idea快捷键

Ctrl+Alt+B:直接跳到实现类的方法：

Ctrl+B：跳转到变量声明的地方

Alt+f8 :打开计算表达式窗口

Ctrl+Alt+f8:直接计算表达式结果

Ctrl + Shift + U:大小写转换

Alt+up/down:切换至上/下一个方法

Alt+left/right:切换左边/右边的选项卡

Ctrl+数字面板的+-：展开/折叠当前函数

Alt+Shift+up/down:向上/下 移动代码行块

Ctrl+Shift+/: 注释代码块

Ctrl+f12：查看文件结构，类+方法

函数式接口

怎么定义一个函数式接口

@FunctionalInterface注解，并且只有一个抽象方法

```
@FunctionalInterface
public interface Consumer<T> {
    void accept(T t);
    default Consumer<T> andThen(Consumer<? super T> after) {
        Objects.requireNonNull(after);
        return (T t) -> { accept(t); after.accept(t); };
    }
}
```

常见的函数式接口

Function 接口

Function 这个单词的意思就有「函数」的意思，就数学中的 y = f(x)，接收一个 x 参数，通过函数 f 运算后，返回一个结果 y。

`Function` 接口包含四个方法：

- `apply(T t)`：这是 `Function` 接口的主要方法，它接收一个参数并返回一个结果。同时它也是唯一的抽象的方法，剩下的都是有默认实现的（Java 8 中接口的抽象方法支持默认实现）。
- `andThen(Function after)`：作用是将两个 `Function` 组合。首先执行当前函数，再执行 `andThen` 函数，并将当前函数的结果作为参数传递给 `andThen` 函数。
- `compose(Function before)`：同理，将两个 `Function` 组合，将先执行 `compose` 函数，再执行当前函数，并将 `compose` 函数的结果作为参数传递给当前函数。

- `identity()`：返回一个执行恒等转换的函数，即返回输入参数本身。

Function 接口通常用于将一个类型的值转换为另一个类型的值

```
//apply
// Function 接口的泛型，第一个参数是入参类型，第二个参数是出参类型
// Function 接口只有一个抽象方法，就是 apply()，下面利用 Lambda 表达式实现这个抽象方法并创建 Function 对象
Function<Integer, String> function = num -> "GTA" + num;
// 将5这个参数传递给function，得到返回结果
String result = function.apply(5);
System.out.println(result); // 打印：GTA5
//andThen 和 compose 方法
//定义两个Function对象进行相关转换操作
Function<String, String> upperCase = s -> s.toUpperCase();
Function<String, String> addPostfix = s -> 5 + "5";
//链式调用，将 gta这个字符串参数先传递upperCase这个函数进行操作，然后将得到的结果传递给 addPostfix函数进行操作，得到返回结果
String str = upperCase.andThen( addPostfix).apply("gta");
System.out.println(str); //打印:GTA5
//identify 方法
//identity 方法返回一个执行恒等转换的函数，该函数将输入参数原样返回。例如：
Function<String,String> identity = Function.identity();
String result = identity.apply("hello");
// result is "hello"
```

```
// 两个参数的情形
BiFunction<Integer, Integer, Integer> add = (x, y) -> x + y;
System.out.println(add.apply(10, 10));
```

消费型Consumer:有参数， 没有返回值

```
Consumer<String> c1 = (s)->{System.out.println(s);};
Consumer<String> c2 = (String s)->{System.out.println(s.length());};
c1.andThen(c2).accept("hello world");
```

供应型Supplier:没有参数，有返回值

```
Supplier<Integer> s1 = ()->{
    return new Random().nextInt(2);
};
Integer integer = s1.get();
System.out.println(integer);
```

功能型Function:有参数，有返回值

```
Function<String, Integer> f1 = (String s)->{return Integer.valueOf(s);};
Integer integer1 = f1.apply("123");
System.out.println(integer1);
```

断言型Predicate:有参数，有返回值，返回类型bool

```
Predicate<String> p1 = (s)->{return s.contains("123");};
boolean b = p1.test("abcd12345");
System.out.println(b);
```

集合常用方法

Stream以及转换

```
public class MyCollections {
    public static void arrayListMap(){
        /**
         * 数组转集合，必须是包装类才能asList
         */
        int[] nums = new int[]{9,4,3,2,5};
        Integer[] nums1 = new Integer[]{1,2,3,4,5};
        List<Integer> integers =
Arrays.stream(nums).boxed().collect(Collectors.toList());//int数组转list
        List<Integer> integers1 = Arrays.asList(nums1);//Integer数组转list

        /**
         * List->Array
         */
        Integer[] integers2 = integers.toArray(new Integer[integers.size()]);

        /**
         * List->排序
         */
        List<Integer> sorted = integers.stream().sorted((e1, e2) -> {
            return e1 > e2 ? -1 : 1;
        }).collect(Collectors.toList());//[9, 5, 4, 3, 2]

        /**
         * Array->排序
         */
        Arrays.sort(integers2,(e1, e2) -> {return e1 > e2 ? -1 : 1;});//[9, 5, 4, 3, 2]

        /**
         * map  filter
         * 构造方法的引用写法是类名::new，因此，此处传入Person::new。
         * <R> Stream<R> map(Function<? super T, ? extends R> mapper);
         * Stream<T> filter(Predicate<? super T> predicate);
         */
        List<String> names = new ArrayList<>();
        names.add("小明");
        names.add("小黑");
        names.add("小花");
```

```
        List<Person> personList =
names.stream().map(Person::new).collect(Collectors.toList());
        personList.forEach(e->{System.out.println(e);});
        List<Integer> collect = names.stream().map((name) -> {return
name.length();}).collect(Collectors.toList());
        System.out.println("collect:"+collect);
        List<Person> list1 = personList.stream().filter((person) -> {return
person.name.contains("明");}).collect(Collectors.toList());
    }
    public static void main(String[] args) {
        arrayListMap();
    }
}
class Person{
    public String name;
    Person(String name){
        this.name=name;
    }
    public void show(String name){
        System.out.println(name);
    }
}
```

Collections

```
Integer max = Collections.max(collect);
Collections.reverse(collect);
Collections.sort(collect);
Collections.sort(collect, (i1,i2)->{
    return i1<i2?1:-1;//i1左边的，i2右边的  return condition:1(交换) -1(不交换)
});
Collections.shuffle(collect, new Random(112));
```

Arrays

```
Arrays.sort(nums);
Arrays.stream(nums).boxed().collect(Collectors.toList());
Arrays.stream(nums).max();
List<Integer> collect = Arrays.stream(nums).filter(e -> {
    return e > 0;
}).boxed().collect(Collectors.toList());
Arrays.stream(nums).distinct();
```

```
java -verbose:class -jar  uscmpub-business-gscm-daily-RELEASE.jar
```

java -verbose:class -jar <your_jar_file> 这个命令的作用是在运行Java应用程序时打印出类加载的详细信息。具体来说：

- java 是用于启动Java虚拟机（JVM）的命令。
- -verbose:class 是一个JVM的启动参数，用于打印类加载信息。

- -jar <your_jar_file> 指定了要运行的Java应用程序的jar文件。

当你执行这个命令时，JVM会在标准输出中打印出每个类的加载情况，包括类的加载来源和加载结果。这对于调试和排查类加载问题非常有帮助，可以帮助你了解到底哪些类被加载了，以及这些类是从哪些jar包中加载的。

Error: Unable to access jarfile

```
vm.getGridModel('contractMaterialList').getEditRowModel().get("wbs_name").on('beforeBrowse'
,function (data) {
                        var condition = {
                            isExtend: true,
                            simpleVOs: [],
                        };
                        condition.simpleVOs.push({
                            field: "orgId",
                            op: "eq",
                            value1: viewmodel.get('orgId').getData(),
                        });
                        this.setFilter(condition);
                    })
```

SEM rush Guru版本(SEO 优化工具， 可查看Google ads 关键词文案和投放策略)
登录地址：taobao-seo.com
账号glimerajalam@hotmail.com
密码050505

```
1.nmietye709jxd@gmail.com--KQ3d65Lrv  已封
2.smolkawu@gmail.com---Wufws@.q  plus版
3.garnera.alice.z3462@gmail.com--,NZGyX%r%/NCn6y
4.fallynwilsoncampbell@gmail.com--ARWDJYFC.SIUF
```