package lab2; (Название пакета, с которым мы работаем)

import (я импортировала библиотеку новую, чтобы брать информацию оттуда)

public class Programm { (наш класс)

public static void (метод ничего не возвращает) main(String[] args) (массив аргументов) { (это метод, который называется main, )

при помощи static я имею доступ к переменным из других классов, не называя саму переменную заново

1. **Объектно-ориентированное программирование. Основные понятия: объекты, наследование, полиморфизм, инкапсуляция.**

Объе́ктно-ориенти́рованное программирование (ООП) — это представление программы в виде совокупности объектов, состоящая из классов, а классы образуют наследование.

Основные принципы ООП: абстракция, инкапсуляция, наследование, полиморфизм.

**Абстракция** — означает выделение значимой информации и исключение из рассмотрения незначимой. С точки зрения программирования это правильное разделение программы на объекты. Абстракция позволяет отобрать главные характеристики и опустить второстепенные.

**То есть я не пишу какие-то ненужные характеристики для объекта, один метод для разных объектов.**

**Инкапсуляция** — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе.

**Это защита методов или функций, указывают это модификаторы доступа**

**Наследование** — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

**Полиморфизм**— свойство системы, при использовании которого мы можем не писать отдельные методы для каждого класса, а воспользоваться полиморфизмом.

**Мы задаём объекту общую функцию, а он сам выбирает что из этого сделать**

1. **Понятие класса. Классы и объекты в Java.**

Класс – это шаблон, описывающий общие свойства группы объектов. Этими свойствами могут быть как характеристики объектов (размер, вес, цвет и т.п.), так и поведения, роли и т.п.

Любой файл является классом

Метод — это последовательность команд, которые вызываются по определенному имени. Можно сказать, что это функция и процедура (в случае void метода).

Функции, которые записываются в классе, называются **методами**.

Переменные, которые записываются в классе, называются **полями**.

1. **Члены класса. Модификаторы доступа.**

**private** (закрытый) — доступ к члену класса не предоставляется никому, кроме методов этого класса.

**default, package, friendly,** доступ по умолчанию, когда никакой модификатор не присутствует — член класса считается открытым внутри своего собственного пакета, но не доступен для кода, расположенного вне этого пакета.

**protected** (защищённый) — доступ в пределах пакета и классов наследников. Доступ в классе из другого пакета будет к методам public и protected главного класса.

**public** (открытый) — доступ для всех из любого другого кода проекта

Модификаторы в списке расположены по возрастающей видимости в программе.

1. Создание и инициализация объектов. Вызов методов.
2. Области видимости переменных.

У каждой переменной есть область видимости, т.е. та область программы, в которой возможна работа с этой переменной. Таким образом у переменных с одинаковыми именами не должны пересекаться области видимости.

{ int n;

int n; {

 } int n; // нельзя

 int n; // можно }

Но избежать ошибки можно, если я в принтлн допишу this.n, так он показывает о том, что эта переменная находится в этом классе.

1. Модификаторы final и static.

**Модификаторы в Java** – это ключевые слова, которые придают классу, полю класса или методу определенные свойства.

Для обозначения видимости класса его методов и полей есть 4 модификатора доступа:

* **private** члены класса доступны только внутри класса;
* **package-private или default (по умолчанию)** члены класса видны внутри пакета;
* **protected** члены класса доступны внутри пакета и в классах-наследниках;
* **public** члены класса доступны всем.

**Static** — модификатор, применяемый к полю, блоку, методу или внутреннему классу. Данный модификатор указывает на привязку субъекта к текущему классу. Его можно применять перед методом, полем и даже классом, когда хотим объявить вложенный класс.

**Можно создавать методы(функции), не создавая самих объектов, то есть сразу записывать функцию, без объявления переменной.**

**Или также можно применить статик к переменной, и тогда мы можем через точку написать переменную в главном классе и заменить на новое значение.**

Применяя **final** модификатор, Вы говорите, что поля не могут быть изменены, методы переопределены, а классы нельзя наследовать. Этот модификатор применяется только к классам, методам и переменным (также и к локальным переменным).

**Это значит, что мы создаём константу, которая не подлежит изменению.**

**Ещё значит, что у класса не может быть классов наследников (это если после паблик написать)**

1. Пакеты, инструкция import.

Конструкторы

Что такое нулевой конструктор