P2

咱们开始吧，这个课程的PPT页数挺多的，报的是六个学时，我争取呢今天能够把PPT上的内容都讲解完，这样所有人的学时就应该都能达到考核要求了。

首先呢看一下目录：总共包括8个大的方面，第一个是新特性简介，主要是说一下Java8到**底有哪些方面的优势，速度更快了，代码更简洁等**

第二个是**Lambda表达式，这是Java8的基础，像函数式接口Stream等都是基于Lambda才能够实现和使用**。

函数式接口:**Lambda只能在函数式接口的地方才能够被使用**，

方法引用：**对Lambda表达式的再一次简化，用更简单易懂的方式来进行数据的处理**

Steam API:这个是重点，Stream即流的意思，我们可以讲数据集合例如List，Set数组能转换成流，在流的基础上进行筛选、排序、分组等

接口中的默认方法和静态方法：Java8中对于接口又有了新的定义，接口中也可以包含方法的实现，而不是仅仅声明一个方法，方法的实现有默认方法、静态方法

新事件日期API：之前的日期时间用起来很麻烦，而且对于dateformat日期格式化是非线程安全的，所以在Java8中对时间日期进行了重新的修改

Optional：解决了我们在代码中为了避免空指针异常而进行的各种非空判断的问题。

---------------------------------------------------------------------------------------------------------------------------------

P3:

速度更快：对于数据的遍历也不需要我们编写类似for循环的操作，其内部循环进行了封装。

代码更少：类似需要使用匿名类的地方，不需要再编写一些不需要的代码。

Stream:可以用类似编写sql语句的思路来对数据集合进行操作

便于并行：不需要我们进行显示的创建线程，其内部就能帮助我们实现多核的并行处理。

P5：我们通过一个例子来引申出为什么要使用Lambda

假设从一个苹果集合中筛选出绿颜色的苹果。我们很自然的会想到，定义一个方法，参数为Apple List，然后对集合进行遍历，如果条件符合就将其加入到结果列表中，随后返回。

现在看来这个方法没有什么问题。

P6：然而新的问题又来了，我们需要筛选多种颜色的苹果，那么第一个方法就不能够使用了，尝试将其抽象化，我们再定义一个方法 第一个参数为苹果类别，第二个参数为颜色，然后对列表进行遍历筛选出颜色与入参相同的苹果，最后返回。

---------------------------------------------------------------------------------------------------------------------------------

P7：例子再发生变化，区分重的苹果和轻的苹果。我们看到这个方法和上一个方法的结构完全一样，不同的只有入参和红色方框中的内容。至此为了解决筛选颜色和重量两种操作，我们定义了两种方法，编写了很多的重复的代码，如果需要修改遍历方式来提升性能，那就得修改所有的实现方法，而不是只改一个。

---------------------------------------------------------------------------------------------------------------------------------

P8：所以再次进行抽象，函数的参数列表变成了颜色、重量、以及flag参数，当flag为真的时候按照颜色进行筛选，当flag为假的时候按照重量进行筛选，这种实现有很多缺点

1：如果再增加其他筛选条件，参数列表中还会继续增加其他参数

2：flag标志的作用变的不直观，不清楚什么时候用true，什么时候用false

---------------------------------------------------------------------------------------------------------------------------------

P9:第五次尝试：行为参数化，首先定义一个ApplePredicate的接口，其中声明一个test方法，然后再创建AppleGreenColorPredicate和AppleHeavyWeightPredicate来具体实现test方法，其中这一个代表筛选绿色苹果的策略，第二个代表筛选重量的策略

---------------------------------------------------------------------------------------------------------------------------------

P10：将接口ApplePredicate作为参数传递个filterApples方法，这段代码比第一次尝试灵活多了，读起来也容易。

P11：这段代码比我们第一次尝试的时候灵活多了，读起来、用起来也更容易！现在你可以创建不同的ApplePredicate对象，并将它们传递给filterApples方法。如果农民让你找出所有重量超过150克的红苹果，你只需要创建一个类来实现ApplePredicate就行了。你的代码现在足够灵活，可以应对任何涉及苹果属性的需求变更了：这种我们把策略传递给filterApples这个筛选方法中，由于该filterApples方法只能接受对象，所以你必须把代码包裹在ApplePredicate对象里 其实最根本上就是传递小方框中的代码，但是为了这一小代码，我们编写了一个接口，以及实现接口的类，类中又定义了实现方法。

P12：匿名内部类不需要在单独的文件中编写类，只需在使用的方法编写实现代码即可，但是匿名内部类有诸多缺点，第一比较笨重，因为他会占用很多的内存空间，第二：不符合阅读习惯，感觉很绕

P13：更灵活，符合我们的阅读习惯，在调用方法上一句话就可以执行不同的策略。不需要再定义类。第八次尝试将更灵活，声明一个泛型

P15：再用一个我们比较常用的例子进行说明，我们对苹果列表按照重量进行排序，可以调用inventorysort方法，参数为Comparator类型的匿名类，然后再类中重写compare方法。现在我们使用Lambda进行重新编写

P16：Labmda总共有三部分内容，第一部分为参数列表，

第二部分为箭头一个短横线以及大于号

第三个是Labmda的主体，具体要调用的diamante

P17：Lambda的语法可以用下列两个公式来进行描述，参数列表一样，不同的是后面的Labmda主体部分，一中没有{}，一中带有{}，其中没有{}的表达式也不需要写；，仅有依据java代码，并且框架会自动帮在前面加上return 返回一个值，第二种带有{}的如果需要有返回值需要手动编写return语句。下面我们就用几个例子来说明

P22：以上所有的Labmda表达式编写的位置都是函数式接口，那么什么才是函数式接口呢，就是只定义了一个抽象方法的接口。在java8中，接口不仅仅能包含抽象接口，还能定义自己的默认方法和静态方法，以及方法的内部实现。后面会讲到。

P23：只有Lambda表达式的签名和函数式接口的方法的签名一致，才是有效的Lambda表达式，才能编译通过，通俗的说就是，Lambda表达式的参数列表和返回类型要与要使用的函数式接口的中的方法的参数列表和返回类型一致。

P27：FunctionalInterface不是必须的，加不加都行，但是最好加上，一是标识这是一个函数式接口，另外一个能够帮助我们及时发现编码错误。

P28：为了让我们更容易的使用一些常用的函数式接口，java8的设计师帮我们在java.util.funciotn中引入了新的函数式接口。对于一些基本情况我们不需要再定义函数式接口，直接拿来用就可以了。第一个是Predicate,我们之前自己编写的ApplePredicate和这个函数式接口一模一样

P29：这里定义了一个forEach函数参数为泛型T的list，第二个参数为Consumer类型的函数是接口，就可以用下面的这种方式进行调用，参数为Integer类型返回类型为void

P 30:我们定义了一个map方法，第一个参数为泛型为T 的list，第二个参数为Function类型的函数式接口，调用的时候传递（String s）->s.length()

我们在后面的方法中会讲到将数据集合转换成stram,然后调用stream的map,foreach，filter方法，其内部实现原理应该和这三个方法是一样的。

P31: 装箱的操作，在性能方面是要付出代价的。装箱后的值本质上就是把原始类型包裹起来，并保存在堆里。因此，装箱后的值需要更多的内存，并需要额外的内存搜索来获取被包裹的原始值。

P32 Java8定义了很多原始类型特化的接口

P36 第一条静态方法的方法引用根号理解，第二个和第三个我们主要讲解一下

我们举了几个例子来比如Apple a作为入参 Lambda的主体调用的是a.getWeight()，那就符合任意类型实例方法的方法引用。现有对象实例方法的方法引用。注意第二种（有两个）如果两个参数调换位置，则不能用此种方法引用了。

P40 谓词复合即将几个谓词通过一定的逻辑复合成一个谓词用来表示特殊的含义。

P42 1、集合是比如set，list，数组类型的数据

2、类型与使用SQL，使用SQL语句对数据进行分组进行排序要比使用Java语句的方式精简的多代码也更具有易读性，他不会再像以前一样显示的进行遍历，对于数据的遍历在流的内部执行

P43 所以对流的操作又分为了中间操作和终端操作。

流水线：对流可以进行筛选，排序，阶段等操作，二这些操作返回的结果仍是流，所以就可以把这些操作连接起来。6不需要我们在进行遍历代码的编写

P44 假设我们有Dish类型的列表，Dish包含名称，是否时素菜，热量，菜系，

在本例中，我们先是对menu调用stream方法，由菜单得到一个流。 数据源是菜肴列表（菜单），它给流提供一个元素序列。接下来，对流应用一系列数据处理操作： filter、 map、 limit  
和collect。除了collect之外，所有这些操作都会返回另一个流，这样它们就可以接成一条流水线，于是就可以看作对源的一个查询。最后， collect操作开始处理流水线，并返回结果（它和别的操作不一样，因为它返回的不是流，在这里是一个List）。在调用collect之前，没有任何结果产生，实际上根本就没有从menu里选择元素。

看看reduce

空的Optional如何使用,会抛出异常吗