

class Model

{

private $db;

public function \_\_constructor($db)

{

$this->db = $db;

}

public function getAll()

{

return $this->db->selectAll();

}

}

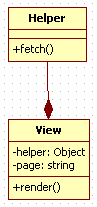
?>

## Композиция

Еще один **«has a»** случай ассоциации, но более строгий. В отличии от агрегации, композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет уничтожено также.

Графически представляется **как и агрегация, но с закрашенным ромбиком**.

Диаграмма:



Код на PHP:

<?php

class Helper

{

public function fetch()

{

//some code

}

}

class View

{

private $helper;

private $page;

public function \_\_constructor()

{

$this->helper = new Helper();

}

public function render()

{

$this->page = $this->helper->fetch();

}

}

?>

## Различие между агрегацией и композицией

Различие между этими двумя видами ассоциации состоит в том, что **композиция** может быть частью одного и только одного целого, в то время как **агрегация** может быть частью нескольких объектов.

## Направленная ассоциация (Message / **Directed Association**)

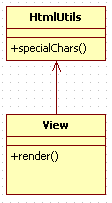
Сообщение используется, когда один класс “общается” с другим при помощи инстанцирования.

**Инстанцирование** (англ. instantiation) — создание экземпляра класса. В отличие от слова «создание» применяется не к объекту, а к классу. То есть говорят создать экземпляр класса или инстанцировать класс. Порождающие паттерны используют полиморфное инстанцирование.

**Экземпляр класса** (англ. instance) — это описание конкретного объекта в памяти. Класс описывает свойства и методы, которые будет доступны у объекта, построенного по описанию, заложенному в класс. Экземпляры используют для представления конкретных сущностей реального мира.

Графически представляется **в виде стрелочки направленой к “вызываемому” классу**.

Диаграмма:



Пример на PHP:

<?php

class HtmlUtils

{

public static function specialChars($str)

{

//Здесь экранируем строку $str, заменяя все нужные символы

return $str;

}

}

class View

{

private $page;

public function render()

{

//.......

$this->page = HtmlUtils::specialChars($text);

//.......

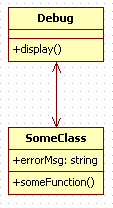
}

}

?>

Так же направленная ассоциация может быть двухсторонней, например:

Диаграмма:



Код:

<?php

class Debug

{

public static function display()

{

echo $this->errorMsg;

}

}

class View

{

public $errorMsg = 'Message';

public function someFunction()

{

//Тело функции и если возникает ошибка то $error = 1

if ($error) Debug::display();

}

}

?>

Вот, собственно и все.