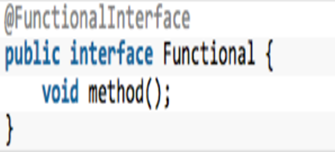
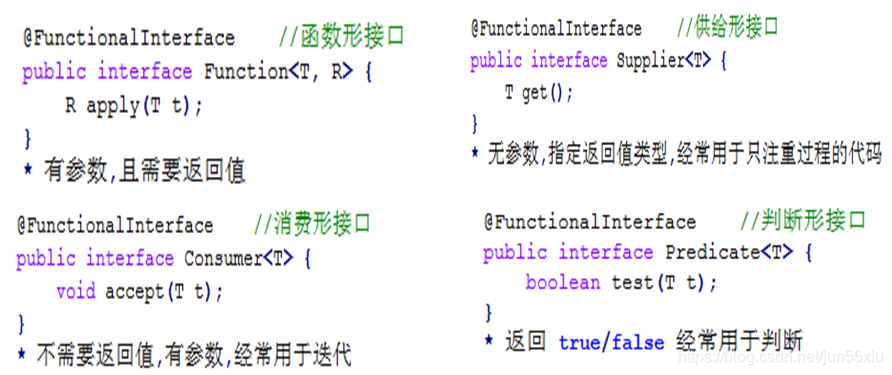
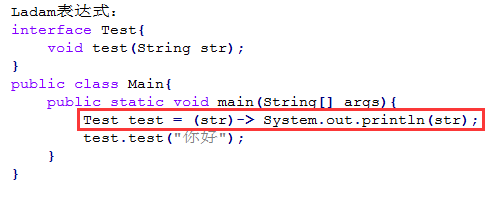
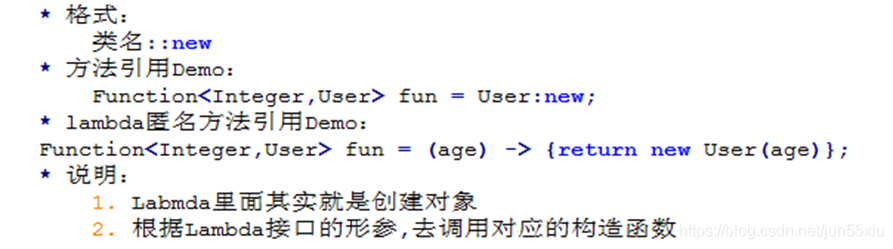
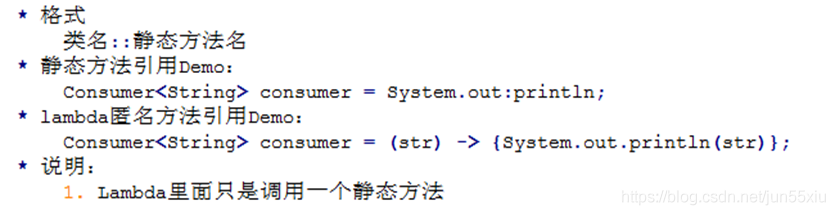
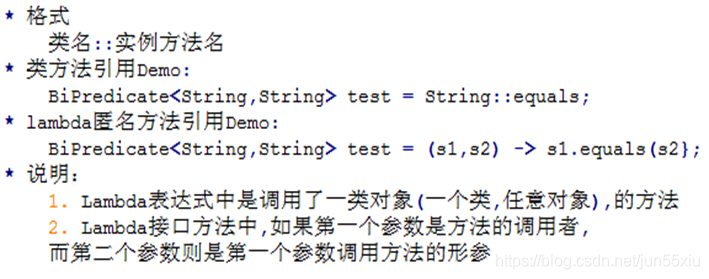
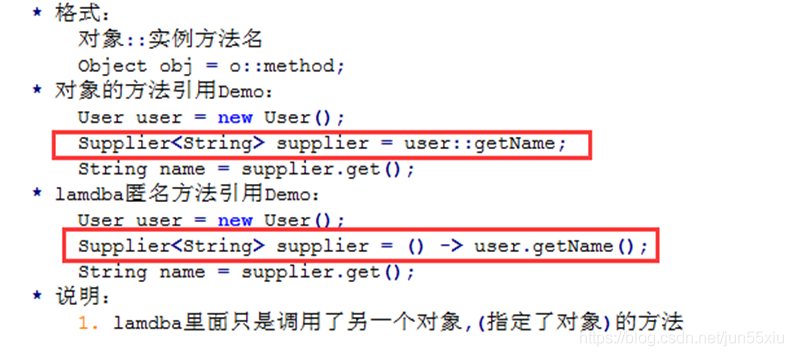
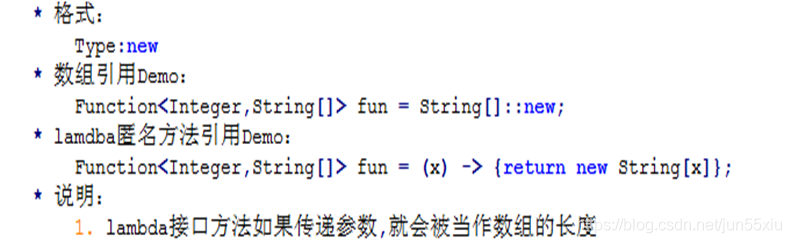
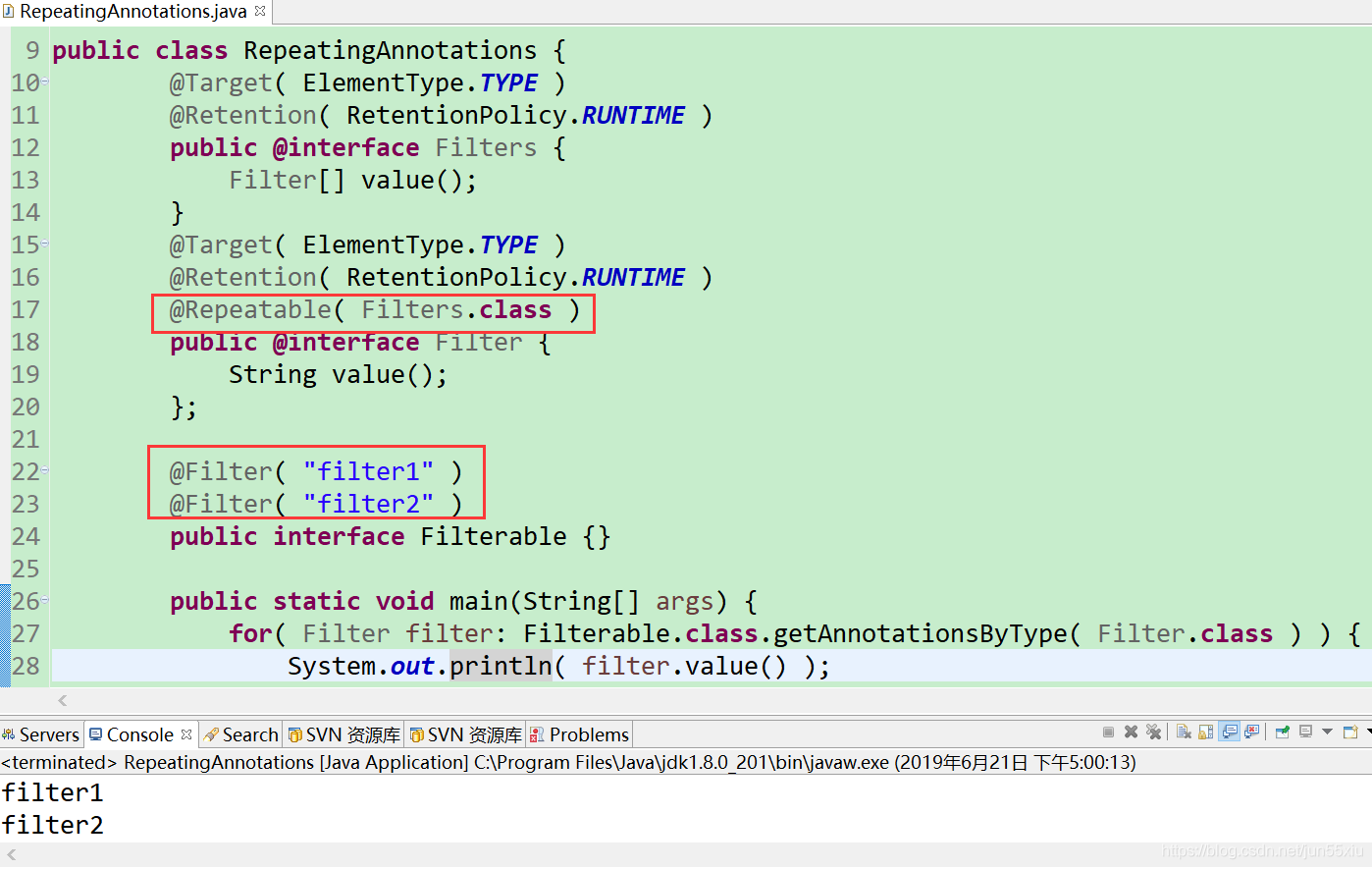
JDK8新特性

* **Lambda 表达式** − Lambda 允许把函数作为一个方法的参数（函数作为参数传递到方法中）。
* **方法引用** − 方法引用提供了非常有用的语法，可以直接引用已有Java类或对象（实例）的方法或构造器。与lambda联合使用，方法引用可以使语言的构造更紧凑简洁，减少冗余代码。
* **默认方法** − 默认方法就是一个在接口里面有了一个实现的方法。
* **新工具** − 新的编译工具，如：Nashorn引擎 jjs、 类依赖分析器jdeps。
* **Stream API** −新添加的Stream API（java.util.stream） 把真正的函数式编程风格引入到Java中。
* **Date Time API** − 加强对日期与时间的处理。
* **Optional 类** − Optional 类已经成为 Java 8 类库的一部分，用来解决空指针异常。
* **Nashorn, JavaScript 引擎** − Java 8提供了一个新的Nashorn javascript引擎，它允许我们在JVM上运行特定的javascript应用。
* 1.1接口新增默认方法与静态方法
* 1.1.1  Interface Default Method：For creating a default method in java interface, we need to use “**default**” keyword with the method signature. For example:
* package com.java8.defaultmethod;
* public interface Interface1 {
* void method1(String str);
* //a default method
* default void log(String str){
* System.out.println("I1 logging::"+str);
* }
* }
* *Notes:类继承多个Interface接口同名方法(如show())时，必须在子类中@Override重写父类show()方法。*
* 1.1.2 Interface Static Method：interface static method is similar to default method except that we can’t override them in the implementation classes.for example:
* package com.java8.staticmethod;
* public interface MyData {
* static boolean isNull(String str) {
* System.out.println("Interface Null Check");
* return str == null ? true : "".equals(str) ? true : false;
* }
* }
* 1.1.2.1 Functional Interfaces:含有一个显式声明函数（抽象方法）的接口称为函数接口，注释@FunctionalInterface用作检查代码块,包package java.util.function，通常使用lambda expressions来实体化函数接口，for example：
* 
* **特性说明：**
* 1,函数式接口仅仅只有一个方法(非默认或静态方法)，用于显示转换成ladbma表达式。
* 2, java.lang.Runnable接口 java.util.concurrent.Callable接口是两个最典型的函数式接口。
* 3.如果一个函数式接口添加一个普通方法，就变成了非函数式接口（一般定义的接口）。
* 4.Jdk8 规范里添加了注解@FunctionalInterface来限制函数式接口不能修改为普通的接口.
* **新增了四个重要的函数式接口：函数形接口 、供给形接口、消费型接口、判断型接口**
* 
* 1.2 新增Lambda表达式
* 1.2.1 Lambda表达式（基于函数的匿名表达式）
* 
* 语法：( object str,....)[参数列表]   ->[箭头符号]     代码块或表达式
* **特性：**Lambda 的类型是从使用 Lambda 的上下文推断出来的。上下文中 Lambda 表达式需要的类型称为目标类型（一个 Lambda表达式所在的类的类型。并且必须存在一个目标类型）;  匿名、函数、传递、简洁。
* 1.3 新增方法引用
* 1)**构造器引用**
* 
* **2)静态方法引用**
* 
* **3)类(任意对象)的方法引用**
* 
* **4)实例对象的方法引用**
* 
* **5)数组引用**
* 
* 1.4 重复注解
* Java 5引入了[注解](http://www.javacodegeeks.com/2012/08/java-annotations-explored-explained.html)机制，这一特性就变得非常流行并且广为使用。然而，使用注解的一个限制是相同的注解在同一位置只能声明一次，不能声明多次。Java 8打破了这条规则，引入了重复注解机制，这样相同的注解可以在同一地方声明多次。重复注解机制本身必须用@Repeatable注解.
* 
* 1.5 扩展注解的支持
* jdk 8扩展了注解的上下文。现在几乎可以为任何东西添加注解：局部变量、泛型类、父类与接口的实现，就连方法的异常也能添加注解。
* 