# ЛАБОРАТОРНАЯ РАБОТА №2
# ООП В JAVA. ПОНЯТИЕ КЛАССА.

## ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ:

Цель данной лабораторной работы - изучить основные концепции объектно-ориентированного программирования, изучить понятие класса и научиться создавать классы.

## ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Язык Java - объектно-ориентированный язык программирования. В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Класс в ООП — это в чистом виде абстрактный тип данных, создаваемый программистом. С этой точки зрения объекты являются значениями данного абстрактного типа, а определение класса задаёт внутреннюю структуру значений и набор операций, которые над этими значениями могут быть выполнены. Желательность иерархии классов (а значит, наследования) вытекает из требований к повторному использованию кода — если несколько классов имеют сходное поведение, нет смысла дублировать их описание, лучше выделить общую часть в общий родительский класс, а в описании самих этих классов оставить только различающиеся элементы.

Необходимость совместного использования объектов разных классов, способных обрабатывать однотипные сообщения, требует

поддержки полиморфизма — возможности записывать разные объекты в переменные одного и того же типа. В таких условиях объект, отправляя сообщение, может не знать в точности, к какому классу относится адресат, и одни и те же сообщения, отправленные переменным одного типа, содержащим объекты разных классов, вызовут различную реакцию.

**Создание классов в Java.**

Для того чтобы создать класс в языке Java необходимо создать файл с расширением java. Имя файла должно быть таким же, как и имя создаваемого класса. В созданном файле должен описываться класс. Синтаксис написания класса:

```
<модификатор досупа> class <имя класса> {
   <тело класса>
}
```

В качестве модификатора доступа можно указать ключевое слово public или private. Если указано слово public, то класс будет доступен из других пакетов. Если указано слово private, то класс будет доступен только внутри того пакета, в котором он находится.

В теле класса можно описать методы, переменные, константы, конструкторы класса.

Конструктор - это специальный метод, который вызывается при создании нового объекта. Не всегда удобно инициализировать все переменные класса при создании его экземпляра. Иногда проще, чтобы какие-то значения были бы созданы по умолчанию при создании объекта. По сути конструктор нужен для автоматической инициализации переменных.

Конструктор инициализирует объект непосредственно во время создания. Имя конструктора совпадает с именем класса, включая регистр, а по синтаксису конструктор похож на метод без возвращаемого значения.

В отличие от метода, конструктор никогда ничего не возвращает.

Пример класса, описывающего прямоугольник с высотой height и шириной width.

```
public class Rectangle {
   //Свойства, поля класса
   private float width;
```

```
    private float height;
    //Конструктор класса
    public Rectangle(float w, float h) {
        width=w;
        height=h;
    }

    //Метод, возвращающий ширину прямоугольника
    public float getWidth() {
        return width;
    }

    //Метод, возвращающий высоту прямоугольника
    public float getHeight() {
        return height;
    }

    //Метод, устанавливающий ширину прямоугольника
    public void setWidth(float w) {
        width=w;
    }

    //Метод, устанавливающий высоту прямоугольника
    public void setHeight(float h) {
        height=h;
    }

}
```

В данном примере был создан класс с одним конструктором, и методами, меняющими поля класса setWidth(), setHeight() ("сеттеры"), и возвращающие их значение getWidth(), getHeight() ("геттеры").

Если конструкторы в классе отсутствуют, то Java автоматически создает конструктор по умолчанию, который не имеет аргументов.

**Создание экземпляра класса.**

Для того чтобы создать экземпляр класса необходимо объявить переменную, тип которой соответствует имени класса или

суперкласса. После чего нужно присвоить этой переменной значение, вызвав конструктор создаваемого класса с помощью оператора new. Например, можно создать экземпляр класса Rectangle следующим образом:

```
Rectangle rect = new Rectangle(20, 10);
```

После этого можно вызывать методы этого класса для объекта rect, указав имя метода через точку:

```
rect.setWidth(20);
System.out.println("Новая ширина: "+rect.getWidth());
```

## ВАРИАНТЫ ЗАДАНИЙ

1. Создать класс, описывающий модель окружности (Circle). В классе должны быть описаны нужные свойства окружности и методы для получения, изменения этих свойств. Протестировать работу класса в классе CircleTest, содержащим метод статический main(String[] args).

2. Создать класс, описывающий тело человека(Human). Для описания каждой части тела создать отдельные классы(Head, Leg, Hand). Описать необходимые свойства и методы для каждого класса. Протестировать работу класса Human.

3. Создать класс, описывающий книгу (Book). В классе должны быть описаны нужные свойства книги(автор, название, год написания и т. д.)и методы для получения, изменения этих свойств. Протестировать работу класса в классе BookTest, содержащим метод статический main(String[] args).