Лабораторная работа №8
Объектно-ориентированное программирование на PHP

Классы и объекты в PHP

Класс - это базовое понятие в объектно-ориентированном программировании (ООП). Классы образуют синтаксическую базу ООП. Их можно рассматривать как своего рода "контейнеры" для логически связанных данных и функций (обычно называемых методами). Если сказать проще, то класс - это своеобразный тип данных.

Экземпляр класса - это объект. Объект - это совокупность данных и функций (методов) для их обработки. Свойства и методы называются членами класса. Вообще, объектом является все то, что поддерживает инкапсуляцию.

Если класс можно рассматривать как тип данных, то объект — как переменную (по аналогии). Скрипт может одновременно работать с несколькими объектами одного класса, как с несколькими переменными.

Внутри объекта данные и код (члены класса) могут быть либо открыты, либо нет. Открытые данные и члены класса являются доступными для других частей программы, которые не являются частью объекта. А вот закрытые данные и члены класса доступны только внутри этого объекта.

Описание классов в PHP начинаются служебным словом class:

class Имя_класса {
//описание членов класса - свойств и методов для их обработки
}

Для объявления объекта необходимо использовать оператор new:

Объект = new Имя_класса;

Данные описываются с помощью служебного слова var. Метод описывается так же, как и обыкновенная пользовательская функция. Методу также можно передавать параметры.

Подведем промежуточные итоги: объявление класса должно начинаться с ключевого слова class (подобно тому, как объявление функции начинается с ключевого слова function). Каждому объявлению свойства, содержащегося в классе, должно предшествовать ключевое слово var. Свойства могут относиться к любому типу данных, поддерживаемых в PHP, их можно рассматривать как переменные с небольшими различиями. После объявлений свойств следуют объявления методов, очень похожие на типичные объявления пользовательских функций.

По общепринятым правилам имена классов ООП начинаются с прописной буквы, а все слова в именах методов, кроме первого, начинаются с прописных букв (первое слово начинается со строчной буквы). Разумеется, вы можете использовать любые обозначения, которые сочтете удобными; главное — выберите стандарт и придерживайтесь его.

Пример класса на PHP:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;
var $addr;
// методы:
 function Name() {
 echo "<h3>John</h3>";
 }
}
// Создаем объект класса Coor:
$object = new Coor;
?>
```

Доступ к класам и объектам в PHP

Мы рассмотрели, каким образом описываются классы и создаются объекты. Теперь нам необходимо получить доступ к членам класса, для этого в PHP предназначен оператор ->. Приведем пример:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;
// методы:
 function Getname() {
 echo "<h3>John</h3>";
 }
}
// Создаем объект класса Coor:
$object = new Coor;
// Получаем доступ к членам класса:
$object->name = "Alex";
echo $object->name;
// Выводит 'Alex'
// А теперь получим доступ к методу класса (фактически, к функции внутри класса):
$object->Getname();
// Выводит 'John' заглавными буквами
?>
```

Чтобы получить доступ к членам класса внутри класса, необходимо использовать указатель $this, которы всегда относится к текущему объекту. Модифицированный метод Getname():

```
function Getname() {
echo $this->name;
}
```

Таким же образом, можно написать метод Setname():

```
function Setname($name) {
$this->name = $name;
}
```

Теперь для изменения имени можно использовать метод Setname():

```
$object->Setname("Peter");
$object->Getname();
```

А вот и полный листинг кода:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;
// методы:
 function Getname() {
 echo $this->name;
 }
function Setname($name) {
$this->name = $name;
 }
}
// Создаем объект класса Coor:
$object = new Coor;
// Теперь для изменения имени используем метод Setname():
$object->Setname("Nick");
// А для доступа, как и прежде, Getname():
$object->Getname();
// Сценарий выводит 'Nick'
?>
```

Указатель $this можно также использовать для доступа к методам, а не только для доступа к данным:

```
function Setname($name) {
$this->name = $name;
$this->Getname();
}
```

Конструкторы

Довольно часто при создании объекта требуется задать значения некоторых свойств. К счастью, разработчики технологии ООП учли это обстоятельство и реализовали его в концепции конструкторов. Конструктор представляет собой метод, который задает значения некоторых свойств (а также может вызывать другие методы). Конструкторы вызываются автоматически при создании новых объектов. Чтобы это стало возможным, имя метода-конструктора должно совпадать с именем класса, в котором он содержится. Пример конструктора:

```
<?
class Webpage {
var $bgcolor;
 function Webpage($color) {
 $this->bgcolor = $color;
 }
}
// Вызвать конструктор класса Webpage
$page = new Webpage("brown");
?>
```

Раньше создание объекта и инициализация свойств выполнялись раздельно. Конструкторы позволяют выполнить эти действия за один этап.

Интересная подробность: в зависимости от количества передаваемых параметров могут вызываться разные конструкторы. В рассмотренном примере объекты класса Webpage могут создаваться двумя способами. Во-первых, вы можете вызвать конструктор, который просто создает объект, но не инициализирует его свойства:

```
$page = new Webpage;
```

Во-вторых, объект можно создать при помощи конструктора, определенного в классе, — в этом случае вы создаете объект класса Webpage и присваиваете значение его свойству bgcolor:

```
$page = new Webpage("brown");
```

Деструкторы

В PHP отсутствует непосредственная поддержка деструкторов. Тем не менее, вы можете легко имитировать работу деструктора, вызывая функцию PHP unset(). Эта функция уничтожает содержимое переменной и возвращает занимаемые ею ресурсы системе. С объектами unset() работает так же, как и с переменными. Допустим, вы работаете с объектом $Webpage. После завершения работы с этим конкретным объектом вызывается функция:

```
unset($Webpage);
```

Эта команда удаляет из памяти все содержимое $Webpage. Действуя в духе инкапсуляции, можно поместить вызов unset() в метод с именем destroy() и затем вызвать его:

```
$Website->destroy();
```

Необходимость в вызове деструкторов возникает лишь при работе с объектами, использующими большой объем ресурсов, поскольку все переменные и объекты автоматически уничтожаются по завершении сценария.

Инициализация объектов

Иногда возникает необходимость выполнить инициализацию объекта - присвоить его свойствам первоначальные значения. Предположим, имя класса Coor и он содержит два свойства:имя человека и город его проживания. Можно написать метод (функцию), который будет выполнять инициализацию объекта, например Init():

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;
var $city;
// Инициализирующий метод:
 function Init($name) {
 $this->name = $name;
 $this->city = "London";
 }
}
// Создаем объект класса Coor:
$object = new Coor;
// Для инициализации объекта сразу вызываем метод:
$object->Init();
?>
```

Главное не забыть вызвать функцию сразу после создания объекта, либо вызвать какой-нибудь метод между созданием (оператор new) объекта и его инициализацией (вызовом Init).

Для того, чтобы PHP знал, что определенный метод нужно вызывать автоматически при создании объекта, ему нужно дать имя такое же, как и у класса (Coor):

```
function Coor ($name)
$this->name = $name;
$this->city = "London";
}
```

Метод, инициализирующий объект, называется конструктором. Однако, PHP не имеет деструкторов, поскольку ресурсы освобождаются автоматически при завершении работы скриптов.

Обращение к элементам классов

Обращение к элементам классов осуществляется с помощью оператора :: "двойное двоеточие". Используя "двойное двоеточие", можно обращаться к методам классов.

При обращении к методам классов, программист должен использовать имена этих классов.

```
<?php
class A {
function example() {
echo "Это первоначальная функция A::example().<br>";
}
}
class B extends A {
function example() {
echo "Это переопределенная функция B::example().<br>";
A::example();
}
}
// Не нужно создавать объект класса A.
// Выводит следующее:
// Это первоначальная функция A::example().
A::example();
// Создаем объект класса B.
$b = new B;
// Выводит следующее:
//   Это переопределенная функция B::example().
//   Это первоначальная функция A::example().
$b->example();
?>
```

Наследование классов в PHP

Наследование - это не просто создание точной копии класса, а расширение уже существующего класса, чтобы потомок мог выполнять какие-нибудь новые, характерные только ему функции.

Итак, пусть у нас есть некоторый класс A с определёнными свойствами и методами. Но то, что этот класс делает, нас не совсем устраивает — например, пусть он выполняет большинство функций, по сути нам необходимых, но не реализует некоторых других. Зададимся целью создать новый класс B, как бы "расширяющий" возможности класса A, добавляющий ему несколько новых свойств и методов. Сделать это можно двумя принципиально различными способами. Первый выглядит примерно так:

```
<?php
class A {
  function TestA() { ... }
  function Test() { ... }
}
class B {
  var $a; // объект класса A
  function B(параметры_для_A, другие_параметры){
    $a = new A(параметры_для_A);
    // инициализируем другие поля B
  }
  function TestB() { ... }
  function Test() { ... }
}
?>
```

Поясним: в этой реализации объект класса B содержит в своем составе подобъект класса A в качестве свойства. Это свойство — лишь "частичка" объекта класса B, не более того. Подобъект не "знает", что он в действительности не самостоятелен, а содержится в классе B, поэтому не может предпринимать никаких действий, специфичных для этого класса.
Мы хотели получить расширение возможностей класса A, а не нечто, содержащее объекты A. Что означает "расширение"? Лишь одно: мы бы хотели, чтобы везде, где допустима работа с объектами класса A, была допустима и работа с объектами класса B. Но в приведенном примере это совсем не так.

Итак, мы имеем некоторые проблемы:

1. Мы не видим явно, что класс B лишь расширяет возможности A, а не является отдельной сущностью;

2. Мы должны обращаться к "части A" класса B через $obj->a->TestA(), а к членам самого класса B как $obj->TestB(). Последнее может быть довольно утомительным, если, как это часто бывает, в B будет использоваться очень много методов из A и гораздо меньше — из B. Кроме того, это заставляет нас постоянно помнить о внутреннем устройстве класса B.

Теперь на практике рассмотрим, что же представляет собой наследование (или расширение возможностей) классов:

```
<?php
class B extends A {
  function B(параметры_для_A, другие_параметры) {
    $this->A(параметры_для_A);
    // инициализируем другие поля B
  }
  function TestB() { ... }
  function Test() { ... }
}
?>
```

Ключевое слово extends говорит о том, что создаваемый класс является лишь "расширением" класса A, и не более того. То есть B содержит те же самые свойства и методы, что и A, но, помимо них и еще некоторые дополнительные, "свои".

Теперь "часть A" находится прямо внутри класса B и может быть легко доступна, наравне с методами и свойствами самого класса B. Например, для объекта $obj класса B допустимы выражения $obj->TestA() и $obj->TestB().

Итак, мы видим, что, действительно, класс B является воплощением идеи "расширение функциональности
класса A". Обратите также внимание: мы можем теперь забыть, что B унаследовал от A некоторые свойства или методы — снаружи все выглядит так, будто класс B реализует их самостоятельно.

Немного о терминологии: родительский класс A принято называть базовым классом, а класс дочерний класс B — производным от A. Иногда базовый класс также называют суперклассом, а производный — подклассом.

Рассмотрим еще один пример на PHP:

```
<?php
class Parent {
 function parent_funct() { echo "<h1>Это родительская функция</h1>"; }
 function test () { echo "<h1>Это родительский класс</h1>"; }
}

class Child extends Parent {
 function child_funct() { echo "<h2>Это дочерняя функция</h2>"; }
 function test () { echo "<h2>Это дочерний класс</h2>"; }
}

$object = new Parent;
$object = new Child;

$object->parent_funct(); // Выводит 'Это родительская функция'
$object->child_funct(); // Выводит 'Это дочерняя функция'
$object->test(); // Выводит 'Это дочерний класс'
?>
```

Полиморфизм классов в PHP

Полиморфизм (многоформенность) является следствием идеи наследования. В общих словах, полиморфность класса — это свойство базового класса использовать функции производных классов, даже если на момент определения ещё неизвестно, какой именно класс будет включать его в качестве базового и, тем самым, становиться от него производным.

Рассмотрим свойство полиморфности классов на основе следующего примера:

```
<?php
class A {
// Выводит, функция какого класса была вызвана
function Test() { echo "Test from A\n"; }
// Тестовая функция — просто переадресует на Test()
function Call() { Test(); }
}
class B extends A {
// Функция Test() для класса B
function Test() { echo "Test from B\n"; }
}
$a=new A();
$b=new B();
?>
```

Используем следующие следующие команды:

```
$a->Call(); // выводит "Test from A
$b->Test(); // выводит "Test from B"
$b->Call(); // Внимание! Выводит "Test from B"!
```

Обратите внимание на последнюю строчку: вопреки ожиданиям, вызывается не функция Test() из класса A, а функция из класса B! Складывается впечатление, что Test() из B просто переопределила функцию Test() из A. Так оно на самом деле и есть. Функция, переопределяемая в производном классе, называется виртуальной.

Механизм виртуальных функций позволяет, например, "подсовывать" функциям, ожидающим объект одного класса, объект другого, производного, класса. Еще один классический пример — класс, воплощающий собой свойства геометрической фигуры, и несколько производных от него классов — квадрат, круг, треугольник и т. д.

Базовый класс имеет виртуальную функцию Draw(), которая заставляет объект нарисовать самого себя. Все производные классы-фигуры, разумеется, переопределяют эту функцию (ведь каждую фигуру нужно рисовать по-особому). Также у нас есть массив фигур, причем мы не знаем, каких именно. Зато, используя полиморфизм, мы можем, не задумываясь, перебрать все элементы массива и вызвать для каждого из них метод Draw() — фигура сама "решит", какого она типа и как ее рисовать.

А вот еще один практический пример, показывающий свойство класса - полиморфизм:

```
<?php
class Base {
 function funct() {
 echo "<h2>Функция базового класса</h2>";
 }
 function base_funct() {
 $this->funct();
 }
}

class Derivative extends Base {
 function funct() {
 echo "<h3>Функция производного класса</h3>";
 }
}

$b = new Base();
$d = new Derivative();

$b->base_funct();
$d->funct();
$d->base_funct();
// Скрипт выводит:

// Функция базового класса
// Функция производного класса
// Функция производного класса
?>
```

В рассмотренном примере функция base_funct() класса Base была перезаписана одноименной функцией класса Derivative.

Работа с объектами классов PHP

Копирование объектов

Так уж устроен PHP, что в нем все переменные, в том числе и объекты, всегда рассматриваются как простой набор значений и копируются целиком. Например, если у нас есть объект $a и мы выполняем оператор $b=$a, то все содержимое $a будет скопировано в $b один-в-один.

```
<?php
class A {
// Создаем новый метод:
```

```
 function Test() {
 echo "<h1>Hello!</h1>";
 }
}
// Создаем объект класса A:
$a=new A();
// Копируем объект $a:
$b=$a;
// Теперь работаем с новым объектом $b
$b->Test(); // Выводит 'Hello!'
?>
```

Сравнение объектов

В PHP объекты сравниваются очень просто: по именам. Два объекта равны, если они имеют те же самые свойства и значения, а также являются экземплярами одного и того же класса. Сравнение двух объектов осуществляют, используя оператор эквивалентности (===). Вот пример:

```
<?php
class A {
// Создаем новый метод:
 function Test() {
 echo "<h1>Hello!</h1>";
 }
}

// Создаем объект класса A:
$a=new A();
// Создаем объект класса A:
$b=new A();
// Выводит 'Объекты равны':
if ($a===$b) echo "<h3>Объекты равны</h2>";
?>
```

Ссылки на объект

PHP позволяет создавать ссылки на объекты. Вот пример:

```
<?php
class A {
// Создаем новый метод:
 function Test() {
 echo "<h1>Hello!</h1>";
 }
}

// Создаем объект класса A:
$a=new A();
// Ссылка на объект класса A:
$b=& new A();
$b->Test();
?>
```

Задания для самостоятельной работы

Задание 1. Разработать классы на языке PHP для описанных ниже объектов.

№ варианта	Задание
1, 15	**Abiturient**: Фамилия, Имя, Отчество, Адрес, Оценки. Создать массив объектов. Вывести:

	а) список абитуриентов, имеющих неудовлетворительные оценки; б) список абитуриентов, сумма баллов у которых не меньше заданной; в) выбрать N абитуриентов, имеющих самую высокую сумму баллов, и список абитуриентов, имеющих полупроходной балл.
2, 16	**Aeroflot**: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Создать массив объектов. Вывести: а) список рейсов для заданного пункта назначения; б) список рейсов для заданного дня недели; в) список рейсов для заданного дня недели, время вылета для которых больше заданного.
3, 17	**Book**: Автор, Название, Издательство, Год, Количество страниц. Создать массив объектов. Вывести: а) список книг заданного автора; б) список книг, выпущенных заданным издательством; в) список книг, выпущенных после заданного года.
4, 18	**Worker**: Фамилия и инициалы, Должность, Год поступления на работу, Зарплата. Создать массив объектов. Вывести: а) список работников, стаж работы которых на данном предприятии превышает заданное число лет; б) список работников, зарплата которых больше заданной; в) список работников, занимающих заданную должность.
5, 19	**Train**: Пункт назначения, Номер поезда, Время отправления, Число общих мест, Купейных, Плацкартных. Создать массив объектов. Вывести: а) список поездов, следующих до заданного пункта назначения; б) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа; в) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.
6, 20	**Product**: Наименование, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести: а) список товаров для заданного наименования; б) список товаров для заданного наименования, цена которых не превышает указанной; в) список товаров, срок хранения которых больше заданного.
7, 21	**Patient**: Фамилия, Имя, Отчество, Адрес, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести: а) список пациентов, имеющих данный диагноз; б) список пациентов, номер медицинской карты которых находится в заданном интервале.
8, 22	**Bus**: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Создать массив объектов. Вывести: а) список автобусов для заданного номера маршрута; б) список автобусов, которые эксплуатируются больше 10 лет; в) список автобусов, пробег у которых больше 10 000 км.
9, 23	**Customer**: Фамилия, Имя, Отчество, Адрес, Телефон, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: а) список покупателей в алфавитном порядке; б) список покупателей, номер кредитной карточки которых находится в заданном интервале.
10, 24	**File**: Имя файла, Размер, Дата создания, Количество обращений. Создать массив объектов. Вывести: а) список файлов, упорядоченный в алфавитном порядке; б) список файлов, размер которых превышает заданный; в) список файлов, число обращений к которым превышает заданное.
11, 25	**Word**: Слово, Номера страниц, на которых слово встречается (от 1 до 10), Число страниц. Создать массив объектов. Вывести:

	а) слова, которые встречаются более чем на N страницах; б) слова в алфавитном порядке; в) для заданного слова номера страниц, на которых оно встречается.
12, 26	**House:** Адрес, Этаж, Количество комнат, Площадь. Создать массив объектов. Вывести: а) список квартир, имеющих заданное число комнат; б) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в определенном промежутке; в) список квартир, имеющих площадь, превосходящую заданную.
13, 27	**Phone**: Фамилия, Имя, Отчество, Адрес, Номер, Время внутри городских разговоров, Время междугородних разговоров. Создать массив объектов. Вывести: а) сведения об абонентах, время внутригородских разговоров которых превышает заданное; б) сведения об абонентах, воспользовавшихся междугородней связью; в) сведения об абонентах, выведенные в алфавитном порядке.
14, 28	**Person**: Фамилия, Имя, Отчество, Адрес, Пол, Образование, Год рождения. Создать массив объектов. Вывести: а) список граждан, возраст которых превышает заданный; б) список граждан с высшим образованием; в) список граждан мужского пола.

Задание 2. Требуется создать базовый класс и определить общие и специфические методы для данного класса. Создать производные классы, в которые добавить свойства и методы. Часть методов переопределить. Создать массив объектов базового класса и заполнить объектами производных классов. Предусмотреть передачу аргументов конструкторам базового класса.

Вариант	Задание
1	Создать базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.
2	Создать базовый класс «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость перевозки для указанных городов и расстояний.
3	Создать класс «Работник фирмы» и производные классы «Менеджер», «Администратор», «Программист».
4	Смоделировать базу данных клиентов бензоколонки. Каждый клиент имеет индивидуальный код, имя. База должна хранить код клиента, дату и время заправки, количество заправленного топлива. Требуется выдавать информацию о заправках как по дате, так и по клиенту.
5	Создать базовый класс «Учащийся» и производные классы «Школьник» и «Студент». Создать массив объектов базового класса и заполнить этот массив объектами. Показать отдельно студентов и школьников.
6	Создать базовый класс «Музыкальный инструмент» и производные классы «Ударный», «Струнный», «Духовой». Создать массив объектов «Оркестр». Выдать состав оркестра, переопределив метод.
7	Создать базовый класс «Садовое дерево» и производные классы «Яблоня», «Вишня», «Груша» и др. С помощью конструктора автоматически установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.
8	Создать базовый класс «Пассажироперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость передвижения.
9	Создать базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай» и др. С помощью конструктора установить имя каждого животного и его характеристики.
10	Создать базовый класс «Садовое дерево» и производные классы «Яблоня», «Вишня»,

	«Груша» и др. С помощью конструктора автоматически установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.
11	Смоделировать процесс сортировки ж/д состава на Т-образном сортировочном узле. Программа должна разделять на 2 направления состав, состоящий из вагонов двух типов (на каждое направление фор-мируется состав из вагонов одного типа).
12	Смоделировать процесс наличия автобусов в автобусном парке. В начальный момент формируются данные о всех автобусах в парке. При работе программы формируются 2 списка: 1 – об автобусах, находящихся в парке, 2 – об автобусах, находящихся на маршруте. При выезде автобуса вводится его номер и программа удаляет его запись из списка автобусов, находящихся в парке, и добавляет его в список автобусов, находящихся на маршруте. При въезде в парк производится обратная операция. По запросу выводится информация об автобусах, находящихся на маршруте и в парке с указанием времени выезда и заезда соответственно.
13	Смоделировать процесс заполнения диска емкостью 360К. В процессе работы файлы произвольной длины от 16К до 32К (выбираются случайным образом) записываются на диск или удаляются с него. При попытке записать файл длиной более длины непрерывного свободного участка файл не записывается и выдается сообщение. По запросу выдается информация о свободных и занятых участках.
14	Написать класс одномерного массива строк, каждая строка может иметь произвольную длину. Определить операции индексации, операции + и – с правым операндом типа char* для добавления и удаления строки в массив.
15	Смоделировать процесс выдачи книг из библиотеки. Каждая книга характеризуется автором (одним), названием, годом издания и общим количеством экземпляров. Кроме списка книг требуется создать список читателей. Для каждого читателя и книги требуется хранить информацию о соответственно книгах, которые читатель взял, и читателях, взявших книгу. Требуется выдавать информацию: по книге - о возможности выдачи, списке читателей, взявших книгу; по читателю – о книгах у читателя.
16	Смоделировать базу данных клиентов бензоколонки. Каждый клиент имеет индивидуальный код, имя. База должна хранить код клиента, дату и время заправки, количество заправленного топлива. Требуется выдавать информацию о заправках как по дате, так и по клиенту.
17	Создать базовый класс «Пассажироперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость передвижения.
18	Создать базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай» и др. С помощью конструктора установить имя каждого животного и его характеристики.
19	Создать базовый класс «Садовое дерево» и производные классы «Яблоня», «Вишня», «Груша» и др. С помощью конструктора автоматически установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

Задание 3. Разработать иерархию классов с использованием виртуальных функций.

Номер варианта	**Задание**
1, 9, 17	Разработать программу с использованием наследования классов, реализующую классы: • графический объект; • круг; • квадрат. Используя виртуальные функции, выведите на экран размер и координаты графического объекта.
2, 10, 18	Разработать программу с использованием наследования классов, реализующую классы:

	• железнодорожный вагон; • вагон для перевозки автомобилей; • цистерна. Используя виртуальные функции, выведите на экран вес железнодорожного вагона и количество единиц товара в вагоне.
3, 11, 19	Разработать программу с использованием наследования классов, реализующую классы: • массив; • стек; • очередь. Используя виртуальные функции, выведите на экран количество элементов и добавьте элемент.
4, 12, 20	Разработать программу с использованием наследования классов, реализующую классы: • воин; • пехотинец(винтовка); • матрос(кортик). Используя виртуальные функции, выведите на экран возраст и вид оружия.
5, 13, 21	Разработать программу с использованием наследования классов, реализующую классы: • точка; • линия; • круг. Используя виртуальные функции, выведите на экран координаты и размер.
6, 14, 22	Разработать программу с использованием наследования классов, реализующую классы: • работник больницы; • медсестра; • хирург. Используя виртуальные функции, выведите на экран возраст и название должности.
7, 15, 23	Разработать программу с использованием наследования классов, реализующую классы: • точка; • квадрат • пирамида. Используя виртуальные функции, выведите на экран размер и координаты.
8, 16, 24	Разработать программу с использованием наследования классов, реализующую классы: • массив; • стек; • очередь. Используя виртуальные функции, выведите на экран количество элементов и добавьте элемент.