个参数的Lambda表达式
(String s) -> System.out.println(s)

// 带两个参数的Lambda表达式
(int a, int b) -> a + b

```

2.5示例 3：带返回值的Lambda表达式
----------

```
// Lambda表达式的主体是一个表达式，直接返回结果
(int a, int b) -> a * b

// Lambda表达式的主体是一个代码块，可以包含多条语句
(int a, int b) -> {
    if (a > b)
        return a;
    else
        return b;
}

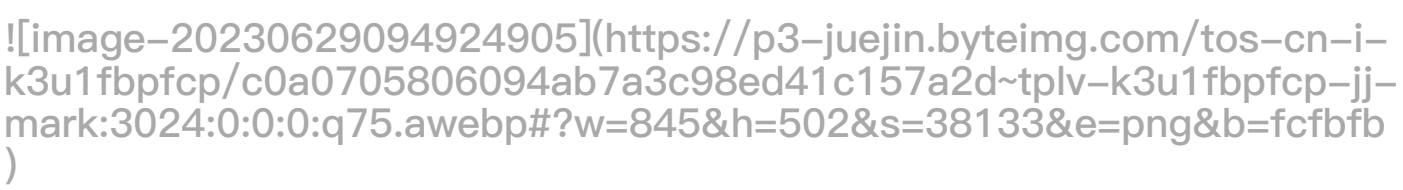
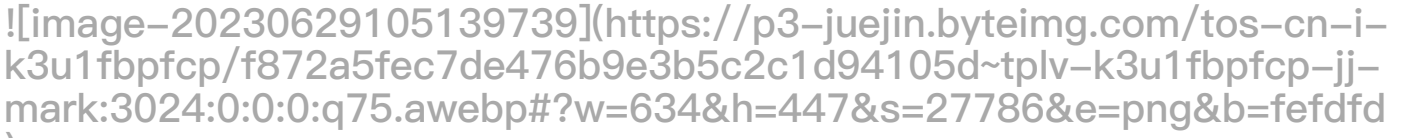
```

三、示例
====

3.1Runnable接口案例
----------

### **传统方式**
```

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class Lamada01ApplicationTests {

    @Test
    void contextLoads() {
    }

    @Test
    public void test01(){
        Runnable r1=new Runnable() {
            @Override
            public void run() {
                System.out.println("好好学习，天天向上");
            }
        };

        r1.run();
    }
}
```

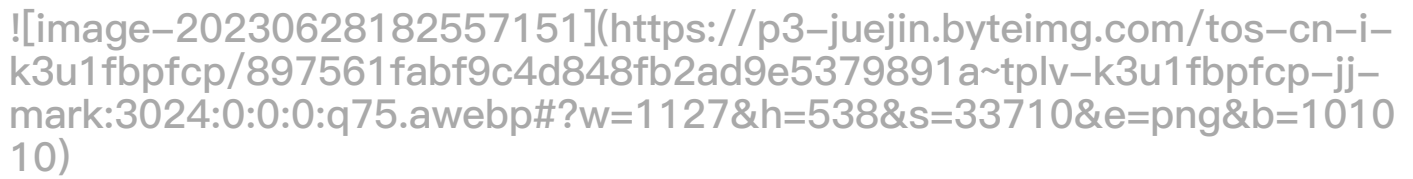

lambda方式

```
    @Test
    public void test02(){
        Runnable r2=()-> System.out.println("北京天安门");
```

```
        r2.run();
    }
```

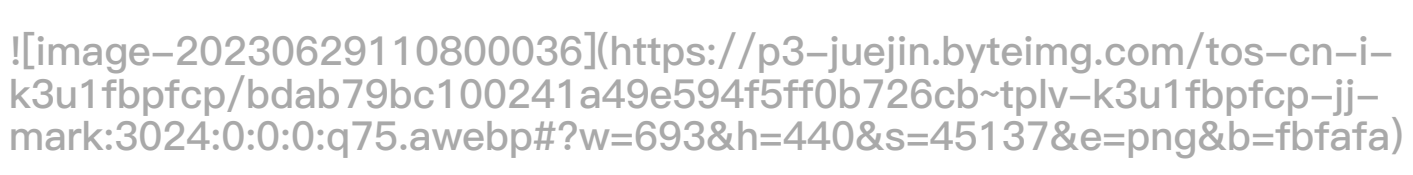
```



### **分析**



3.2比较大小案例
---

### **传统方式**

```
    @Test
    public void test03(){
        Comparator<Integer> comparator=new Com   });
```
```

```
        System.out.println(names);

    }
}
```

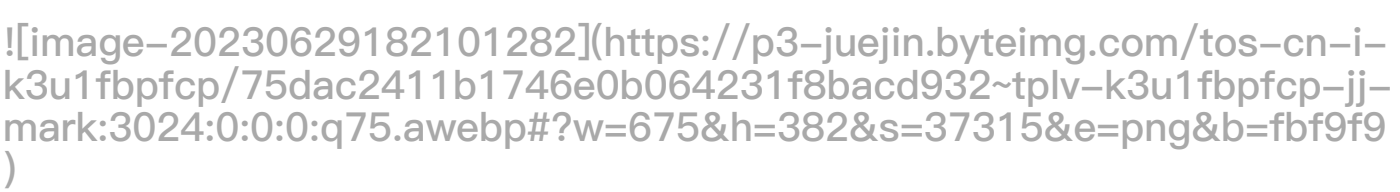


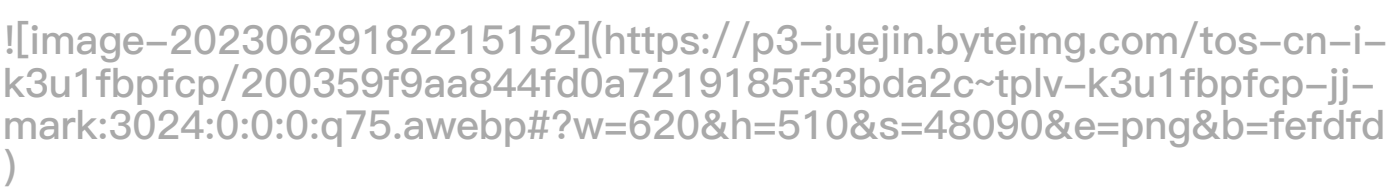
分析

```
    @Test
    public void Test07(){
        List<String> names= Arrays.asList("peter", "anna", "mike",
"xenia");
        Collections.sort(names, new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                return o1.compareTo(o2);
            }
        });

        System.out.println(names);
    }
```



```
    @Test
    public void Test08(){
        List<String> names= Arrays.asList(”peter”, ”anna”, ”mike”,
”xenia”);
        Collections.sort(names, (String o1,String o2)->{
            return o1.compareTo(o2);
        } );

        System.out.println(names);
    }
```





3.5使用 Lambda 表达式与 Streams API

Java 8 的 Streams API 允许你以声明性方式处理集合。以下是使用 Lambda 表达式过滤和处理集合的示例：

```
List<String> filteredNames = names.stream()  // 创建 Stream
    .filter(name -> name.length() > 4)         // 使用 Lambda 表达式过滤
    .collect(Collectors.toList());               // 收集结果
System.out.println(filteredNames); // 输出 [Peter, Kim]
```

```
```