به نام خدا

پروژه پایان‌ترم درس مبانی کامپیوتر و برنامه‌سازی، نیمسال اول ۹۶-۹۵

دانشکده ریاضی، آمار و علوم کامپیوتر
پردیس علوم، دانشگاه تهران

شناسه پروژه: FINALPRJ

مهلت تحویل: ۱۵ بهمن ۱۳۹۵

ساعت ۵۹ : ۵۹ : ۲۳

توضیحات پروژه پایان‌ترم را کامل مطالعه کنید و موارد خواسته شده را به دقت پیاده‌سازی کنید. لازم به ذکر است در صورتی که کدهای تحویل داده شده کامپایل و یا اجرا نشود، ۷۵ درصد از نمره‌ی پروژه کسر خواهد شد. همچنین تحویل پروژه در دو مرحله‌ی غیر حضوری و حضوری صورت می‌گیرد. در مرحله غیر حضوری، تمامی دانشجویان باید کدهای خود را قبل از پایان مهلت تحویل معین شده (بالای صفحه) به ایمیل `milad.mozafari@ut.ac.ir` ارسال کنند. سپس در تاریخ دیگری که بعداً اعلام خواهد شد، هر دانشجو به صورت حضوری پروژه خود را تحویل می‌دهد. رعایت نکات زیر در انجام و ارسال پروژه ضروری است:

- تمامی فایل‌های مربوط به پروژه را در قالب یک فایل فشرده ارسال کنید.
- الگوریتم و توضیحات چگونگی پیاده‌سازی خود را به زبان فارسی و با استفاده از نرم‌افزارهای مناسب برای پردازش متن، به طور کامل بنویسید و در قالب یک فایل PDF همراه با دیگر فایل‌های پروژه ارسال کنید.
- برای متغیرها و توابع نام‌های مناسب با عملکرد آن‌ها انتخاب کنید.
- مفهوم encapsulation در طراحی کلاس‌ها را رعایت کنید.
- هر کلاس باید در فایلی جداگانه نوشته شود و نام کلاس‌ها همانند نام‌های نوشته شده در توضیحات باشد.
- پس از ارسال پروژه در حداکثر ۱۲ ساعت پیغام دریافت آن به شما ارسال خواهد شد. اگر بعد از ۱۲ ساعت پیغامی دریافت نکردید نسبت به ارسال دوباره پروژه اقدام نمایید. توصیه می‌شود ارسال فایل پروژه را به لحظات آخر موکول نکنید تا در صورت بروز مشکلات مختلف، نمره شما شامل دریافت جریمه تاخیر نشود.

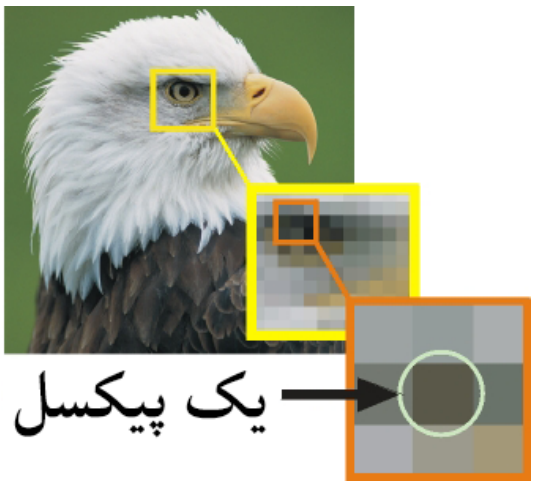
نهان‌سازی متن در تصویر

هدف این پروژه نوشتن برنامه‌ای برای نهان‌سازی اطلاعات متنی در اطلاعات تصویری و استخراج متن پنهان شده است. در ادامه یک الگوریتم برای رسیدن به این هدف شرح داده خواهد شد، اما قبل از آن لازم است با نحوه بازنمایی تصویر در سیستم‌های کامپیوتری و ابزارهای جاوا برای کار با تصاویر آشنا شویم.

۱ بازنمایی تصویر رنگی

در سیستم‌های کامپیوتری تصاویر دیجیتال به صورت ماتریسی از نقاط به نام پیکسل بازنمایی و ذخیره می‌شوند (شکل ۱). به عنوان مثال هنگامی که گفته می‌شود یک تصویر دارای ابعاد ۳۰۰ در ۲۰۰ پیکسل است، بدین معنا است که تصویر مذکور به صورت یک ماتریس با ۲۰۰ سطر و ۳۰۰ ستون از پیکسل‌ها نمایش داده می‌شود.

می‌دانیم گستره بسیاری از رنگ‌های مختلف، توسط ترکیب سه رنگ اصلی قرمز، سبز و آبی قابل تولید است. در همین جهت، ساده‌ترین نوع بازنمایی یک تصویر رنگی در سیستم‌های کامپیوتری، ذخیره میزان روشنایی هر یک از این سه رنگ (که به کانال رنگی نیز معروف هستند) برای هر پیکسل از تصویر است (شکل ۲). در قالب‌های متداول ذخیره‌سازی تصاویر رنگی، میزان روشنایی هر یک از این سه رنگ توسط یک عدد صحیح در بازه $[۰, ۲۵۵]$ مشخص می‌شود. از آنجایی که در این بازه تعداد ۲۵۶ عدد صحیح حضور دارد،



شکل ۱: یک پیکسل از یک تصویر.

تعداد ۸ بیت برای پوشش آن کافی است. در نتیجه، برای نمایش اطلاعات روشنایی یک پیکسل، که شامل اطلاعات روشنایی هر سه کانال رنگی آن می‌شود، تعداد ۲۴ (۳×۸) بیت نیاز است.

۲ ابزار کار با تصویر در جاوا

با توجه به توضیحات قسمت قبل، در جاوا ابزاری طراحی شده است که می‌تواند اطلاعات یک تصویر رنگی را بخواند و یا بر اساس اطلاعات محاسبه شده، یک تصویر رنگی را تولید کند. قالب‌های مختلفی برای تصاویر رنگی وجود دارند که در اینجا با توجه به هدف پروژه از قالب BMP استفاده خواهیم کرد.

در جاوا برای کار با تصاویر به کلاس‌های `ImageIO` و `BufferedImage` نیاز داریم. از کلاس `ImageIO` برای بارگذاری و ذخیره‌سازی تصاویر و از `BufferedImage` برای انجام محاسبات روی اطلاعات روشنایی هر پیکسل از تصویر استفاده می‌شود. همچنین می‌توان از کلاس `Color` برای آسان‌سازی کار با انواع مختلف رنگ‌ها بهره برد.

۱.۲ کتابخانه‌های مورد نیاز

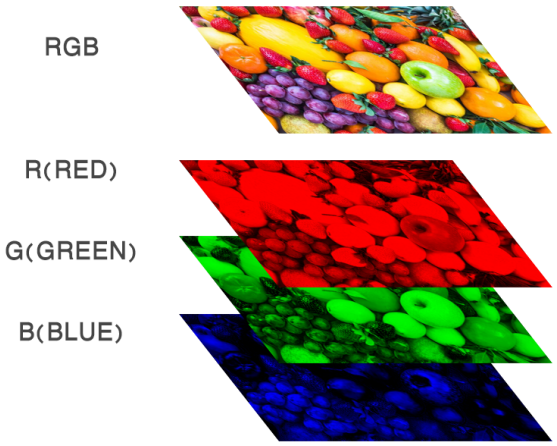
برای استفاده از ابزار بالا به import کردن کتابخانه‌های زیر نیاز است:

- `javax.imageio.ImageIO`
- `java.awt.image.BufferedImage`
- `java.awt.Color`

۲.۲ بارگذاری تصویر

برای بارگذاری اطلاعات یک تصویر رنگی از دستور زیر استفاده کنید. این دستور آدرس تصویر را از حافظه جانبی دریافت می‌کند و سپس در قالب یک نمونه شئ از کلاس `BufferedImage` در حافظه اصلی بارگذاری می‌کند.

```
BufferedImage sample1 = ImageIO.read(new File("image_address"));
```



شکل ۲: هر پیکسل از یک تصویر رنگی شامل اطلاعات میزان روشنایی هریک از سه رنگ اصلی است.

۳.۲ ذخیره‌سازی تصویر

برای ذخیره‌سازی اطلاعات یک تصویر رنگی روی حافظه جانبی از دستور زیر استفاده کنید. این دستور یک نمونه شئ از کلاس BufferedImage به همراه یک آدرس در حافظه جانبی را دریافت می‌کند و سپس تصویر را در آدرس مذکور ذخیره می‌کند

```
ImageIO.write(sample2, "BMP", new File("saving_address"));
```

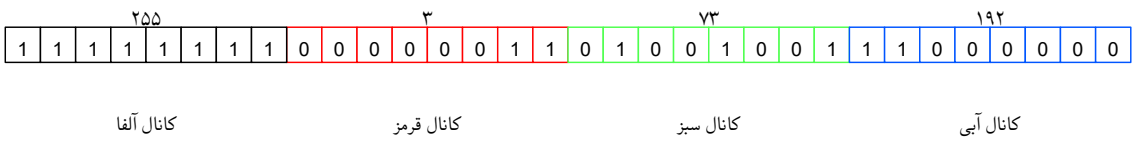
۴.۲ خواندن اطلاعات روشنایی در یک نقطه از تصویر

همانطور که پیش‌تر توضیح داده شد، یک تصویر از ماتریسی از پیکسل‌ها ساخته می‌شود که در هر پیکسل اطلاعات رنگ‌های (کانال‌های رنگی) قرمز (R)، سبز (G)، و آبی (B) ذخیره می‌شود. پیکسل‌های تصاویر رنگی از کانال دیگری به نام کانال آلفا نیز برخوردارند. این کانال میزان شفافیت یک پیکسل را در قالب‌هایی مانند PNG تعیین می‌کنند که مقادیر مجاز آن همچون دیگر کانال‌های رنگی می‌باشد و تعداد ۸ بیت برای آن کافیست. در نتیجه به طور کلی اطلاعات یک پیکسل در ۳۲ بیت قابل ذخیره‌سازی است.

فرض کنید یک تصویر رنگی با قالب BMP در یک نمونه شئ BufferedImage با نام sample ذخیره شده باشد. حال برای خواندن اطلاعات روشنایی یک پیکسل که در موقعیت عرضی x و ارتفاعی y قرار دارد از دستور زیر استفاده کنید:

```
sample.getRGB(x, y);
```

هنگامی که دستور بالا اجرا می‌شود، اطلاعات کانال‌های رنگی و کانال آلفا در قالب یک عدد صحیح ۳۲ بیتی بازگردانده می‌شود. به عبارت دیگر، عدد بازگردانده شده شامل ۴ قسمت ۸ بیتی است که هر کدام از این قسمت‌ها مقدار یکی از کانال‌ها را تعیین می‌کند. شکل ۳ یک مثال از عدد خروجی دستور بالا را نشان می‌دهد (توجه داشته باشید با توجه به نوع قالب عکس این پروژه، مقدار کانال آلفا همواره ۲۵۵ است).



شکل ۳: مثالی از خروجی تابع getRGB در مبنای دو که در آن میزان روشنایی رنگ قرمز، سبز و آبی به ترتیب برابرند با ۳، ۷۳ و ۱۹۲.

جدول ۱: معرفی عملگرهای بیتی. در ستون مثال‌ها مقدار A برابر با ۶۰ و مقدار B برابر با ۱۳ است. همچنین فرض کنید هر کدام از دو مقدار در ۸ بیت نمایش داده شوند.

Operator	Description	Example
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100.
| (bitwise or)	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101.
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001.
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000.
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111.
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>> 2 will give 15 which is 0000 1111.

۵.۲ نوشتن اطلاعات روشنایی در یک نقطه از تصویر

فرض کنید یک تصویر رنگی با قالب BMP در یک نمونه شئ `BufferedImage` با نام `sample` ذخیره شده باشد. حال برای نوشتن اطلاعات روشنایی یک پیکسل که در موقعیت عرضی `x` و ارتفاعی `y` قرار دارد از دستور زیر استفاده کنید:

```
sample.setRGB(x, y, rgbValue);
```

که در آن `rgbValue` یک عدد صحیح ۳۲ بیتی است که اطلاعات روشنایی را همانند قالب توضیح داده شده در بخش قبل ذخیره کرده است.

۶.۲ تولید تصویر خام

در صورتی که نیاز باشد یک تصویر خام (کامل سیاه) با عرض `width` و طول `height` برای ذخیره سازی مقادیر محاسبه شده آتی تولید کنید، دستور زیر را استفاده کنید:

```
BufferedImage raw = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
```

۷.۲ کلاس `Color`

از آنجایی که مقادیر روشنایی کانال‌های رنگی به صورت یکجا توسط یک عدد صحیح ۳۲ بیتی نمایش داده می‌شود، برای استخراج هر کدام از آن‌ها نیازمند استفاده از عملگرها و محاسبات بیتی هستیم. جدول ۱ عملگرهای بیتی جاوا را نشان می‌دهد. از دیگر راه‌کارهای کار با کانال‌های رنگی، استفاده از کلاس `Color` است. فرض کنید عدد صحیح `rgb` حاوی اطلاعات روشنایی یک پیکسل باشد، حال برای استخراج مقادیر هر یک از کانال‌های رنگی به صورت زیر عمل کنید:

```
Color c = new Color(rgb);
int r = c.getRed();
int g = c.getGreen();
int b = c.getBlue();
```

همچنین برای تبدیل سه مقدار رنگی red، green و blue به یک مقدار یکپارچه اطلاعات روشنایی rgb به صورت زیر عمل کنید:

```
Color c = new Color(red, green, blue);
int rgb = c.getRGB();
```

فایل ضمیمه با نام Gray.java یک نمونه کد برای تبدیل تصویر رنگی به تصویر سیاه و سفید است. مطالعه این قطعه کد برای آشنایی بیشتر با ابزار کار با تصویر جاوا اکیداً توصیه می‌شود.

۳ الگوریتم نهان‌سازی

در الگوریتم‌های مختلف نهان‌سازی، سعی بر این است که تا حد امکان اثرات نهان کردن اطلاعات در تصویر از دید افراد و سیستم‌های مکاشفه پنهان باشد. در این پروژه ایده کلی نهان‌سازی به این صورت است که ابتدا تمامی کاراکترهای متن مورد نظر به قالب دودویی تبدیل می‌شوند (برای هر کاراکتر ۸ بیت در نظر بگیرید)، سپس هر بیت از آن با کم‌ارزش‌ترین بیت مربوط به اطلاعات روشنایی تنها یکی از کانال‌های رنگی یک پیکسل که به صورت تصادفی انتخاب شده است، تعویض می‌شود. با این روش، در هر نقطه انتخاب شده از تصویر هدف، میزان روشنایی در حداکثر یک کانال رنگی و حداکثر یک واحد (به دلیل تغییر بیت انتهایی) تغییر می‌کند که شناسایی آن را دشوار می‌کند.

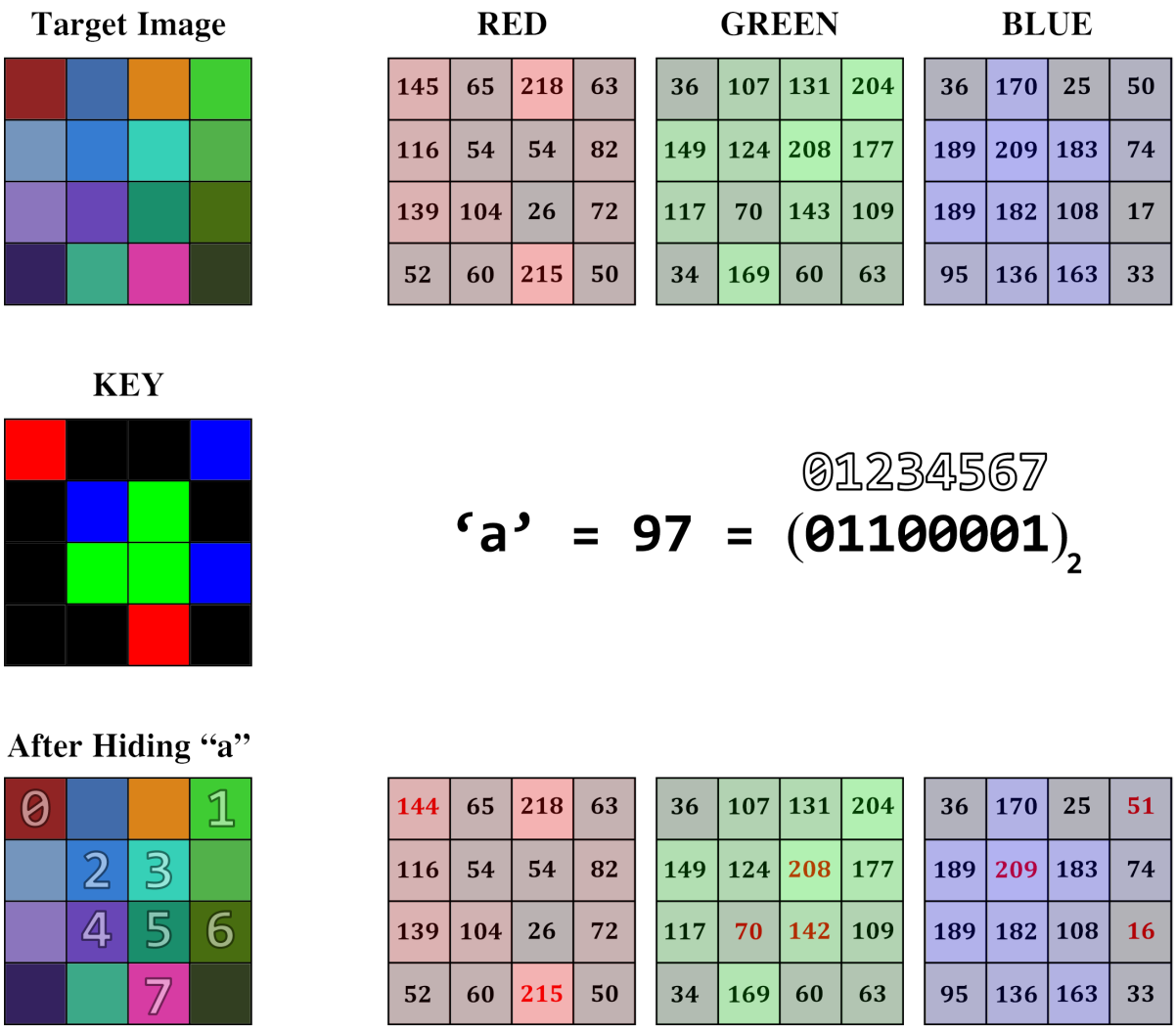
با توجه به اینکه پیکسل‌ها به صورت تصادفی انتخاب شده‌اند، نیازمند یک ترتیب و یک کلید برای بازگردانی اطلاعات پنهان شده در تصویر هستیم. در این الگوریتم بعد از اینکه به تعداد بیت‌های متن ورودی پیکسل تصادفی انتخاب می‌شود، ذخیره‌سازی بیت‌ها در این پیکسل‌ها به ترتیب اولویت شماره سطر (ارتفاع) و سپس شماره ستون (عرض) پیکسل انجام می‌شود. حال با مشخص بودن ترتیب، تنها کافیست اطلاعات مکانی پیکسل‌هایی که برای نهان‌سازی انتخاب شده‌اند را در یک کلید ذخیره کنیم. در این پروژه کلید نیز یک فایل تصویری است که به صورت زیر تولید می‌شود:

- ابتدا یک تصویر خام (کامل سیاه) به ابعاد تصویر هدف تولید می‌شود.
- به ازای هر پیکسل انتخاب شده برای نهان‌سازی که در موقعیت عرضی x و ارتفاعی y قرار دارد و کانال رنگی c آن انتخاب شده است (که c برابر است با یکی از رنگ‌های R، G و یا B)، مقدار روشنایی کانال رنگی c مربوط به پیکسل قرارگرفته در موقعیت عرضی x و ارتفاعی y در تصویر کلید برابر با ۲۵۵ قرار داده می‌شود (شکل ۴ یک مثال از الگوریتم را نمایش می‌دهد).

به این ترتیب با پیمایش تصویر کلید، می‌توان محل بیت‌های مربوط به کاراکترهای متن ورودی را پیدا کرد. حال برای بازگردانی متن پنهان شده، کافیست روی تصویر کلید به ترتیب اول سطر و سپس ستون حرکت کرد، پیکسل‌های حاوی اطلاعات را شناسایی کرد، آخرین بیت را از مقدار روشنایی کانال رنگی انتخاب شده استخراج کرد و در نهایت با به هم چسباندن هر ۸ بیت از بیت‌های استخراج شده، هر یک از کاراکترها را آشکار کرد.

۴ جزئیات پیاده‌سازی

برای پیاده‌سازی این پروژه باید دو کلاس با نام‌های TextHider و TextExtractor طراحی کنید. هدف TextHider انجام عمل نهان‌سازی متن در تصویر و هدف TextExtractor انجام عمل استخراج متن نهان‌سازی شده در تصویر است. رابط عمومی (public interface) این دو کلاس در ادامه متن آورده شده است.



شکل ۴: یک مثال از الگوریتم نهان‌سازی. در این مثال یک رشته حاوی یک کاراکتر a در یک تصویر ۴ × ۴ پیکسل پنهان شده است. در این شکل، اعدادی که روی تصویر نهایی قرار داده شده‌اند، ترتیب دخیره‌سازی بیت‌های مربوط به کاراکتر a را مشخص می‌کنند.

۱.۴ `TextHider`

- **Constructors**

این کلاس تنها یک سازنده دارد. این سازنده باید آدرس فایل متنی برای نهان‌سازی را دریافت کند.

- **Methods**

این کلاس تنها یک متد با نام `hide` دارد. این متد خروجی ندارد و به عنوان ورودی، آدرس تصویر هدف برای نهان‌سازی و آدرس فولدر مقصد برای ذخیره‌سازی دو فایل کلید و تصویری که متن در آن پنهان شده است را دریافت می‌کند. هدف این تابع اجرای عملیات نهان‌سازی متن دریافتی در هنگام ساخته شدن نمونه شئ در تصویر ورودی خود است. همچین در صورتی که تعداد پیکسل‌های مورد نیاز برای نهان‌سازی از $\frac{3}{4}$ام تعداد کل پیکسل‌های تصویر بیشتر باشد، باید یک exception با پیام مناسب تولید شود و عملیات نهان‌سازی انجام نشود.

۲.۴ TextExtractor

- **Constructors**

این کلاس تنها یک سازنده دارد. این سازنده باید آدرس فایل تصویری که در آن متن نهان شده است را دریافت کند.

- **Methods**

این کلاس تنها یک متد با نام extract دارد. این متد خروجی ندارد و به عنوان ورودی، آدرس فایل کلید و آدرس فولدر مقصد برای ذخیره‌سازی فایل متنی استخراج شده را دریافت می‌کند. هدف این تابع اجرای عملیات آشکارسازی متن پنهان شده به وسیله کلید ورودی در تصویری است که در هنگام ساخته شدن نمونه شئ دریافت شده است.

اکنون متد main باید به گونه‌ای طراحی و پیاده‌سازی شود که کاربر برنامه بتواند با استفاده از آرگومان‌های خط فرمان (command-line arguments) از امکانات برنامه استفاده کند. قالب آرگومان‌ها به شرح زیر است:

- **انجام عملیات نهان‌سازی:**

```
java Main -h TEXT_FILE_ADDRESS IMAGE_FILE_ADDRESS SAVING_ADDRESS
```

که در آن TEXT_FILE_ADDRESS آدرس فایل متنی و IMAGE_FILE_ADDRESS آدرس فایل تصویر هدف برای نهان‌سازی است. همچنین آرگومان SAVING_ADDRESS که فولدر مقصد برای ذخیره‌سازی را مشخص می‌کند اختیاری بوده و در صورتی که کاربر آن را وارد نکند، فایل‌های خروجی باید در محل فایل اجرایی ذخیره شوند.

- **انجام عملیات آشکارسازی:**

```
java Main -u IMAGE_FILE_ADDRESS KEY_FILE_ADDRESS SAVING_ADDRESS
```

که در آن IMAGE_FILE_ADDRESS آدرس فایل تصویر دارای متن نهان‌شده و KEY_FILE_ADDRESS آدرس فایل تصویر کلید برای استخراج متن است. همچنین آرگومان SAVING_ADDRESS که فولدر مقصد برای ذخیره‌سازی را مشخص می‌کند اختیاری بوده و در صورتی که کاربر آن را وارد نکند، فایل‌های خروجی باید در محل فایل اجرایی ذخیره شوند.

توجه کنید که تمامی exceptionها باید دریافت شوند و در خروجی پیغامی متناسب با نوع خطایی که رخ داده است نمایش داده شود.