

محمد مهدی نظری

دانشجوی مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر

شماره دانشجویی - ۹۹۳۱۰۶۱

نمای کلی پروژه

هدف: هدف این پروژه تخمین کرایه خدمات تحویل بر اساس مختصات جغرافیایی و زمان‌های ثبت شده نقاط تحویل است. کرایه بر اساس قوانین قیمت‌گذاری مختلف، با در نظر گرفتن فاصله، سرعت و زمان روز برای هر تحویل محاسبه می‌شود.

فناوری‌ها: جاوا، پردازش فایل‌های CSV، چند نخ (Multi-threading)، کشینگ (Caching) و فرمول هاورساین (Haversine).

معماری سیستم

اجزا:

کلاس DeliveryPoint: نشان‌دهنده یک نقطه تحویل با فیلدهایی همچون شناسه تحویل، عرض جغرافیایی، طول جغرافیایی و زمان.

کلاس DistanceCalculator: مدیریت محاسبات فاصله با استفاده از فرمول هاورساین. این کلاس از کشینگ برای ذخیره فواصل محاسبه‌شده قبلی جهت دسترسی سریع استفاده می‌کند.

کلاس DeliveryFareEstimation: مدیریت خواندن داده‌های تحویل از فایل CSV، فیلتر کردن نقاط نامعتبر (جایی که سرعت از حد آستانه تجاوز می‌کند)، محاسبه کرایه‌ها و خروجی نتایج.

چند نخ (Multi-threading): برای بهینه‌سازی پردازش مجموعه داده‌های بزرگ، وظایف به دسته‌هایی تقسیم شده و به صورت موازی پردازش می‌شوند.

مکانیزم کشینگ (Caching): پیاده‌سازی شده برای جلوگیری از محاسبات فاصله تکراری، که زمان پردازش را به ویژه در هنگام پردازش مجموعه داده‌های بزرگ کاهش می‌دهد.

محاسبه فاصله

فرمول هاورساین: برای محاسبه فاصله بین دو نقطه بر روی سطح زمین بر اساس عرض و طول جغرافیایی آنها استفاده می‌شود.

استراتژی کشینگ: از یک HashMap برای ذخیره فواصل بین جفت‌های مختصاتی که قبلاً محاسبه شده‌اند، استفاده می‌شود. این کش در ابتدای برنامه بارگذاری و در پایان برنامه ذخیره می‌شود.

منطق محاسبه کرایه

اجزای کرایه:

کرایه اولیه: مبلغ اولیه که برابر با ۱.۳۰ واحد تعیین شده است.

کرایه بر اساس فاصله: بر اساس زمان روز (نرخ‌های روزانه یا شبانه) محاسبه می‌شود.

کرایه سرعت پایین: در صورتی که وسیله نقلیه تحویل با سرعت کمتر از ۱۰ کیلومتر در ساعت حرکت کند اعمال می‌شود.

حداقل کرایه: تضمین می‌شود که کرایه نهایی حداقل ۳.۴۷ واحد باشد.

فیلتر سرعت: نقاطی که سرعت آنها بیش از ۱۰۰ کیلومتر در ساعت باشد نامعتبر در نظر گرفته شده و قبل از محاسبه کرایه فیلتر می‌شوند.

ورودی/خروجی

ورودی: برنامه یک فایل CSV حاوی نقاط تحویل را می‌خواند. هر نقطه شامل شناسه تحویل، عرض جغرافیایی، طول جغرافیایی و زمان است.

خروجی: یک فایل CSV که حاوی تخمین کرایه برای هر تحویل است.

چند نخ‌ی برای بهبود عملکرد

ExecutorService: برنامه از چند نخ‌ی برای مدیریت مجموعه داده‌های بزرگ استفاده می‌کند. تحویل‌ها به دسته‌هایی تقسیم شده و هر دسته به صورت موازی پردازش می‌شود. دسته‌بندی: تحویل‌ها بر اساس تعداد کل تحویل‌ها و تعداد نخ‌های موجود به گروه‌هایی تقسیم می‌شوند.

مکانیزم کشینگ

فایل کش: محاسبات فاصله در یک فایل سریال شده (`distanceCache.ser``) ذخیره می‌شود تا از انجام محاسبات تکراری جلوگیری شود. کش در ابتدای اجرا بارگذاری شده و در طول اجرا به‌روز می‌شود.

توضیحات متدها

• **writeOutputToCSV:**

- **هدف:** این متد نتایج نهایی محاسبه کرایه را به یک فایل CSV خروجی می‌نویسد.
- **ورودی:** لیستی از نتایج تحویل، که شامل شناسه تحویل و کرایه محاسبه شده برای هر تحویل است.
- **عملکرد:** این متد یک فایل CSV ایجاد کرده و برای هر تحویل یک خط شامل شناسه و کرایه مربوطه می‌نویسد.

• **calculateFare:**

- **هدف:** این متد مسئول محاسبه کرایه برای هر تحویل است.
- **ورودی:** نقاط تحویل شامل اطلاعات مربوط به مختصات جغرافیایی و زمان ثبت شده.
- **عملکرد:** کرایه بر اساس فاصله بین نقاط و سرعت بین نقاط محاسبه می‌شود. این متد قوانین قیمت‌گذاری را اعمال کرده و فیلترهای مربوط به حداقل کرایه و کرایه بر اساس زمان روز را پیاده‌سازی می‌کند.

- خروجی: کرایه نهایی برای هر تحویل.

• filterInvalidPoints:

- هدف: این متد نقاط نامعتبر را بر اساس سرعت بالای ۱۰۰ کیلومتر در ساعت فیلتر می‌کند.
- ورودی: نقاط تحویل شامل اطلاعات مختصات و زمان.
- عملکرد: با استفاده از فرمول محاسبه سرعت، نقاطی که سرعت غیرمنطقی دارند فیلتر شده و از لیست نقاط حذف می‌شوند.
- خروجی: لیستی از نقاط تحویل معتبر.

• readData:

- هدف: این متد داده‌های نقاط تحویل را از یک فایل CSV ورودی می‌خواند.
- ورودی: فایل CSV حاوی مختصات و زمان تحویل‌ها.
- عملکرد: داده‌ها را خط به خط پردازش کرده و هر نقطه تحویل را به عنوان یک شیء از کلاس DeliveryPoint به لیست اضافه می‌کند.
- خروجی: لیستی از نقاط تحویل.

• haversine:

- هدف: این متد فاصله بین دو نقطه بر روی سطح زمین را بر اساس مختصات جغرافیایی (عرض و طول جغرافیایی) محاسبه می‌کند.
- ورودی: مختصات جغرافیایی دو نقطه (عرض و طول جغرافیایی).
- عملکرد: از فرمول هاورساین استفاده می‌کند تا فاصله کروی دقیق بین دو نقطه را محاسبه کند.
- خروجی: فاصله بین دو نقطه به کیلومتر.

• calculateSpeed:

- هدف: این متد سرعت بین دو نقطه تحویل را بر اساس فاصله و زمان محاسبه می‌کند.

- **ورودی:** مختصات و زمان دو نقطه تحویل.
- **عملکرد:** ابتدا فاصله بین نقاط با استفاده از متد **haversine** محاسبه می‌شود، سپس سرعت با تقسیم این فاصله بر اختلاف زمانی بین دو نقطه به دست می‌آید.
- **خروجی:** سرعت بین دو نقطه به کیلومتر در ساعت.

آزمون و اعتبارسنجی

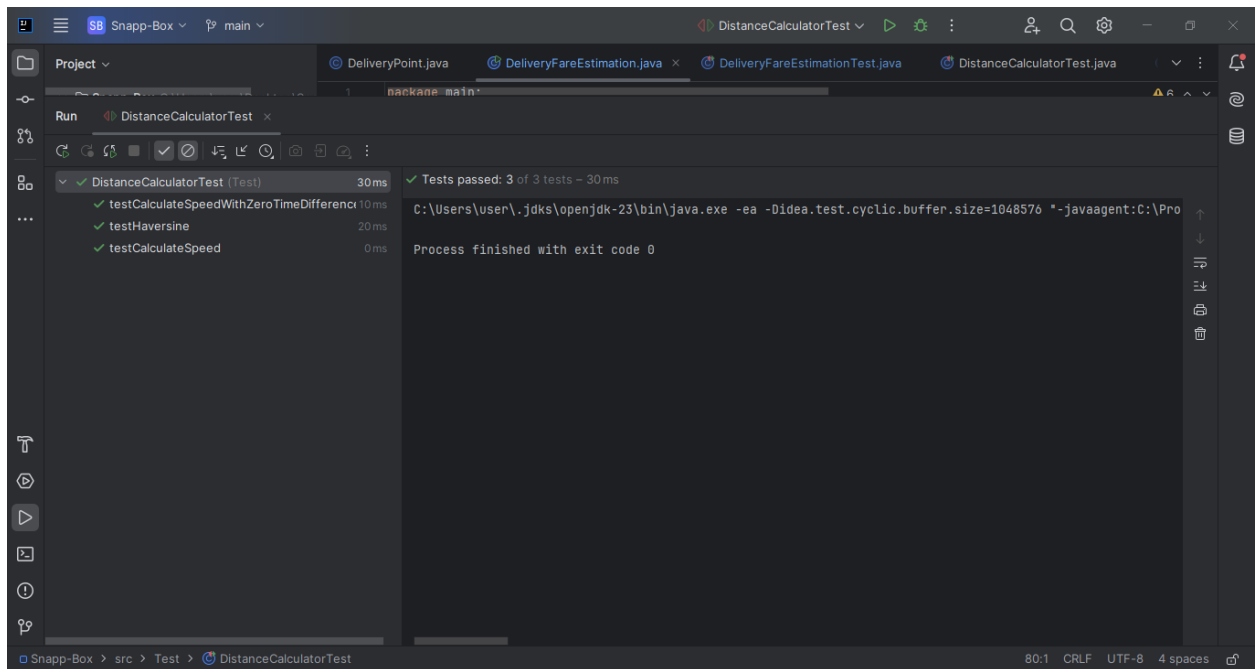
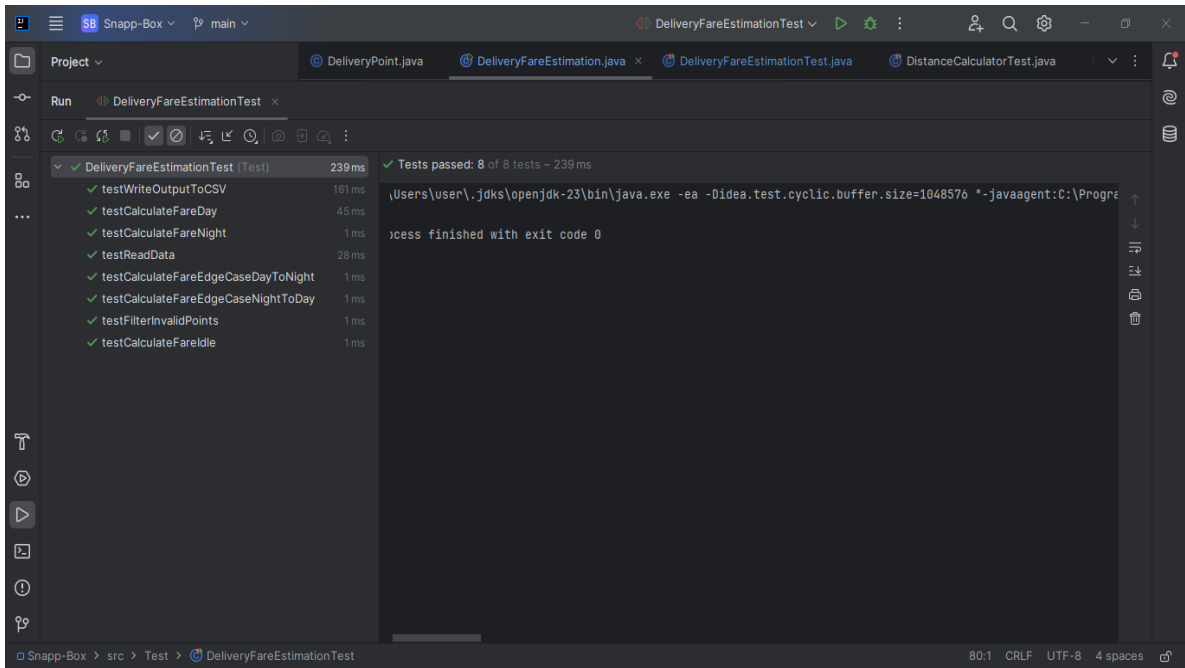
تست واحد

در این قسمت با استفاده از کتابخانه **Junit** برای متدهای گفته شده تست واحد مینویسیم:

تمامی متدها با دونقطه پیش فرض لندن و پاریس با اختلاف زمانی یکساعت در نظر گرفته شده که چون میدانیم فاصله لندن و پاریس ۳۴۳ کیلومتر است، پس سرعت حرکت ۳۴۳ کیلومتر بر ساعت است.

برای متد محاسبه کرایه ۵ حالت در نظر گرفته شده که شامل:

۱. حرکت در روز در بازه روز و ورود به شب در بازه ۲۳:۳۰ تا ۰۰:۳۰
۲. حرکت در روز در بازه ۱۳:۰۰ تا ۱۴:۰۰
۳. حرکت در بازه شب در ۲:۰۰ تا ۳:۰۰
۴. حرکت در شب و ورود به روز در بازه ۴:۳۰ تا ۵:۳۰
۵. بدون حرکت



تست انتها به انتها

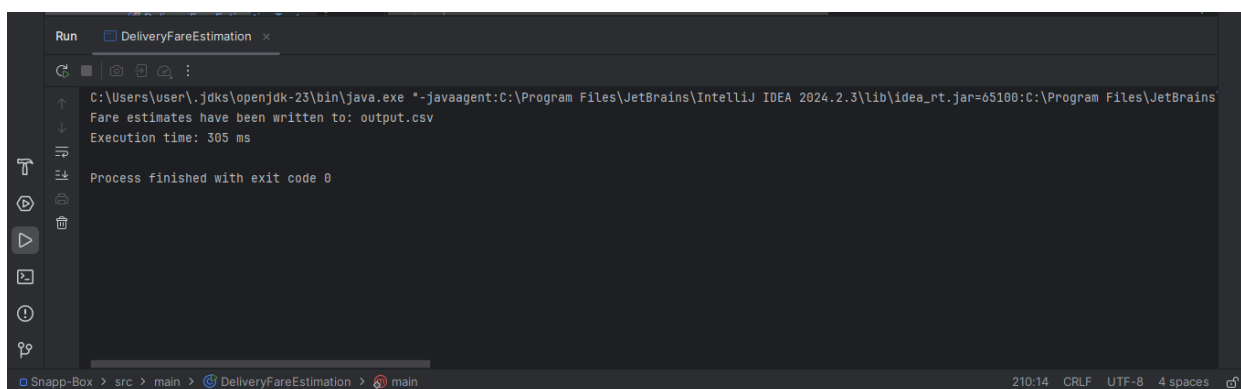
خروجی برای فایل تستی :

id_delivery	fare_estimate
۱	۳,۶۱
۲	۳,۴۷
۳	۳,۴۷

محاسبه زمان بندی:

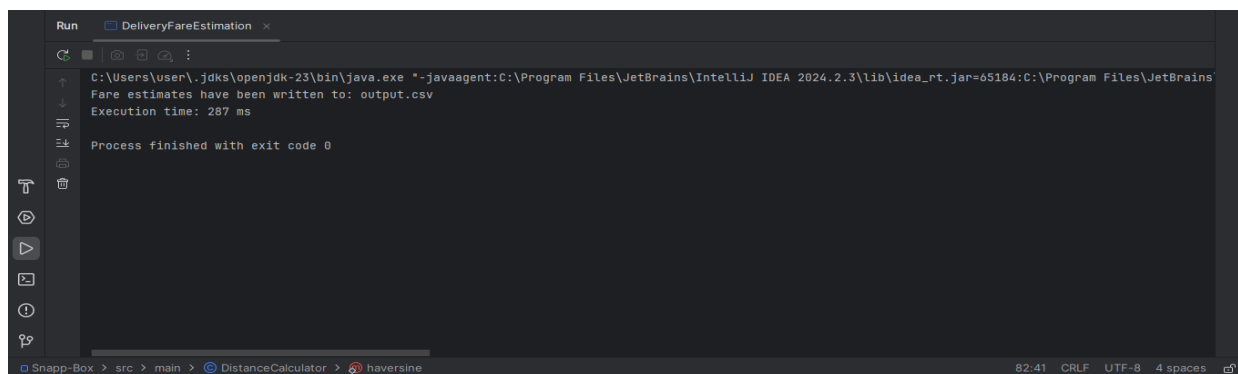
برای افزایش بهره وری و سرعت برنامه از چند نخ و کشینگ استفاده شده و زمان اجرای برنامه را در حالت های زیر بررسی میکنیم، برای اندازه گیری زمان از یک نمونه داده با اندازه بزرگتر استفاده کردیم تا اختلاف زمان محسوس تر باشد که شامل ۱۰۰ ارسال است. در استفاده از چندنخی کارها بین ۴ نخ تقسیم شده که با توجه سیستم موردنظر میتواند تغییر کند:

۱. بدون چندنخی و کشینگ



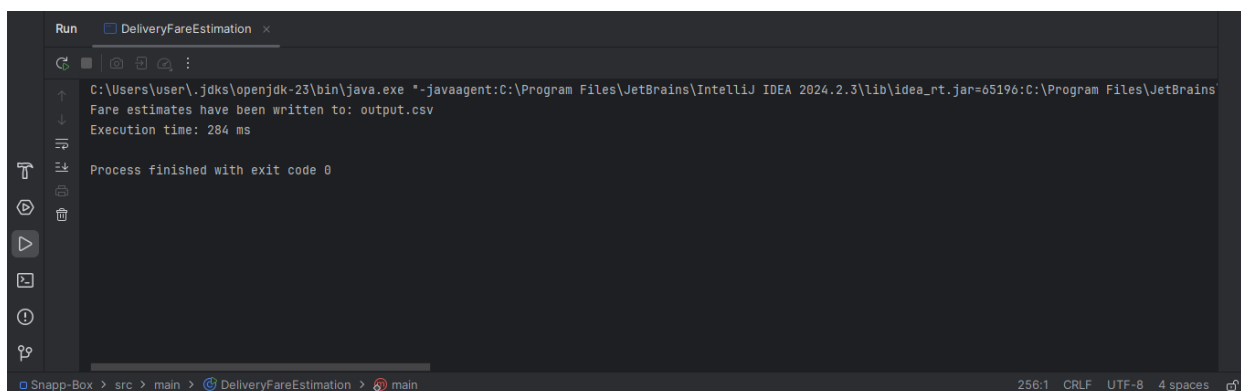
```
Run DeliveryFareEstimation x
C:\Users\User\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=65100:C:\Program Files\JetBrains
Fare estimates have been written to: output.csv
Execution time: 305 ms
Process finished with exit code 0
```

۲. تنها با کشینگ



```
Run DeliveryFareEstimation x
C:\Users\User\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=65184:C:\Program Files\JetBrains
Fare estimates have been written to: output.csv
Execution time: 287 ms
Process finished with exit code 0
```

۳. تنها با چند نخی

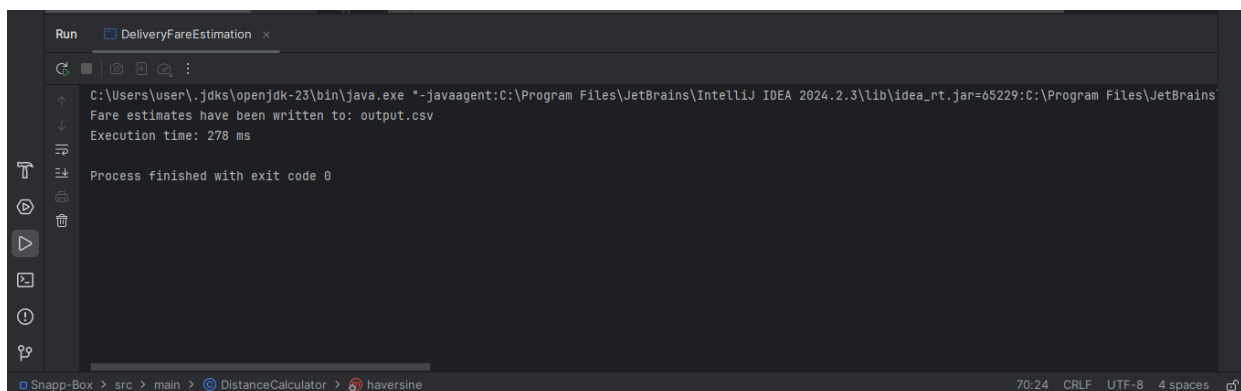


```
Run DeliveryFareEstimation x
C:\Users\user\jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=65196:C:\Program Files\JetBrains
Fare estimates have been written to: output.csv
Execution time: 284 ms

Process finished with exit code 0

Snapp-Box > src > main > DeliveryFareEstimation > main 256:1 CRLF UTF-8 4 spaces
```

۴. استفاده از چننخی و کشینگ



```
Run DeliveryFareEstimation x
C:\Users\user\jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=65229:C:\Program Files\JetBrains
Fare estimates have been written to: output.csv
Execution time: 278 ms

Process finished with exit code 0

Snapp-Box > src > main > DistanceCalculator > haversine 70:24 CRLF UTF-8 4 spaces
```