此作业评估您实现类并在小程序中使用它们的能力。

考虑由诸如移动电话的设备生成的GPS数据。 您当前的位置表示为一个点，

包括时间戳，经度（以度为单位），纬度（以度为单位）和海拔高度（以米为单位）。 启用GPS时的移动会生成一个轨道：一系列点代表来自GPS传感器的连续样本。

您的任务是实现可用于表示点和轨道的类，以及一个小程序

演示了如何使用类。 为了帮助您入门，我们为您提供了部分实施

其中一个类名为Point，另外一个类表示可能出现在你的异常中

码。

请务必严格按照以下说明进行操作。我们强烈建议您在Linux环境中执行这些步骤。从Minerva下载cwkfiles1.zip并将其放入存储库的顶层 - 即包含cwk1，cwk2和exercise子目录的目录。在存储库的顶层打开一个终端窗口，然后使用命令unzip cwkfiles1.zip解压缩Zip存档。这应该在cwk1 nameddata，lib，src和test之下创建子目录。确保这正是你所看到的！例如，您不应该有一个cwk1的子目录，它本身名为cwk1.您的代码中的所有代码都应该写在src子目录中的文件中。其他子目录应保持不变。请注意，如果您更改数据，库或测试中的任何内容，或者如果您更改，则可能会丢失标记

你有错误的目录结构来开始。删除cwkfiles1.zip并使用Git添加并提交新文件，然后将你的提交推送到togitlab.com。以下命令在存储库顶层的终端中执行，可以完成以下所有操作：git add cwk1git commit -m“Coursework 1的初始文件”git push origin mast

这值得18分。

在开始工作之前，请阅读以下所有三个小节。我们也建议您获益

在开始之前通过练习集3实现课程的一些经验。

3.1点类

完成我们通过添加提供的Point类的实现

•表示ZonedDateTime类型的时间戳的字段（见下文）

•表示经度，纬度和海拔的字段，全部为double类型

•使用作为参数提供的值初始化所有字段的构造函数（参见下文）

1

•字段的getter方法，名为getTime，getLongitude，getLatitude，getElevation（you

必须使用这些确切的名称）

•toString方法，返回Point的字符串表示形式，如下所示：

（-1.54853,53.80462），72.5米

（这里的值是经度，然后是纬度，然后是高程。）

ZonedDateTime是来自Java标准日期/时间API的类，在java.time包中 - 请参阅API

文档了解更多细节。

确保无法创建具有无效纬度或经度的Point对象。使用

类中提供的常量可以帮助您完成此操作，并抛出提供的异常类的实例，

GPSException，如果提供了不合适的坐标（请参阅讲座中讨论的时间类

这个例子）。

3.2赛道课

在名为Track.java的新文件中，编写一个名为Track的类，它可以存储一系列Point对象。

给这堂课

•适合存储Point对象序列的字段

•默认构造函数，用于创建不包含点的轨道

•一个readFile方法，它从文件中的数据创建一系列Point对象，文件的名称

作为String参数提供（见下文）

•名为add的方法，它将新点添加到轨道的末尾

•名为size的方法，返回当前存储在轨道中的点数

•名为get的方法，返回存储在指定序列中给定位置的Point对象

作为int参数（见下文）

readFile方法需要读取CSV文件，其示例可以在数据目录中找到。它

应该使用扫描仪来执行此操作。这里一个好方法是逐行读取文件，拆分行

在逗号上，然后分别解析每个项目。您可以使用ZonedDateTime的静态方法解析

用于解析时间戳的类。

readFile不应该捕获在读取文件期间可能发生的任何异常。它需要一个

异常规范，如果指定的文件不能，则声明它可能抛出FileNotFoundException

访问。如果文件中的任何记录不包含，您的实现应抛出GPSException

创建Point对象所需的确切值。

get方法应该使用传递给它的int值作为Point对象序列的索引，但是

在此之前，该方法应该检查此int值并抛出一个GPSException实例（如果它不在其中）

允许的范围。

3.3测试您的解决方案

为了帮助您检查是否已根据需要实现了这些类，我们提供了一些代码

对类进行一系列测试。要从终端窗口运行测试，请进入测试目录并执行

输入以下内容：

./basictests

这适用于Linux和macOS终端。它应该在Windows子系统Linux和Linux中工作

可能在Git Bash shell中工作，尽管尚未经过测试。从IntelliJ中运行测试应该

也是可能的;有关如何执行此操作的建议将单独提供。

我们强烈建议您在实施其余解决方案之前和之前运行这些测试

提交任何东西。您的分数部分取决于通过的这些测试的数量。

另请注意，在上面列出的所有方法都存在于各自的类中之前，这些测试不会编译。

因此，如果您想成功运行测试，则需要提供存根 - 虚拟实现

你尚未实现的任何方法。 void方法的存根可以有一个空方法

2

身体。返回值的方法的存根只能返回适当类型的常量（例如，0 in

数字返回类型的情况）

4完整解决方案

这值得进一步增加12分。

4.1赛道课

将以下内容添加到Track类：

•方法LowestPoint和highestPoint返回具有最低和最高的Point对象

海拔分别

•方法totalDistance，返回从点移动时以米为单位的总行进距离

指向轨道的整个长度

•方法averageSpeed，返回沿轨道的平均速度，以米/秒为单位

如果没有足够的点来进行必要的计算，所有这些方法都应抛出GPSException。

要实现totalDistance，您需要计算相邻点之间的“大圆距离”

轨道。 Point类中已经存在执行此操作的方法。给定两个Point对象，p和q，

它们之间以米为单位的大圆距离（忽略高程）将由。给出

double distance = Point.greatCircleDistance（p，q）;

要实现averageSpeed，您需要计算两次之间经过的时间

测量轨道上的第一个和最后一个点。你可以使用ChronoUnit类型 -

具体来说，可以在ChronoUnit.SECONDS上调用between方法来测量时间间隔

以秒为单位。

4.2测试

这里适用的注意事项与基本解决方案相同。你可以运行测试

./fulltests

4.3 TrackInfo计划

在名为TrackInfo.java的文件中，创建一个小程序，该程序根据文件中的数据创建Track对象

其名称作为命令行参数提供。你的程序应该显示：点数

在赛道上;它的最低点和最高点;行走的总距离;和平均速度。

当您的程序在数据目录中提供的文件walk.csv上运行时，它应该生成输出

非常相似：

跑道194分

最低点是（-1.53​​637,53.79680），35.0米

最高点是（-1.54835,53.80438），73.6米

总距离= 1.904公里

平均速度= 1.441 m / s

如果命令行中未提供文件名，则程序应打印有用的错误消息然后

使用System.exit终止，退出状态的值为非零值。

程序应该拦截从文件读取或执行计算时发生的任何异常。程序应该打印与异常关联的错误消息，然后使用System.exit

终止，退出状态的值为非零值。

3

5个高级任务

若要额外标记，请执行以下建议的两个选项之一。

这些任务更具挑战性，需要额外的阅读/研究;只有在尝试时才会尝试

您设法快速，轻松地完成以前的工作。

5.1选项1：KML文件

这值得额外2分。

1.向Track类添加一个名为writeKML的新方法。这应该有一个String参数，

表示文件名。它应该使用Google的Keyhole Markup将跟踪数据写入给定文件

语言格式。

2.修改TrackInfo，使其调用writeKML生成KML文件。

3.为walk.csv表示的轨道生成KML文件。通过将文件上传到可视化文件

谷歌地图（见图1）或将其导入谷歌地球。抓取结果的屏幕截图

将它放在cwk1目录中，以便它包含在您的提交中。

图1：Google Maps渲染的轨迹

5.2选项2：高程图

这值得额外4分。

1.通过阅读梁书第14章并尝试其中的一些例子来研究JavaFX，或者

通过研究文档

https://docs.oracle.com/javase/8/javase-clienttechnologies.htm

请特别注意绘图图表文档中的部分。

2.编写一个名为ElevationPlot的程序，该程序使用Java FX将高程绘制为距离的函数

沿着轨道。图2显示了这可能是什么样子的一个例子。

4

图2：使用Java FX生成的高程图

6提交

gitlab.com上的项目区域中有一个README文件，解释了提交过程。跟随

这些说明仔细。请记住通过Minerva提供的链接提交您的Zip档案。

提交截止日期为2月28日星期四上午10点。标准大学罚款5％

每天可用的商标将适用于迟到的工作，除非因正版而安排了延期

情有可原的。

请注意，所有提交都将受到自动抄袭检查。

7标记

此作业可以使用40个标记。

基本解决方案最多可以获得24分（占可用分数的60％）;完整的解决方案最多可以获得36分

（90％的可用）。

标记分配细分如下：

18测试基本解决方案

8测试完整解决方案

4 TrackInfo计划

4高级任务

4编码风格和评论

2使用版本控制

40