PRODUCT
STAR.RU

Объектно-ориентированное программирование

Содержание урока

- Что такое ООП и почему оно необходимо?
- Классы и объекты
- Абстракция и Инкапсуляция
- Наследование
- Полиморфизм

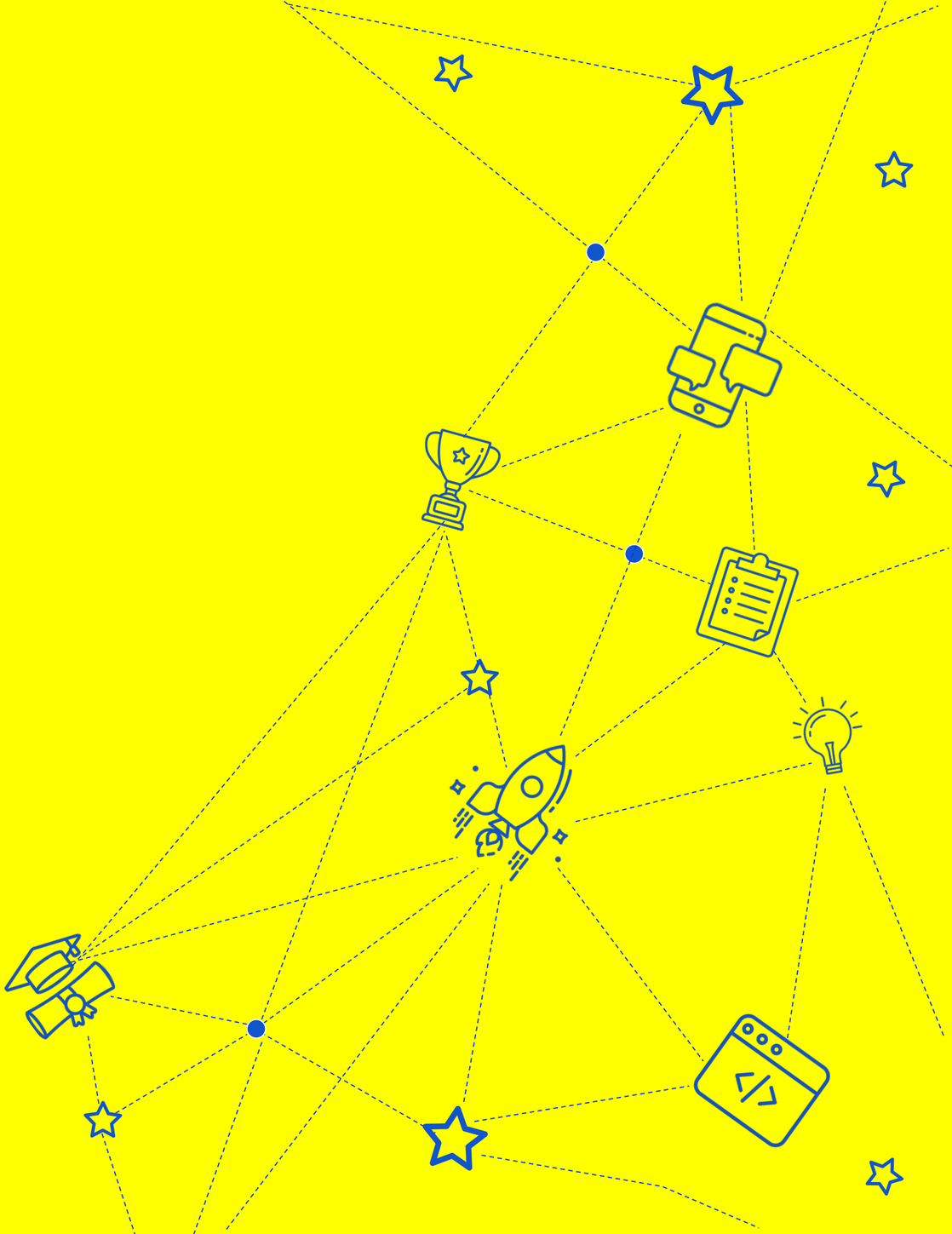
ДМИТРИЙ ПОНОМАРЕВ

Senior Software Engineer at FAANG

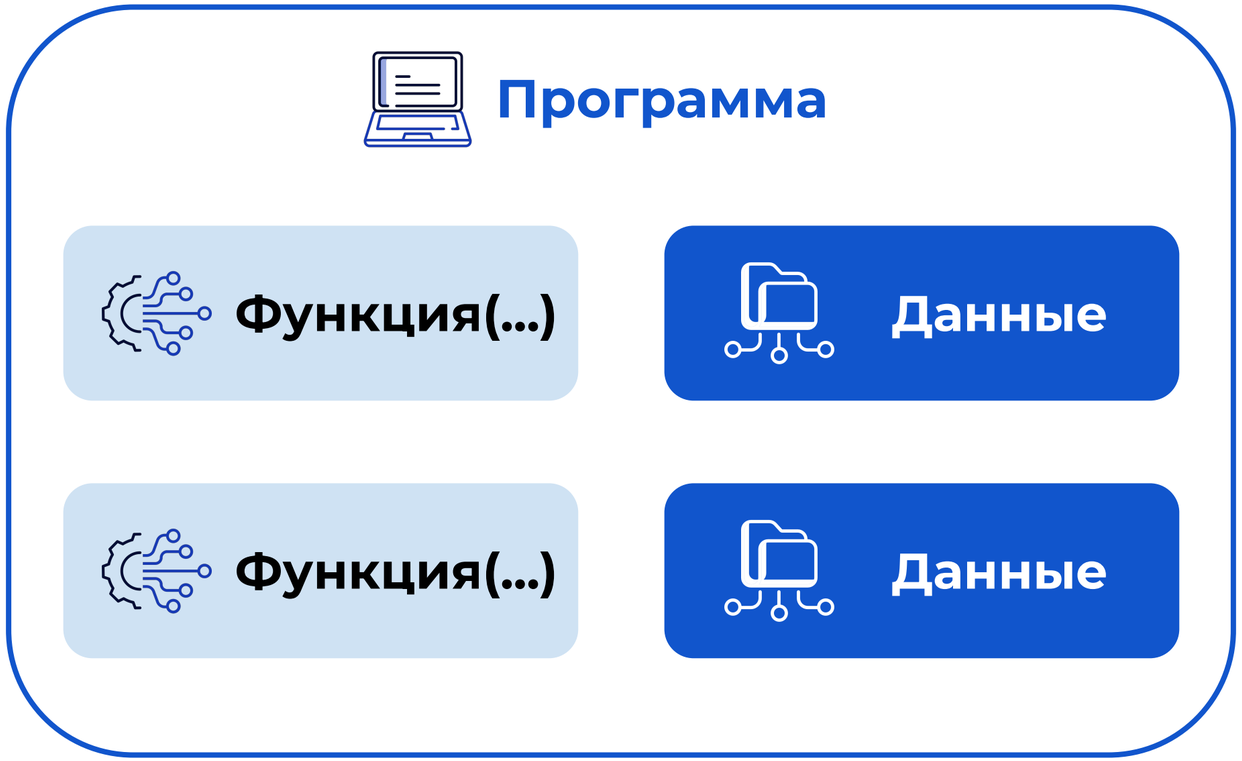
Ex-Software Engineer at Amazon,
Trivago, Deutsche Bank

- Выпускник МФТИ
- Кандидат технических наук
- Более 15 лет опыта работы в IT

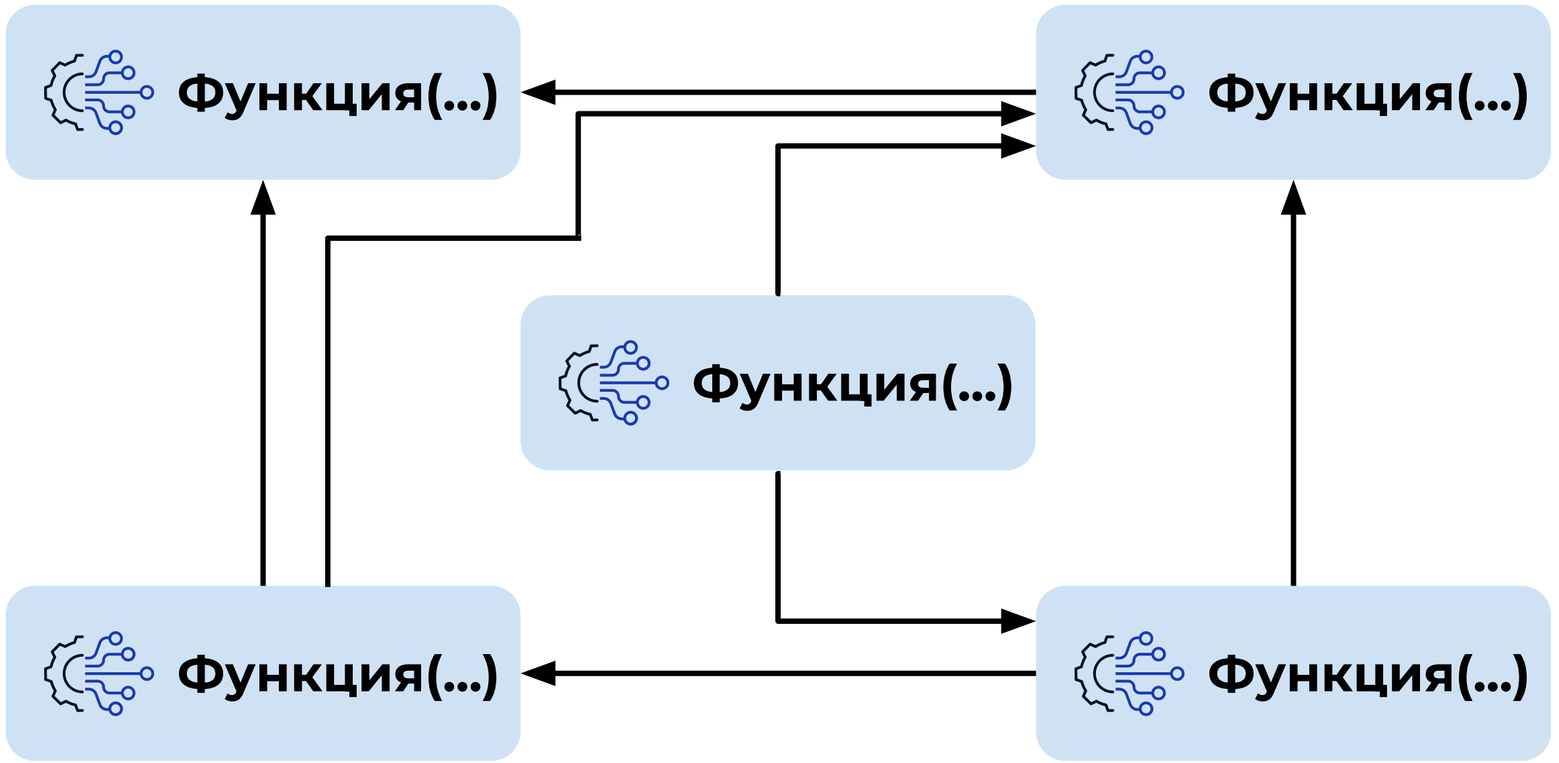
Что такое ООП и почему оно необходимо?



Процедурное программирование



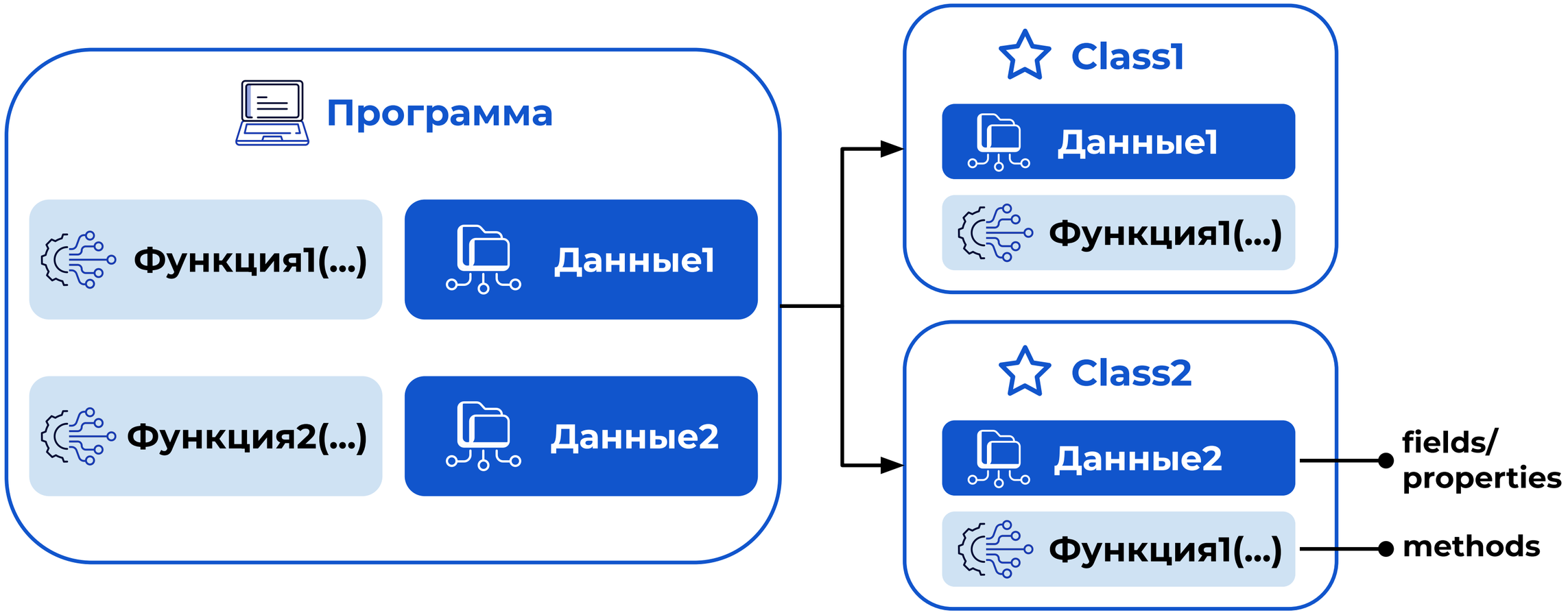
Процедурное программирование



Процедурное программирование

Спагетти-код

ООП


Программа
Функция1(...)
Данные1
Функция2(...)
Данные2
Class1
Данные1
Функция1(...)
Class2
Данные2
Функция1(...)
fields/
properties
methods

Определение ОПП

Объектно-ориентированное программирование
(Формальное определение)
Методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Объектно-ориентированное программирование
Помогает структурировать код путем группирования кода и данных в связанные по смыслу классы.

Определение ОПП

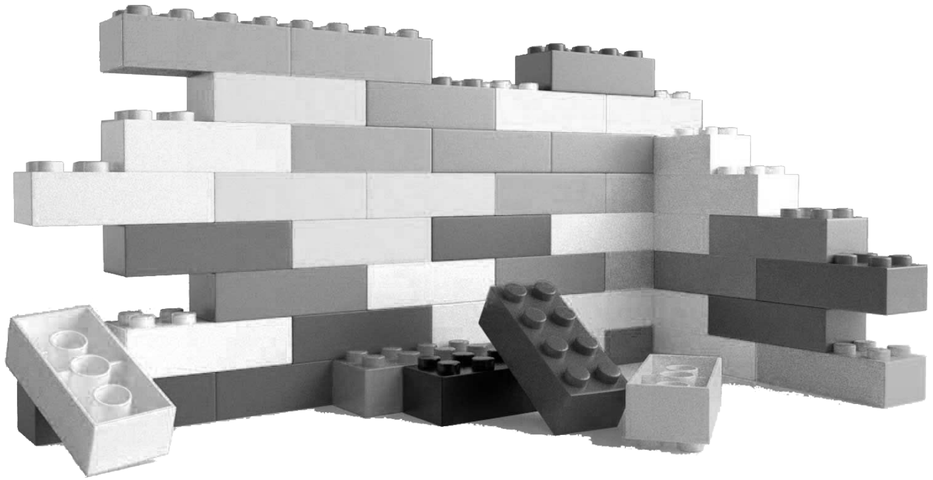
Объектно-ориентированное программирование

(Формальное определение)

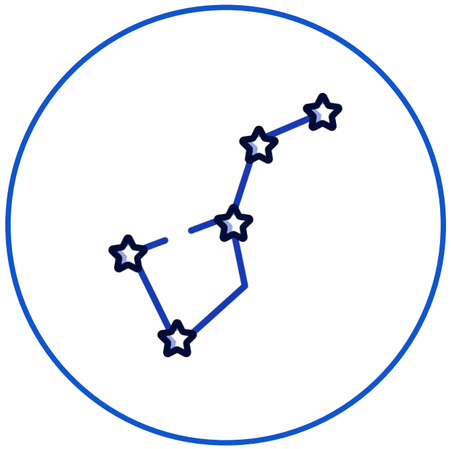
Методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Объектно-ориентированное программирование

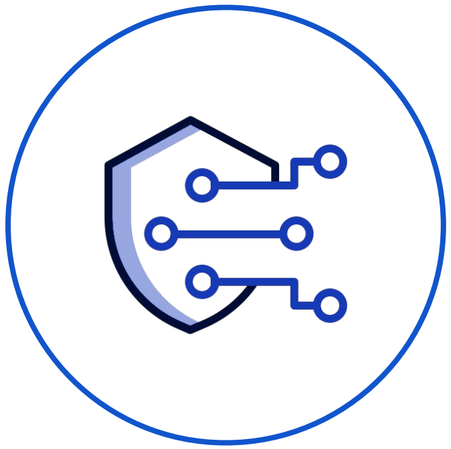
Помогает структурировать код путем группирования кода и данных в связанные по смыслу классы.



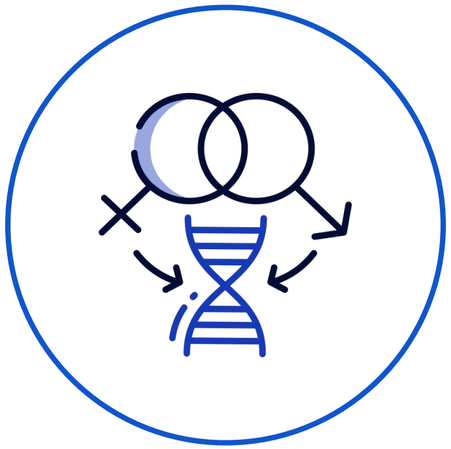
Основные принципы ООП



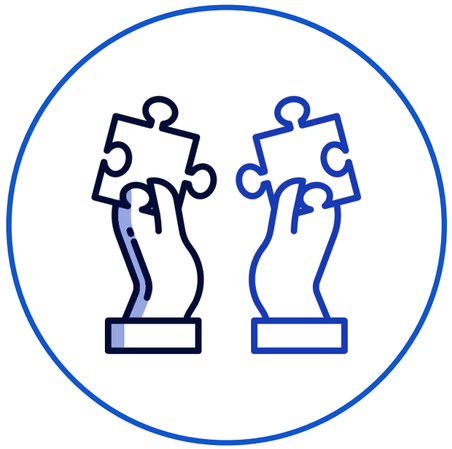
Абстракция



Инкапсуляция



Наследование



Полиморфизм

Основные парадигмы программирования

Императивное программирование

- Процедурное программирование
- ООП

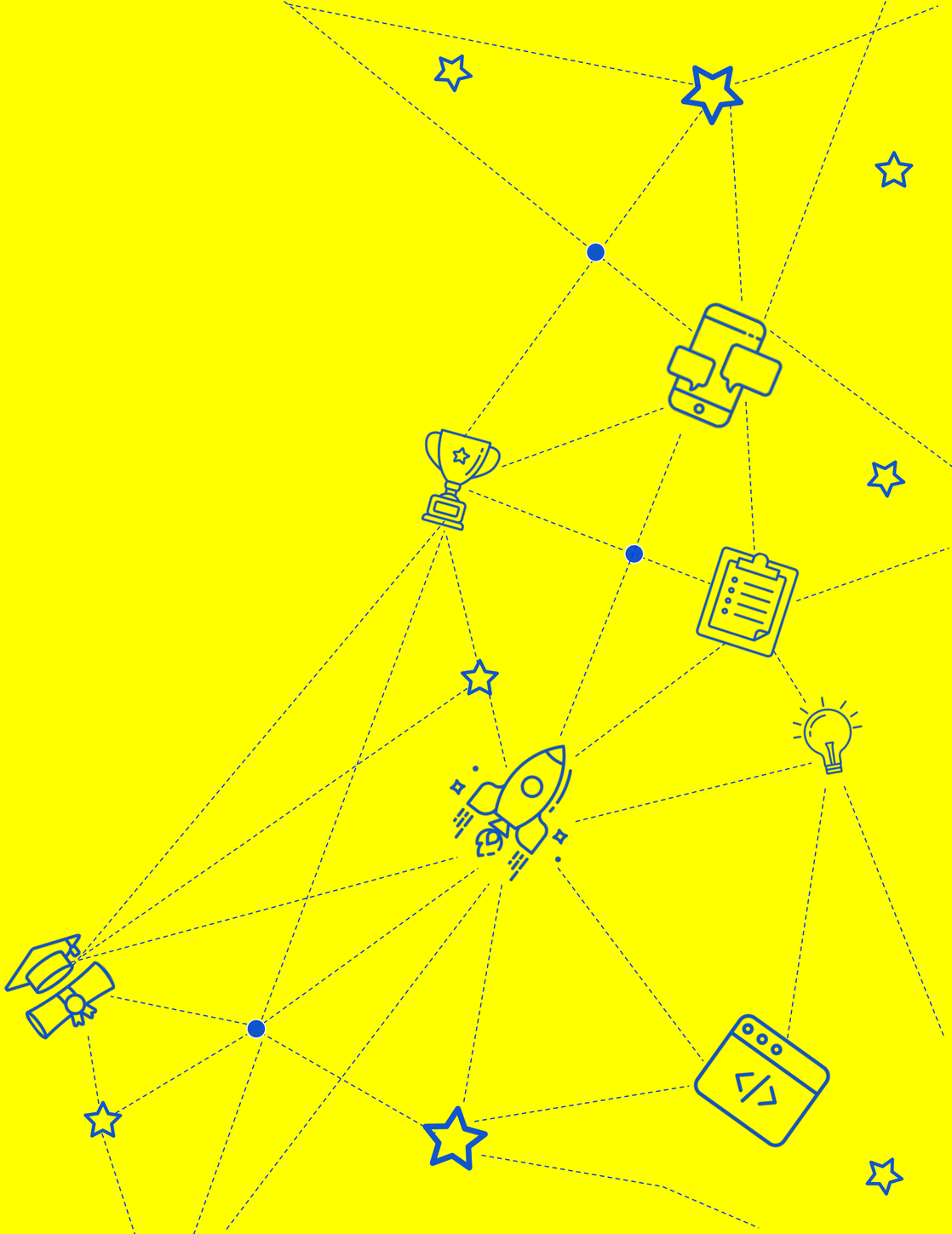
Декларативное программирование

- Функциональное программирование
- Реактивное программирование
- Логическое программирование
- Математическое

Примеры языков программирования с поддержкой ООП

- Java
- JavaScript
- C++
- C#
- Scala
- Kotlin
- Python

Классы и объекты



Что такое класс и объект

Класс – универсальный, комплексный тип данных, состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями).

Класс = поля + методы связанные по смыслу.

Например:

- ★ класс BankAccount (банковский счет)
- ★ класс Employee (сотрудник)
- ★ класс Client (клиент)
- ★ класс Button (кнопка)
- ★ и т.д.

Что такое класс и объект

Объект – экземпляр класса.

Например, класс Button и объекты класса Button:

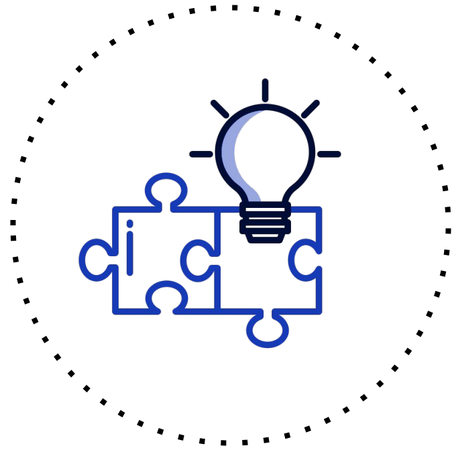
- ★ кнопка Submit
- ★ кнопка Next
- ★ кнопка Back

Пример класса

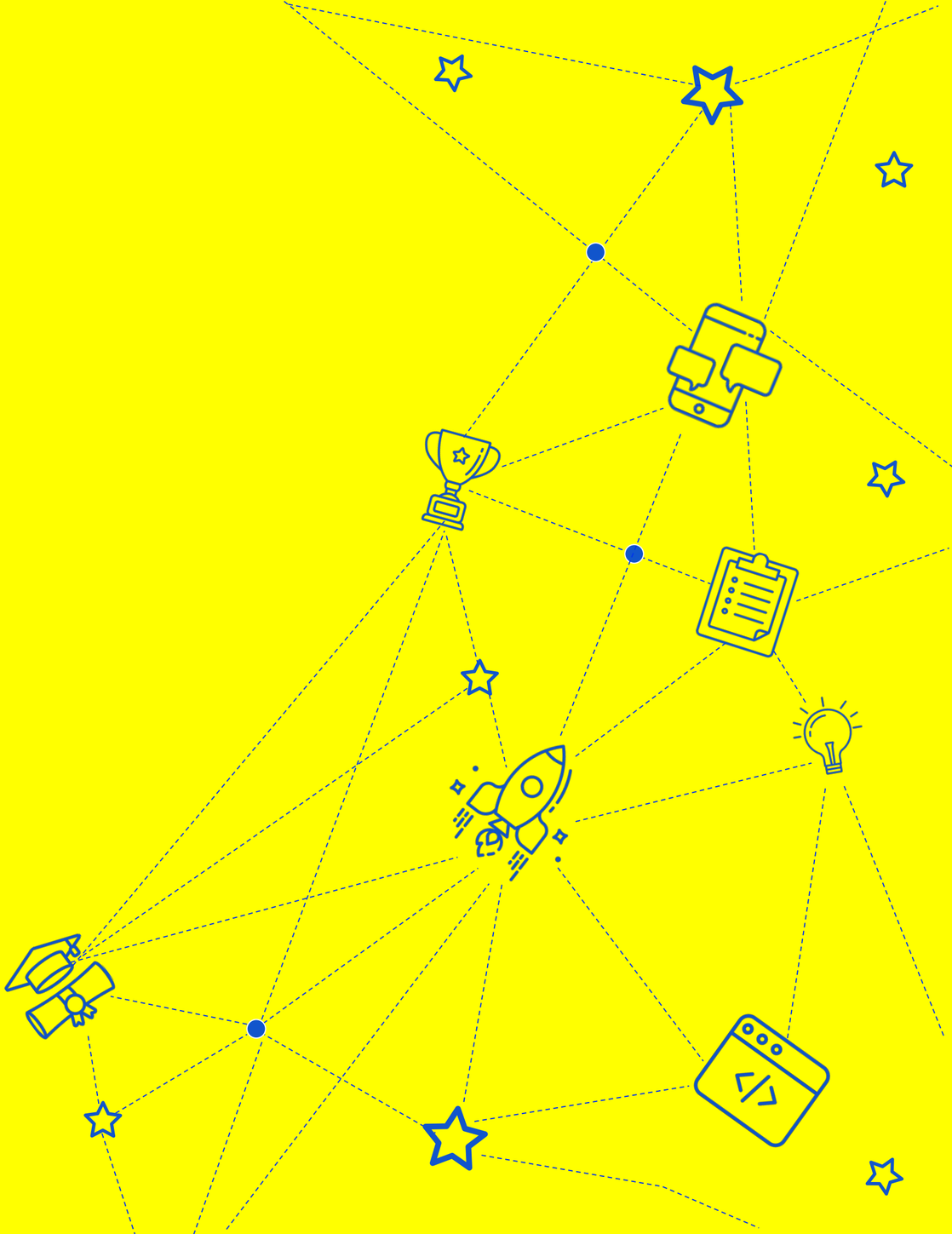
```
public class Employee {
    private String name;
    private String surname;
    private String position;
    private int age;
    private double salary;

    public void increaseSalary(double salaryIncrease) {
        this.salary = this.salary + salaryIncrease;
    }

    public void promote(String newPosition) {
        this.position = newPosition;
    }
}
```

Пример создания объекта

```
public class Main {

    public static void main(String[] args) {
        Employee softwareEngineer = new Employee(“Dmitry”, “Ponomarev”, “Senior Software Engineer”, 35, 10, true);
        Employee writer = new Employee(“Alexander”, “Pushkin”, “Poet”, 37, 10, false);
        softwareEngineer.increaseSalary(10);
    }
}
```



Абстракция
и Инкапсуляция



Абстракция

Абстрагирование означает выделение значимой информации и исключение из рассмотрения незначимой.

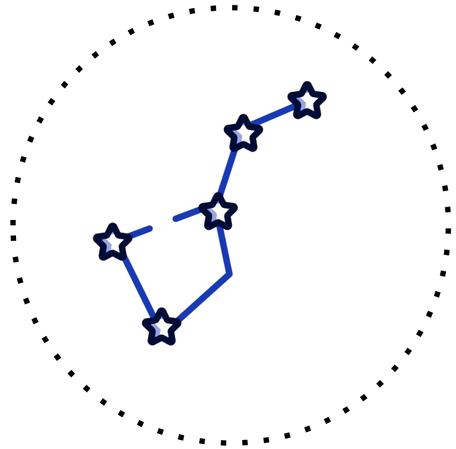
Делает интерфейс взаимодействия проще

Абстракция относится к процессу дизайна(проектирования) кода

Абстракция позволяет скрыть детали реализации

Абстракция

```
class CofeeMachine {
    public Cofee makeCofee() {
        .....
    }
    ....
}
```

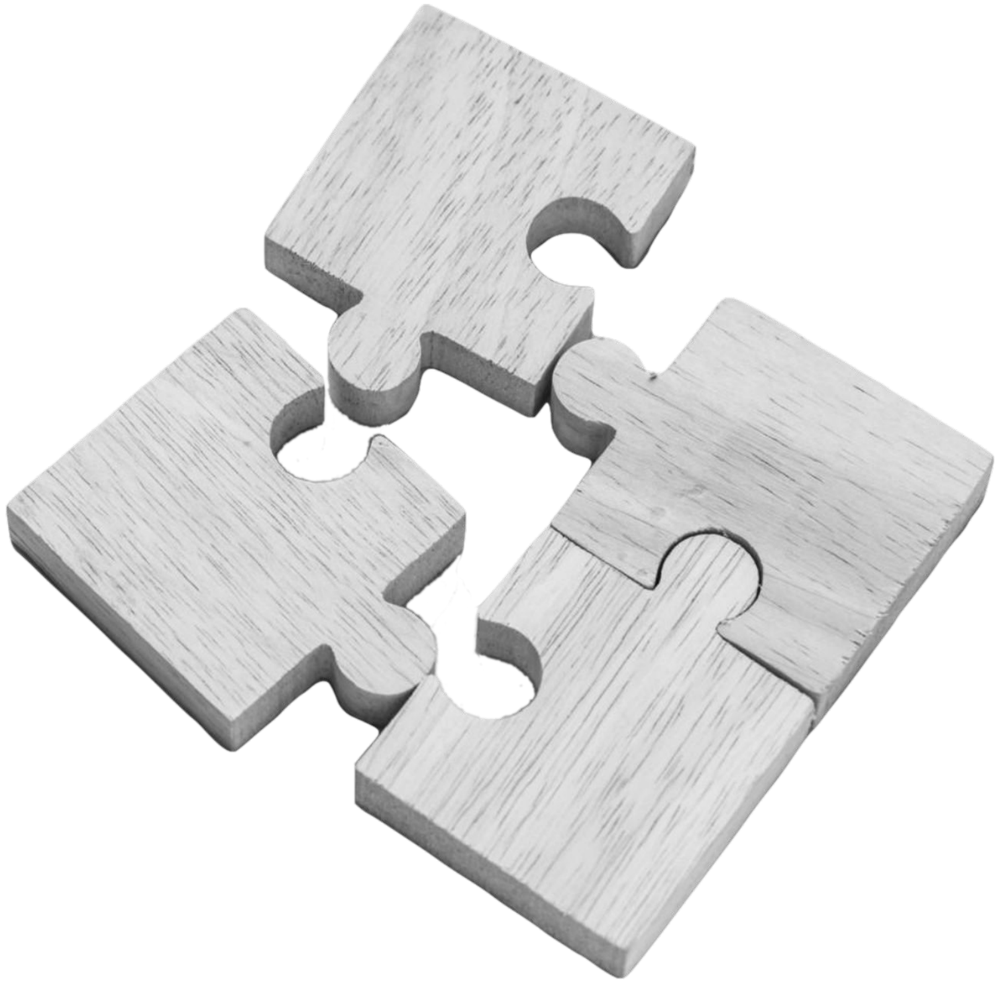


Инкапсуляция

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

Инкапсуляция ограничивает доступ к данным и управляет видимостью кода

Инкапсуляция относится к процессу реализации

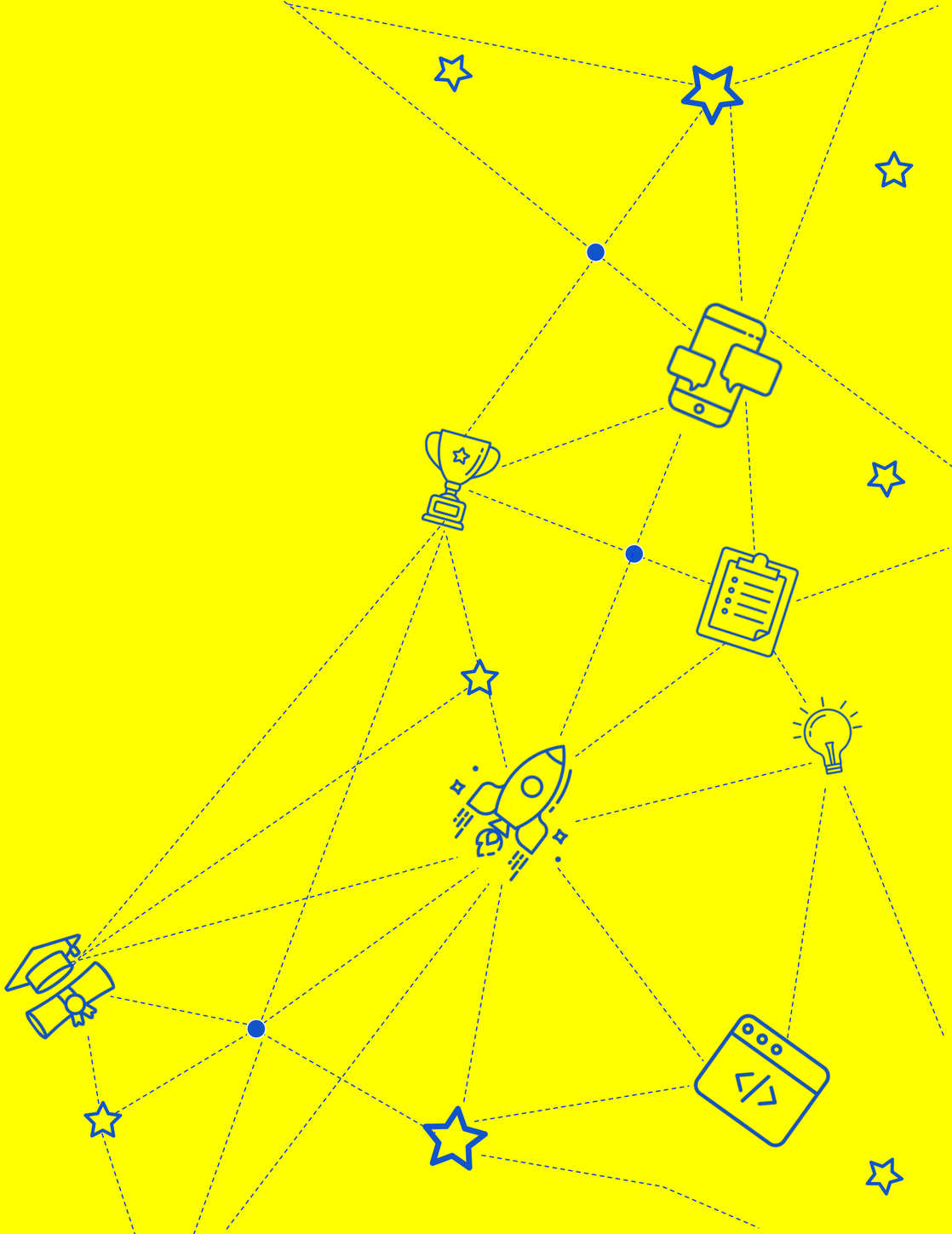


Инкапсуляция

```
class CofeeMachine {
    public Cofee makeCofee() {
        heatWater();
        grindCofeeBeans();
        mixWaterAndCofee();
    }

    private void heatWater() {
        ...
    }

    private void grindCofeeBeans() {
        ...
    }

    private void mixWaterAndCofee() {
        ...
    }
    ...
}
```

```
public class Employee {
    private String name;
    public String getName() {
        return name;
    }
    public void setName (String name) {
        this.name = name
     }
}

class Main {
    public static void main(String[] args) {
        Employee employee = new Employee();
        employee. setName ("Dmitry");
        System.out.printin (s.getName ( )) ;
    }
}
```

Наследование

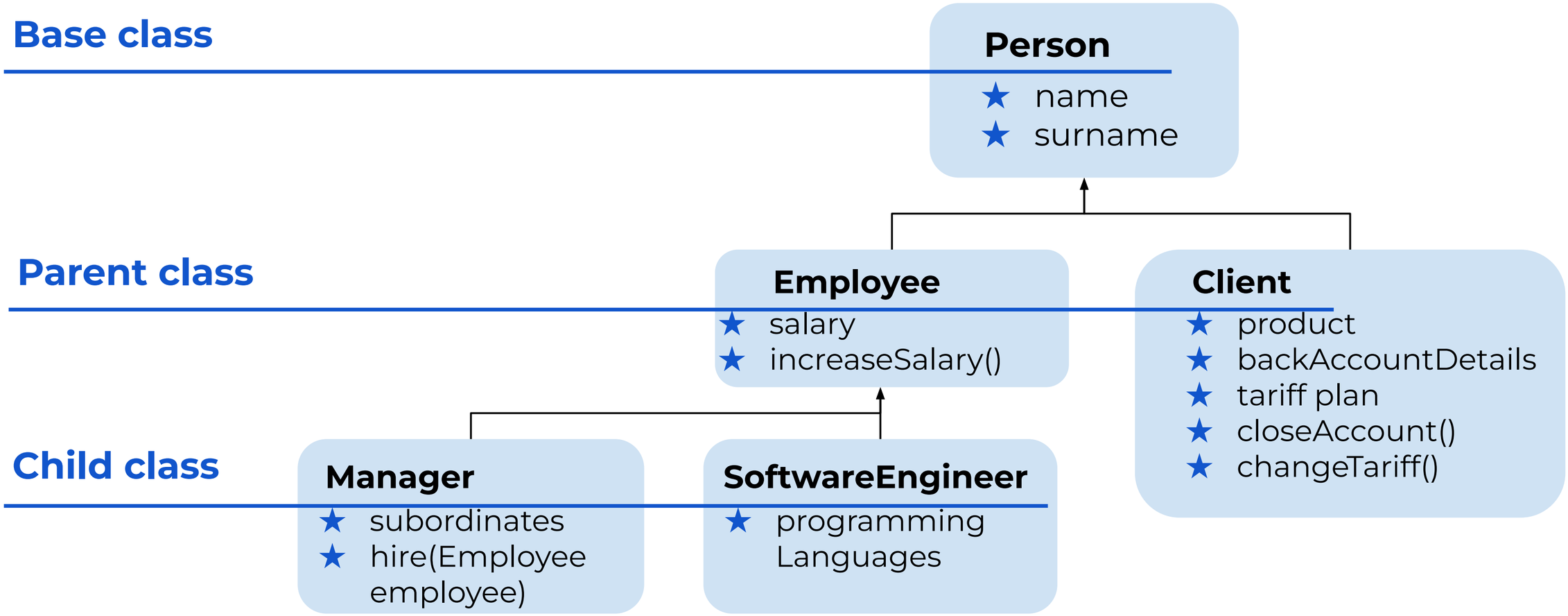


Что такое наследование

Наследование – свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствованной функциональностью.

Позволяет сократить количество повторяющегося кода путем его переиспользования

Пример наследования



Пример наследования

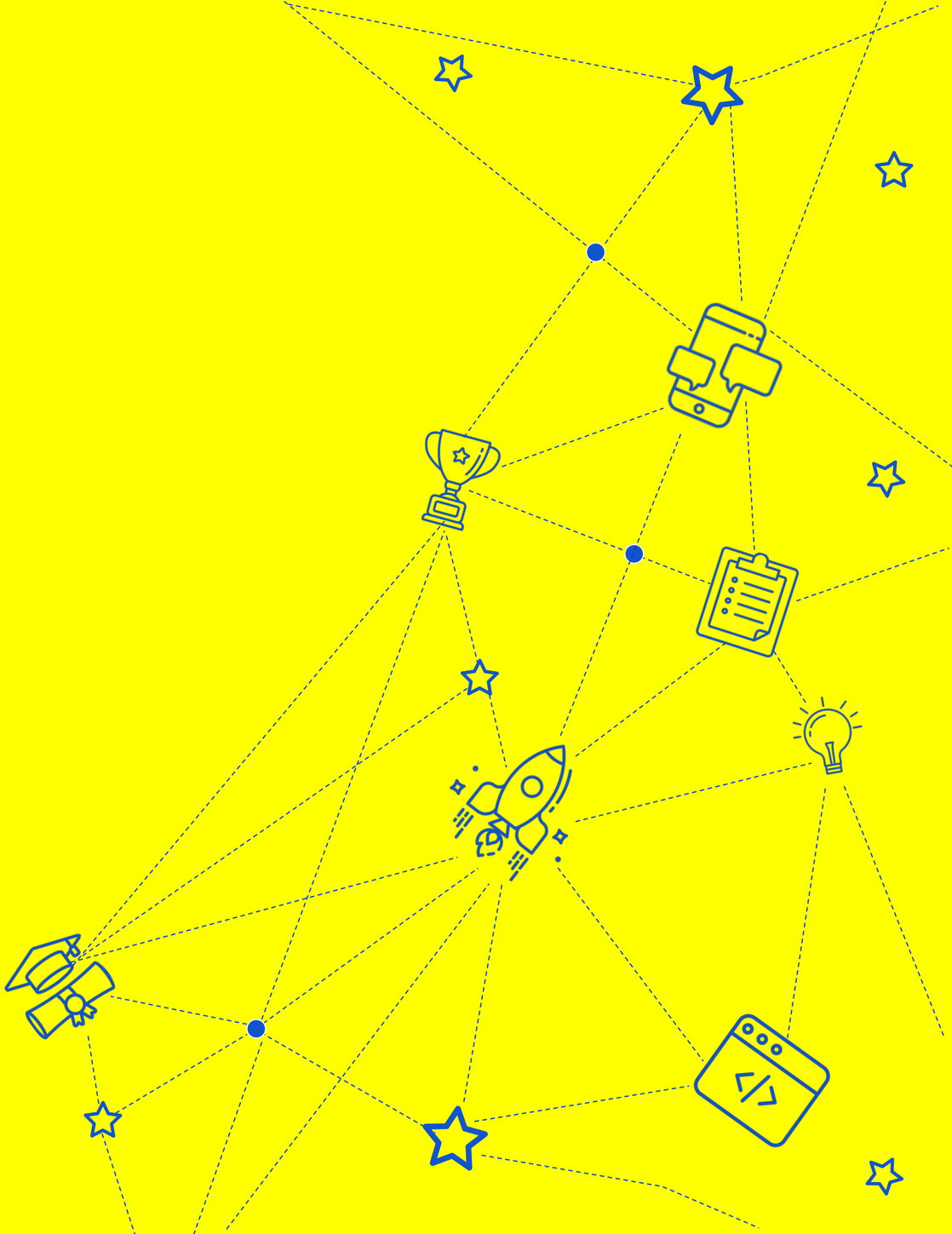
```
class Person {
    String name;
    String surname;
    ...

}
```

```
class Employee extends Person {
    int salary;

    public void increaseSalary(int increase) {
        this.salary = this.salary + increase;
    }
    ....
}
```

Пример наследования

```
class Manager extends Employee {
    List<Employee> subordinates;

    public void hire(Employee employee) {
        this.subordinates.add(employee);
    }
    ...
}
```

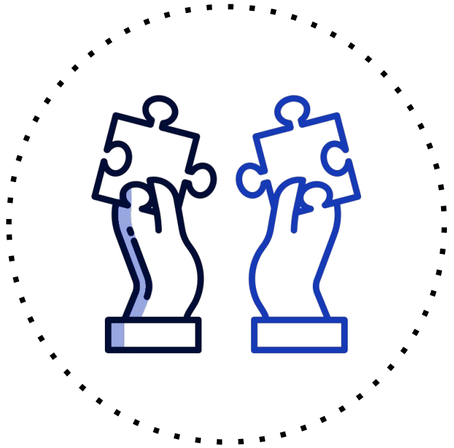
Полиморфизм



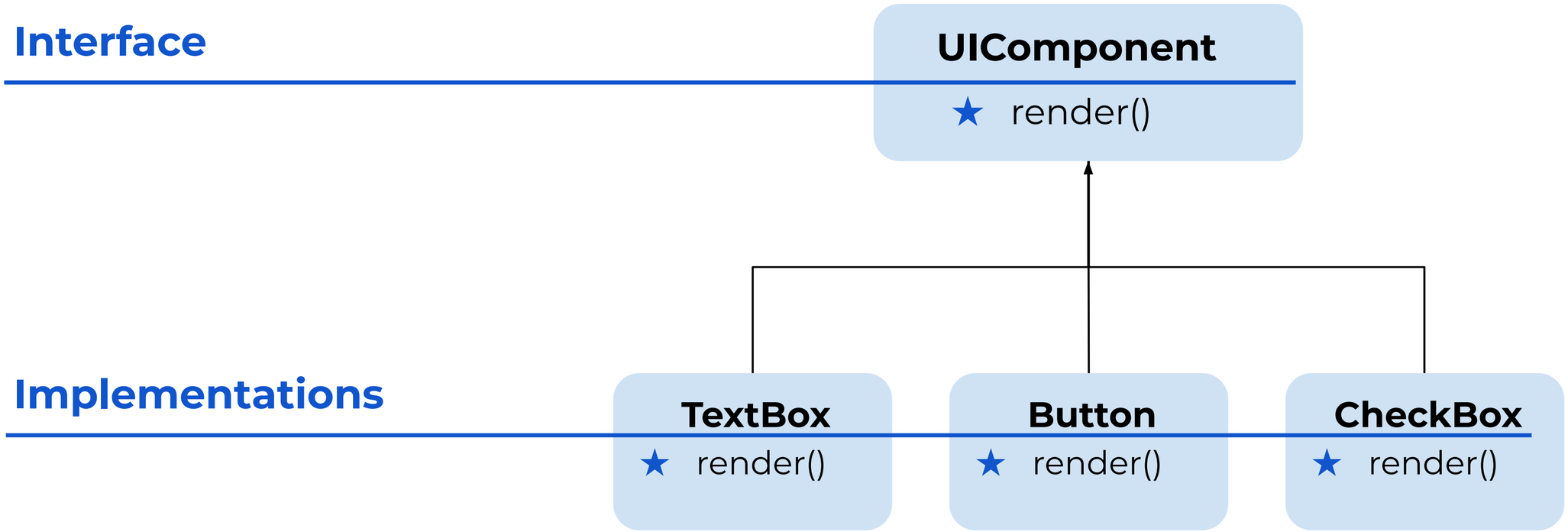
Пример полиморфизма

«Поли» – много, **«морфизм» –** форм

```
public void renderUIComponent(String UIelementType, ...) {
  if (UIelementType.equals("Checkbox")) {
    renderCheckBox(...);
  } else if (UIelementType. equals ("TextBox")) {
    renderTextBox(...);
  } else if (UIelementType.equals ("Button")) {
     renderButton(...);
  }
  ...
}
```



Пример полиморфизма



Пример полиморфизма

```
interface UIComponent {
    void render();
}

class CheckBox implements UIComponent {
    public void render() {
        ....
    }
}

class Button implements UIComponent {
    public void render() {
        ....
    }
}

class TextBox implements UIComponent {
    public void render() {
        ....
    }
}
```

```
public void renderUIComponent(UIComponent
uiComponent) {
        uiComponent.render();
    }
}
....
UIComponent component = new CheckBox();
renderUIComponent(component);
```

Что такое полиморфизм

Полиморфизм – свойство системы,
позволяющее использовать объекты
с одинаковым интерфейсом без информации
о типе и внутренней структуре объекта.

Позволяет избавится
от ненужных **switch** и **if-else**
выражений

Заключение

1. ООП помогает структурировать программу при помощи группирования данных и методов по смыслу в классы.
2. Класс это объединенные по смыслу поля и методы.
3. Объект это экземпляр/инстанс класса. Например, класс Person и конкретные объекты класса Person: Elon Musk, Jeff Bezos, Vasya, Petya.
4. Практически все современные языки программирования используют ООП.
5. Основные принципы ООП: абстракция, инкапсуляция, наследование и полиморфизм.

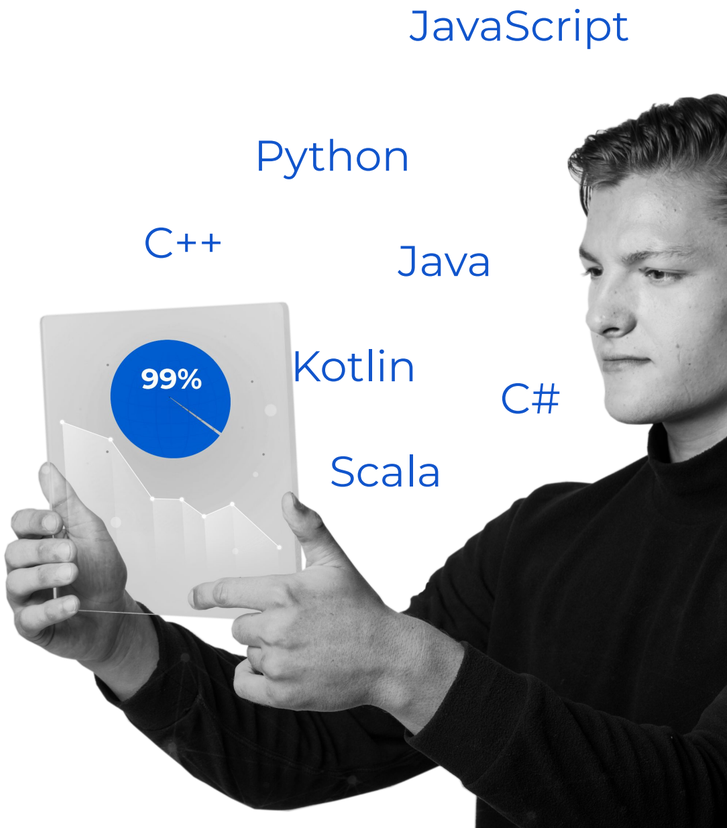
Заключение

6. Инкапсуляция – свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе. А также скрыть внутреннее состояние объекта.

7. Абстракция позволяет скрыть несущественные для клиента класса детали реализации.

8. Наследование позволяет переиспользовать функциональность уже существующего класса.

9. Полиморфизм позволяет использовать объекты разных классов одним и тем же образом, если они реализуют один и тот же интерфейс. Полиморфизм поможет в избавлении от ненужных if-else и switch выражений.



СПАСИБО ЗА ВНИМАНИЕ

Дмитрий Пономарев

https://www.linkedin.com/in/dmitry-ponomarev-77732970/