7. Что такое ООП, расскажите, что вы знаете о ООП.

Объектно-ориентированное программирование - методология программирования, основанная на представлении программного продукта в виде совокупности объектов, каждый из которых является экземпляром конкретного класса.

Для объектно-ориентированного языка Java характерен следующий **подход** к программированию:

- все является объектом;
- программа является набором объектов, указывающих друг другу, что делать посредством посылки сообщений;
- каждый объект имеет некоторый тип;
- все объекты одного типа могут получать одни и те же сообщения.

Объект – это набор данных и методов. Программа, написанная с использованием ООП, обычно состоит из множества объектов, и все эти объекты взаимодействуют между собой.

Объектно-ориентированный подход **помогает**:

- уменьшить сложность ПО;
- повысить надежность ПО;
- обеспечить возможность модификации отдельных компонентов ПО без изменения остальных его компонентов.
- обеспечить возможность повторного использования отдельных компонентов ПО.

Объектно-ориентированное программирование строится на **абстракции**, **инкапсуляции**, **наследовании**, **полиморфизме**.

**Наследование** это процесс благодаря которому один объект может приобрести свойства другого объекта (наследование всех свойств одного объекта другим) и добавлять черты характерны только для него самого. Например:

```
public class Animal {
        public Animal() {
                System.out.println("New animal");
        }

        public void voice() {
                System.out.println("voice...");
        }

        public void sleep() {
                System.out.println("An animal sleeps");
        }

}

public class Dog extends Animal {
        public Dog() {
                super();
                System.out.println("A new dog");
        }

        @Override
        public void sleep() {
```

```
                System.out.println("A dog sleeps");
        }

        @Override
        public void voice () {
                System.out.println("GAV-GAV");
        }
}
public class MainClass {
        public static void main(String[] args) {
                Animal animal = new Animal();
                Dog dog = new Dog();

                animal.sleep();
                dog.sleep();
                dog.voice();
        }
}
```

**Инкапсуляция** - это сокрытие реализации класса и отделение его внутреннего представления от внешнего (интерфейса). При использовании объектно-ориентированного подхода не принято использовать прямой доступ к свойствам какого-либо класса из методов других классов. Для доступа к свойствам класса принято использовать специальные методы этого класса для получения и изменения его свойств.

Открытые члены класса составляют внешний интерфейс объекта. Эта та функциональность, которая доступна другим классам. Закрытыми обычно объявляются все свойства класса, а так же вспомогательные методы, которые являются деталями реализации и от которых не должны зависеть другие части системы.

Благодаря сокрытию реализации за внешним интерфейсом класса можно менять внутреннюю логику отдельного класса, не меняя код остальных компонентов системы. Например:

```
class Student {

//объявление переменной класса как private

private String name;

// предоставление публичных методов (геттеров и сеттеров), чтобы изменять и
просматривать значения переменной name.

public String getName(){  return name;  }

public void setName(String name){  this.name=name;}

}

class Example {

public static void main(String[] args){

Student alexey=new Student ();

//  Обращение через метод

alexey.setName("Alex");

}

}
```

**Абстракция** - это выделение общих характеристик объекта, исключая набор незначительных. С помощью принципа абстракции данных, данные преобразуются в объекты. Абстракция отделяет реализацию объектов от их деталей.

**Полиморфизм** - это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных.

Концепцией полиморфизма является идея один интерфейс, множество методов. Это означает, что можно создать общий интерфейс для группы близких по смыслу действий. Например:

```
// Human is an abstract concept

interface Human {
...
    void gender ();
}

public static class Male implements Human{
...
    @Override
    public void gender (){
        System.out.println("Male");
    }
}
public static class Female implements Human{
...
    @Override
    public void gender (){
        System.out.println("Female");
    }
}

public static void main(String[] args){
    ArrayList<Human> group = new ArrayList<Human>();
    group.add(new Male());
    group.add(new Female());
     ...
    for (Human person : group) person.gender();
}
```