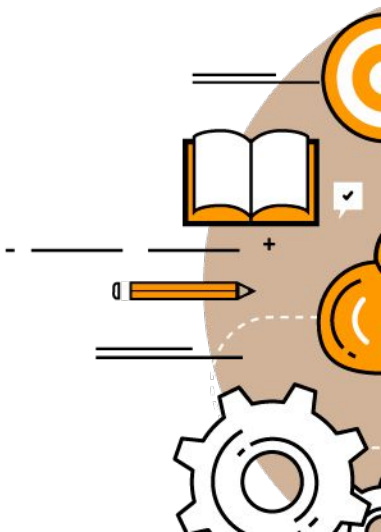
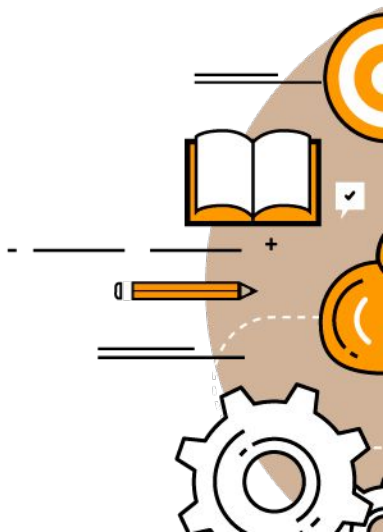
Основы методологии



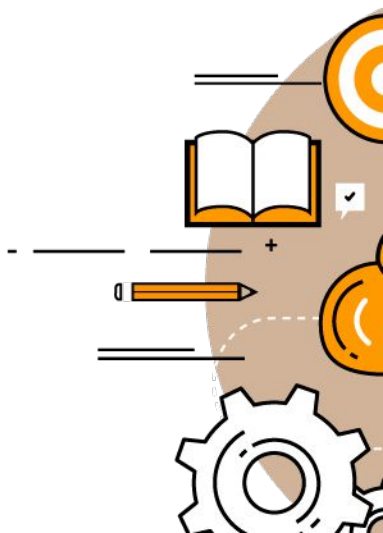
Контрольная

Принципы ООП

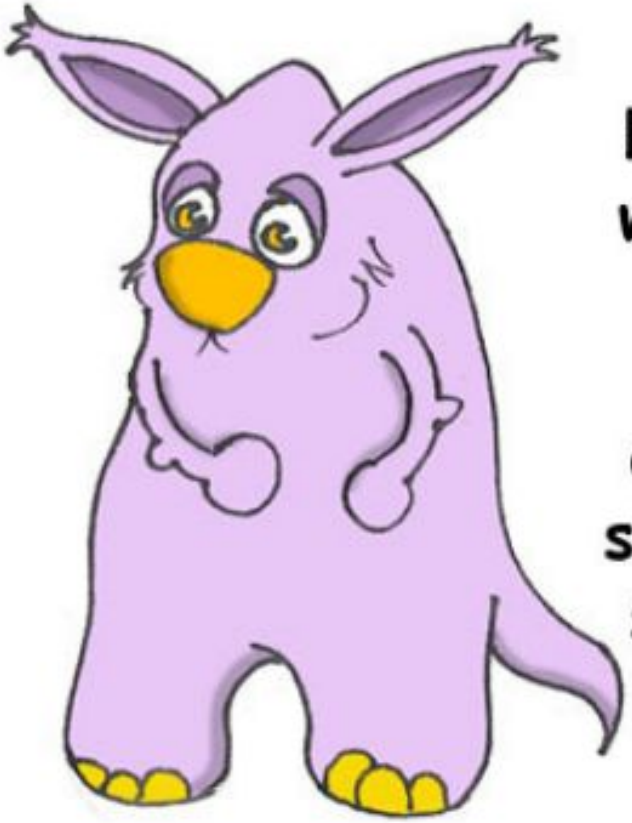
- Наследование
- Полиморфизм
- Инкапсуляция



- **Абстра́кция** в объектно-ориентированном программировании — это придание объекту характеристик, которые чётко определяют его концептуальные границы, отличая от всех других объектов.

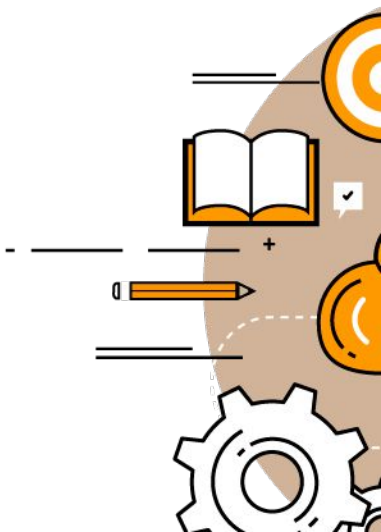


Давайте создадим вот такое существо из реального мира)

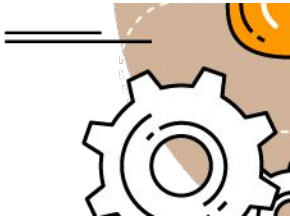


age
height
weight
color

eat( )
sleep( )
say( )

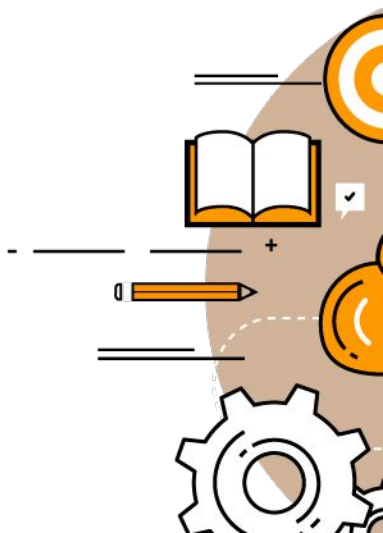


```java
public class Pet {
    int age;
    float weight;
    float height;
    String color;
    public void sleep(){
        System.out.println("Спокойной ночи! До завтра");
    }
    public void eat(){
        System.out.println("Я очень голоден, давайте перекусим чипсами!");
    }
    public String say(String aWord){
        String petResponse = "Ну ладно!! " +aWord;
        return petResponse;
    }
}
```



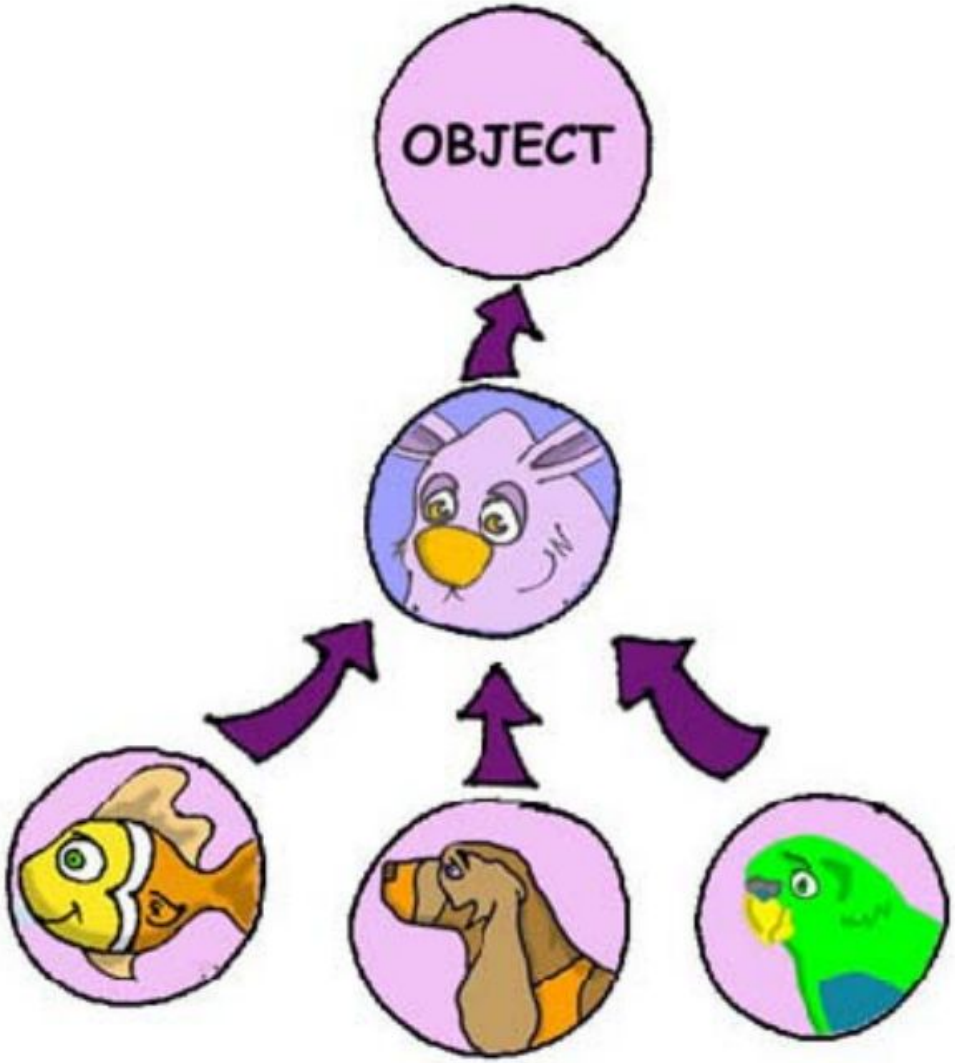
Сигнатура и возвращаемы значения

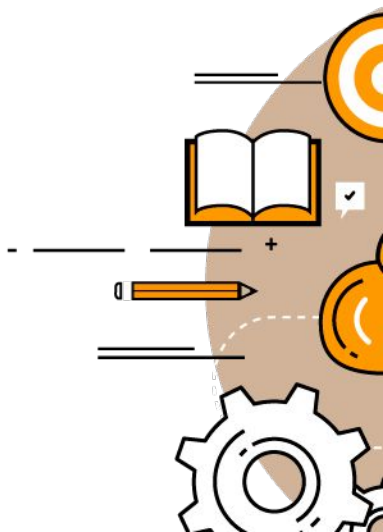
```
public void sleep()

public String say(String aWord)
```



```
2
3  public class NewApp {
4      public static void main(String[] args) {
5          String petReaction;
6          Pet myPet = new Pet();
7          myPet.eat();
8          petReaction = myPet.say("Чик!! Чирик!!");
9          System.out.println(petReaction);
10         myPet.sleep();
11     }
12
13 }
14
```

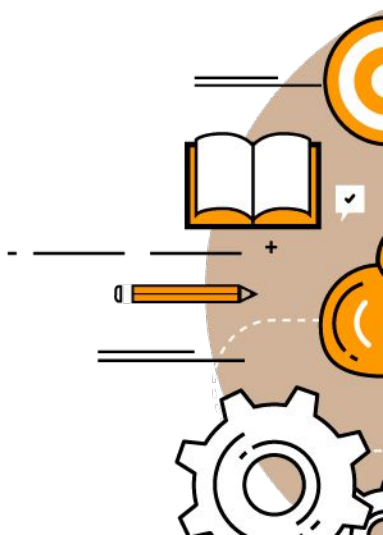
Наследование – Рыбка Тоже Домашнее Животное





Наследуемся от питомца

```
2
3  public class Fish extends Pet{
4
5  }
6
```



Вызов метода класса родителя

```
2
3  public class NewApp {
4      public static void main(String[] args) {
5          Fish myLittleFish = new Fish();
6          myLittleFish.sleep();
7      }
8
9  }
10
```

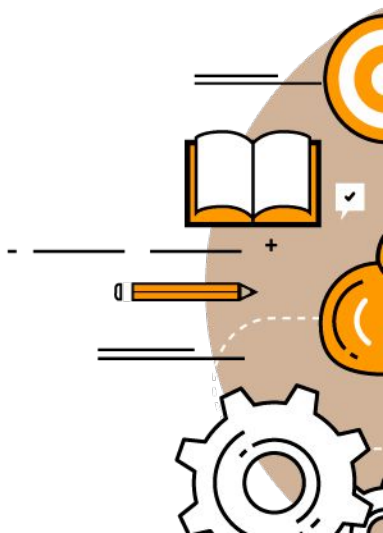
Problems @ Javadoc Declaration Console

<terminated> NewApp [Java Application] C:\Program Files\Java\jre1.8.0_151\bi

Спокойной ночи! До завтра

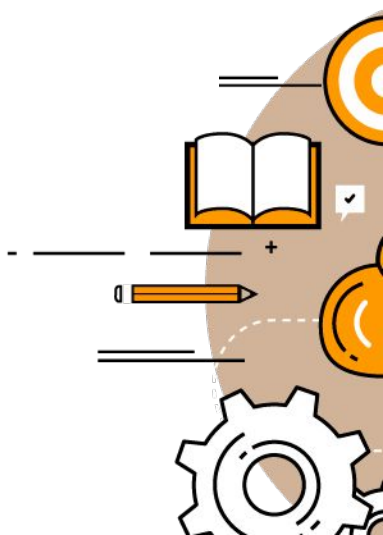
Наследование рыбы от питомца

```
2
3  public class Fish extends Pet{
4      int currentDepth=0;
5      public int dive(int howDeep)
6      {
7          currentDepth=currentDepth + howDeep;
8          System.out.println("Ныряю на глубину " + howDeep + " футов");
9          System.out.println("Я на глубине " + currentDepth + " футов ниже уровня моря");
10         return currentDepth;
11     }
12 }
13
```



Вызов собственных методов рыби и методов питомца

```
3  public class NewApp {
4      public static void main(String[] args) {
5          Fish myFish = new Fish();
6          myFish.dive(2);
7          myFish.dive(3);
8          myFish.sleep();
9      }
10 }
11
```

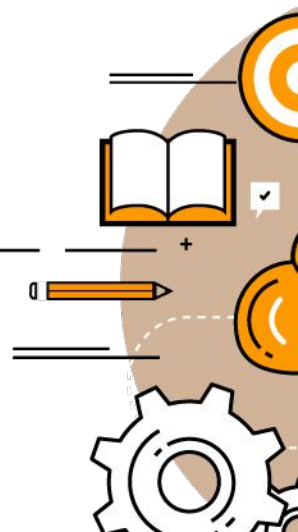


```
public class NewApp {
    public static void main(String[] args) {
        Fish myFish = new Fish();
        System.out.println(myFish.say("Привет"));
    }
}
```

Problems @ Javadoc Declaration Console

<terminated> NewApp [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\j

```
Ну ладно!! Привет
```



- **Полиморфизм** (polymorphism) (от греческого polymorphos) - 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.

Полиморфизм в действии

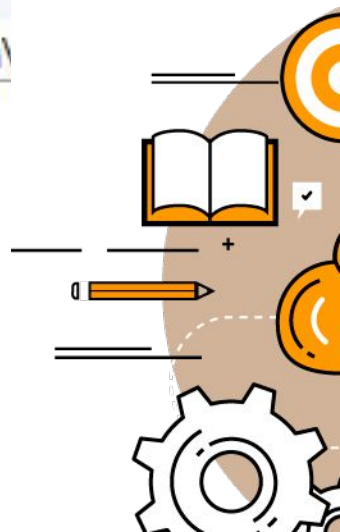
```
2
3  public class Fish extends Pet{
4      int currentDepth=0;
5      public int dive(int howDeep)
6      {
7          currentDepth=currentDepth + howDeep;
8          System.out.println("Ныряю на глубину " + howDeep + " футов");
9          System.out.println("Я на глубине " + currentDepth + " футов ниже уровня моря");
10         return currentDepth;
11     }
12     public String say(String something)
13     {
14         return "Ты чё не знаешь, что рыбы не разговаривают?";
15     }
16 }
17
```

```
2
3  public class NewApp {
4      public static void main(String[] args) {
5          Fish myFish = new Fish();
6          System.out.println(myFish.say("Привет"));
7      }
8  }
9
```

Problems @ Javadoc Declaration Console

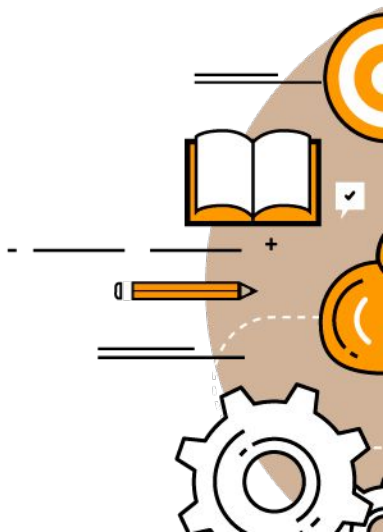
<terminated> NewApp [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\

```
Ты чё не знаешь, что рыбы не разговаривают?
```



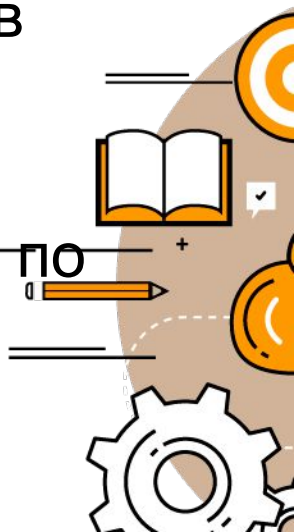
Что делает final?

```
final public void sleep(){…}
```



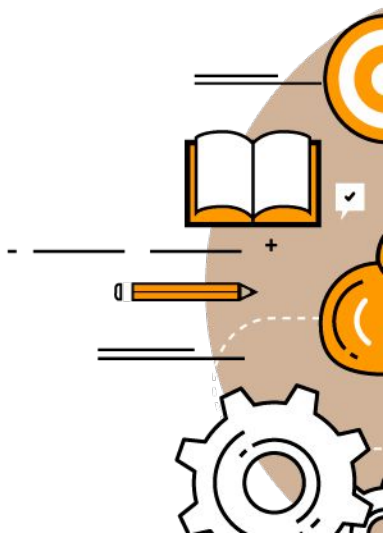
Модификаторы доступа

- public: публичный, общедоступный класс или член класса. Поля и методы, объявленные с модификатором public, видны другим классам из текущего пакета и из внешних пакетов.
- private: закрытый класс или член класса, противоположность модификатору public. Закрытый класс или член класса доступен только из кода в том же классе.
- protected: такой класс или член класса доступен из любого места в текущем классе или пакете или в производных классах, даже если они находятся в других пакетах
- Модификатор по умолчанию. Отсутствие модификатора у поля или метода класса предполагает применение к нему модификатора по умолчанию. Такие поля или методы видны всем классам в текущем пакете.

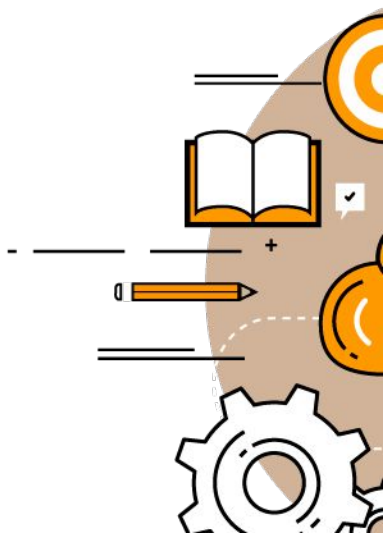


Давайте попробуем в действии модификаторы доступа

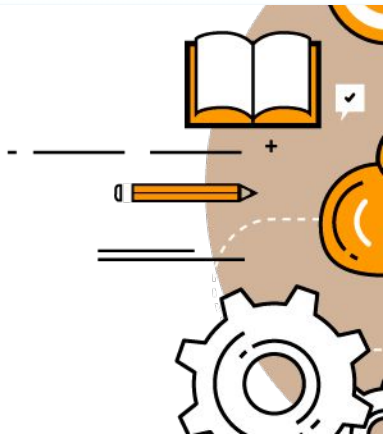
```
public class Fish extends Pet{
    int currentDepth=0;
    public int dive(int howDeep)
    {
        currentDepth=currentDepth + howDeep;
        System.out.println("Ныряю на глубину " + howDeep + " футов");
        System.out.println("Я на глубине " + currentDepth + " футов ниже уровня моря");
        return currentDepth;
    }
    public String say(String something)
    {
      return "Ты чё не знаешь, что рыбы не разговаривают?";
    }
}
```



- Казалось бы, почему бы не объявить все переменные и методы с модификатором public? Однако использование различных модификаторов гарантирует, что данные не будут искажены или изменены не надлежащим образом. Подобное сокрытие данных называется **инкапсуляцией**.



```java
import java.util.Scanner;

public class NewApp {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Введите имя: ");
        String name = in.nextLine();
        System.out.print("Введите возраст: ");
        int age = in.nextInt();
        System.out.println("Ваше имя: " + name + "    Ваш возраст: " + age);
    }

}
```

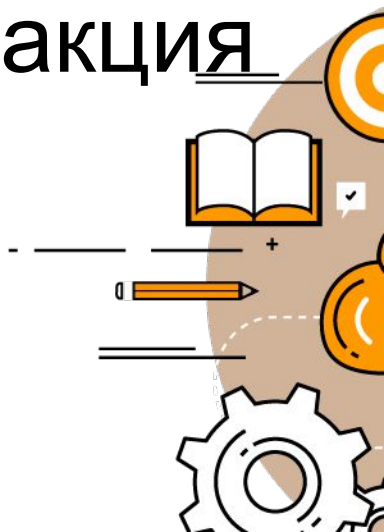


Принципы ООП

- Наследование
- Полиморфизм
- Инкапсуляция

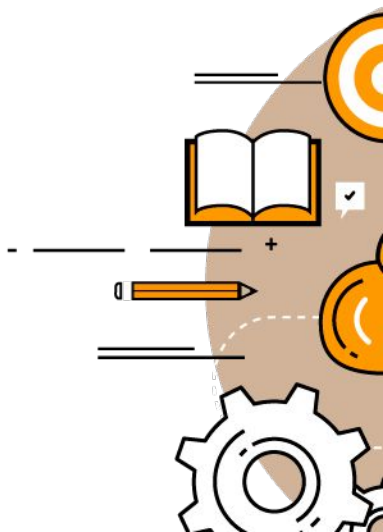
+

Абстракция



Классы и объекты

- Автомобиль
- Компьютер
- Телефон
- Часы
- Посуда
- Игра
- Магазин
- Приложение
- Мебель
- Одежда
- Бытовая техника
- Игровой персонаж
- Футболист
- Напиток
- Игрушка
- Предприятие



Домашняя работа