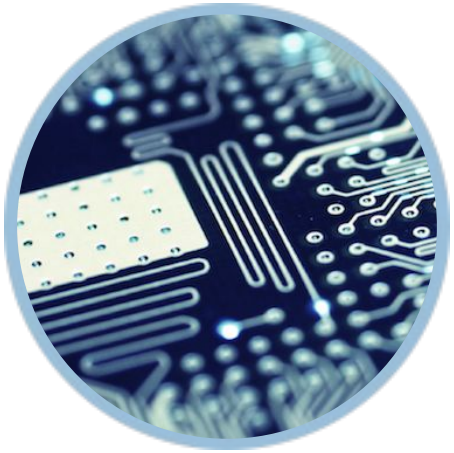


Объектно-ориентированное программирование

ООП

Небольшой flashback

Абстракция

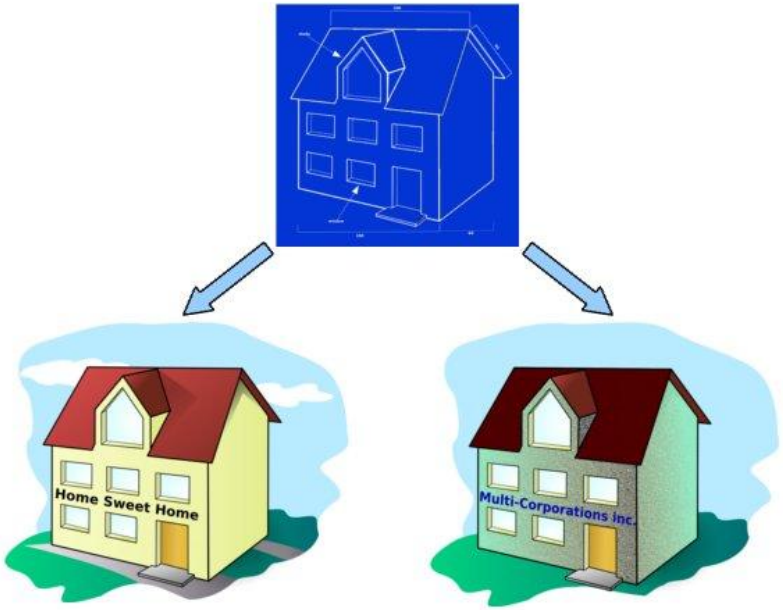
Основной принцип ООП. Заключается в выделении деталей, необходимых для решения конкретной задачи. Несущественные детали отбрасываются.

Объект

- Данные

Класс

- Описание данных
- Методы



Инкапсуляция

Принцип ООП, обеспечивающий сокрытие деталей реализации класса и тем самым защищая его внешнего вмешательства или неправильного использования

```
public namespace Program
{
    public class Car
    {
        private string _name;
        private string _color;

        public string GetName()
        {
            return _name;
        }

        public void SetName(string name)
        {
            _name = name;
        }
    }
}
```

Инкапсулированные переменные

Интерфейс доступа

Наследование

Концепция ООП, означающая возможность описания нового класса на основе уже существующего, частично или полностью заимствуя его функциональность. Класс, от которого производится наследование, называется базовым, родительским. Новый класс — потомком, наследником, дочерним или производным классом.

Транспорт

Колесный транспорт

Полиморфизм

Полиморфи́зм — возможность объектов с одинаковой спецификацией иметь различную реализацию.

Статические члены класса

- В сигнатуре есть ключевое слово static
- Взаимодействуют только с другими статическими членами (не имеют доступа к состоянию объекта, поскольку объекта просто нет)
- К ним нужно обращаться через тип (класс) в котором они описаны

Примеры

```
8 references | 0 changes | 0 authors, 0 changes
public class CarType
{
    private string _typeName;

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string GetTypeName()
    {
        return _typeName;
    }

    5 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public CarType(string typeName)
    {
        _typeName = typeName;
    }
}
```

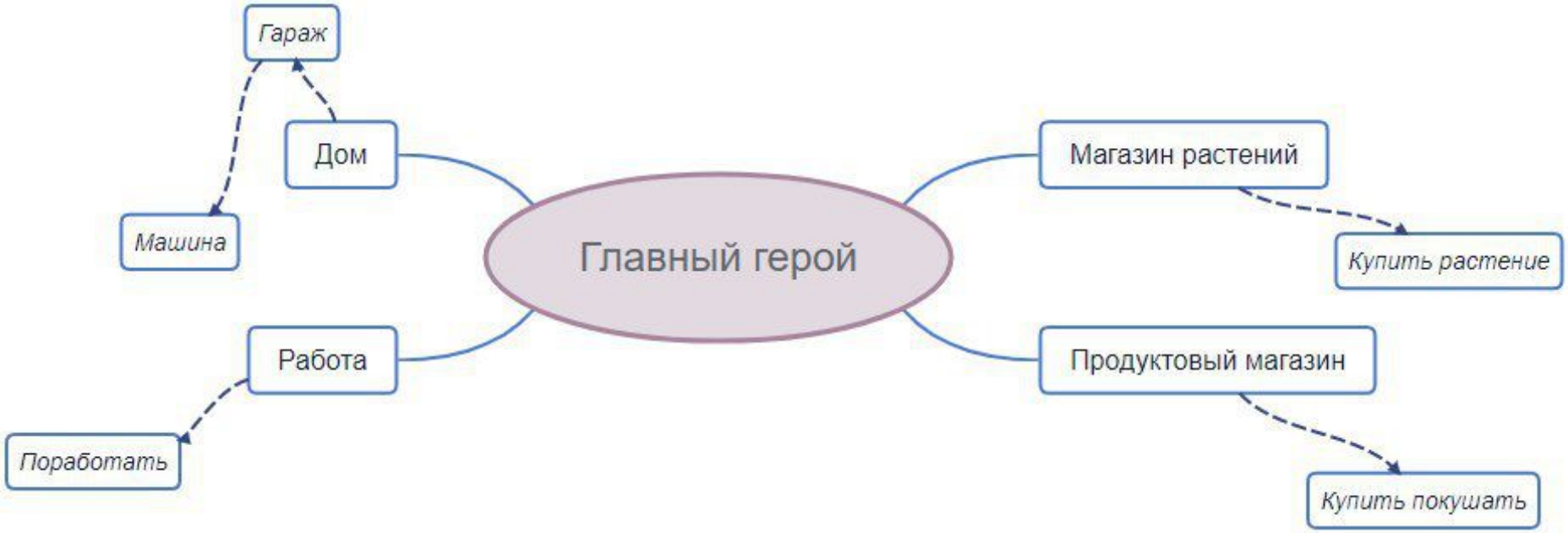
```
0 references | 0 changes | 0 authors, 0 changes
public class Car
{
    private static CarType[] _definedTypes =
    {
        new CarType("Автомобили"),
        new CarType("Автобусы"),
        new CarType("Мотоциклы"),
        new CarType("Прицепы"),
        new CarType("Полуприцепы")
    };

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public static CarType[] GetCarTypes()
    {
        return _definedTypes;
    }
}
```

Примеры

```
8 references | 0 changes | 0 authors, 0 changes
public class CarType
{
    private readonly string _typeName;

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string GetTypeName()
    {
        return _typeName;
    }

    5 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public CarType(string typeName)
    {
        _typeName = typeName;
    }
}
```

```
1 reference | 0 changes | 0 authors, 0 changes
public class Car
{
    private static readonly CarType[] DefinedTypes;

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    static Car()
    {
        DefinedTypes = new[]
        {
            new CarType("Автомобили"),
            new CarType("Автобусы"),
            new CarType("Мотоциклы"),
            new CarType("Прицепы"),
            new CarType("Полуприцепы")
        };
    }

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public static CarType[] GetCarTypes()
    {
        return DefinedTypes;
    }
}
```

Задание


Гараж
Дом
Машина
Главный герой
Магазин растений
Купить растение
Работа
Поработать
Продуктовый магазин
Купить покушать