- jdk8特性，可以避免内部类过多，简化代码，属于函数式编程概念
- 正常调用：一个类实现接口，重写该接口方法，创建该实现类实例，调用该实例的重写方法
- 但是为了一个实现类就创建一个单独的class会不会有点耗资源呢，试着优化一下
- class有多种
    - 普通class：和public的class同级放在Java文件中，不能有public修饰符
    - 静态内部类：在别的class中，不处于方法中，用static修饰符修饰
    - 局部内部类：在别的class的方法中，没有public和static修饰符
    - 匿名内部类：只能借助接口或父类存在，生成接口或父类实例时，直接编写类体，例如有Car接口，Car c = new Car(){public void say(String str){System.out.println(str);}};，这叫匿名内部类，这个类是Car接口的实现类
    - Lamda表达式：用于简化匿名内部类，Car c = (String str)->{System.out.println(str)};
        - 继续简化：
            - String str的String可以去掉
            - 只有一个方法参数可以去掉()
            - 只有一条代码可以去掉{}
        - 适用场景：函数式接口
            - 一个有且只有一个抽象方法的接口Car接口，可以有多个非抽象方法
            - 或者接口只有一个方法，可抽象可不抽象

        - 抽象方法public abstract void say(String str);
        - 有些方法的参数就是一个接口对象，方法体中调用该接口的抽象方法，使用Lamda表达式可以大大简化代码
    - Lamda表达式会自动将等式后半部分识别为等式前半部分的接口类型，为该接口创建实现类
    - 大括号内容当做该实现类重写接口抽象方法say的自身say方法的方法体
    - 通过c.say("我是车")调用

- Thread类的构造参数是Runnable接口实现类，run方法其实就是一个抽象方法，所有实例化Thread时可以使用Lamda表达式