Course: EECS4313
Deadline: Apr 7th, 11:59pm
Submission: via eClass
This is a group (of 3) project.

This project is designed to assess your ability in testing a program using specification and structured-based testing.

Before you begin, I would like to remind you that this assignment must be treated as a take-home exam. This implies that you cannot discuss your solution with peers, except your teammate, or seek help from other resources, including but not limited to peers, others, or paid tutors. Submitting the solution for this project implies your agreement that the work is 100% yours. Therefore, we reserve the right to check for any forms of academic dishonesty, including plagiarism and cheating.

Refusing to adhere to the policy stated above is a violation of academic integrity that will have serious consequences. Please read about the definition of academic integrity and the penalties one may face in case of a violation at https://lassonde.yorku.ca/student-life/academic-support/academic-honesty-integrity.

Problem Description

You will work in a group of three to test Thumbnailator, a thumbnail generation library for Java. This library simplifies thumbnail generation in Java without any reliance on external dependencies. As well it reduces the hurdle of learning how to use the Image I/O API, the Java 2D API, image processing techniques, and image scaling techniques to generate thumbnails.

The project can be found here: https://github.com/Hashir-Jamil/eecs-4313-w25-project. Follow the instructions in Appendix A of this text to download and set up the project. The descriptions for each method can be found in each file as a multiline java comment above the method. If a description is missing or not clear, please reach out to the TA: Hashir Jamil: hashirj1@yorku.ca.

Depending on the group you register for, you will be responsible for testing a segment of the code. To learn which part you should test, please refer to Appendix B of this text. **All tests should be created in the src/test/java directory.**

Your task is to:

- Determine the most suitable testing technique for your code. Write a report justifying your selection. Note that you may select different techniques for different functions. If any faults are discovered, document them using the bug analysis and reporting template provided during the lecture.
- Use Miller's metric (JaCoCo) to measure the coverage percentage of the code you are responsible for. If the coverage is not 100% (or close to it), develop and add special test cases, and explain these additions in your report. Discuss the achieved coverage percentage based on the method selected in part A, and describe the additional test cases designed to maximize coverage.
- Incorporate generative AI tools as part of your testing process. Document this aspect in your report and be prepared to present it if requested.

Submission:

You are required to submit two files:

- A **PDF file** containing:
  - Your report on the selected testing approach, with a justification for your choice.
  - Completed bug report templates. If no bugs were discovered, include a refactoring suggestion with a justification explaining how it would impact testing.
- A **code file** containing your tester, implemented using either **JUnit 5** or the **Reflect** framework.

Grading Scheme:

[15 points]: are assigned to the report that is clear, detailed, and effectively justifies the testing method used.

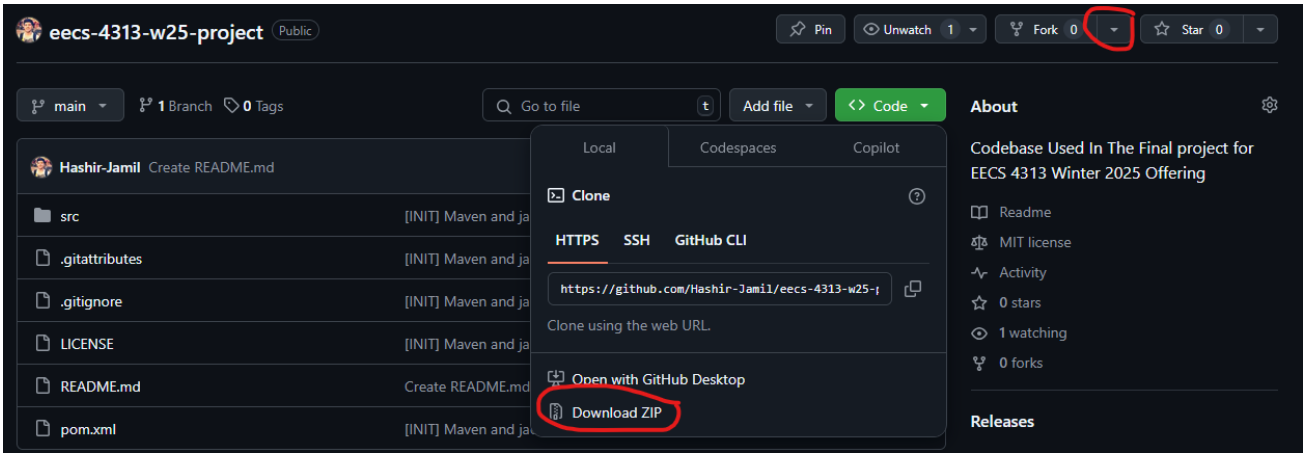
	5	3.5	2	1
Organization of the report	Well-organized with a clear structure.	Mostly organized, with some areas for improvement.	Organized but lacks clarity in structure.	Poorly organized report, hindering understanding
Identification of appropriate technique	Provides good justification for used techniques for all the methods by addressing the strength and weakness of it.	Provides good justification for used techniques for some of the methods by addressing the strength and weakness of it.	Provides poor justification for used techniques for all the methods by addressing the strength and weakness of it.	Provides poor justification for used techniques for some of the methods by addressing the strength and weakness of it..
Identification of fault and/or refactoring suggestions as well as providing test cases.	Provides detailed explanation of the found fault and/or provide convincing reason for refactoring. Coverage is close enough to 100%	Provides good explanation of the found fault and/or provide good reason for refactoring. The coverage is not complete.	Provides poor explanation of the found fault and/or provide poor reason for refactoring. The coverage is not complete.	Provides no explanation of the found fault and/or provide no reason for refactoring. No test cases were provided.

[5 points]: These points are awarded based on how effectively you have utilized generative AI to test this project. You will present this component on your report. The assistance you can receive from AI includes, but is not limited to, the following tasks. You must demonstrate the use of generative AI for at least **two** of these:

- Automated test case generation
- Applying more comprehensive coverage testing methods
- Predicting faults by analyzing the source code
- Categorizing bugs based on their severity (if any bugs are found)
- Providing refactoring suggestions
- Predicting performance under varying loads and identifying bottlenecks
- Improving the effectiveness and efficiency of the test suites
- Identifying security vulnerabilities and threats

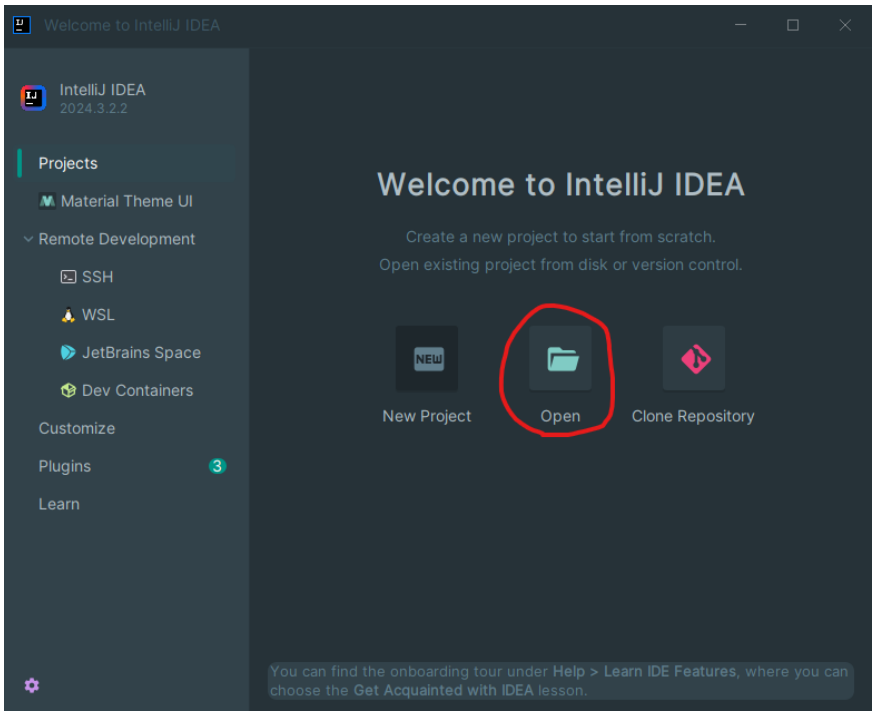
Appendix A: Project Setup

1. Go to https://github.com/Hashir-Jamil/eecs-4313-w25-project and either fork the repository to your own GitHub account or download a zip of the codebase and extract it to your preferred directory (recommended).

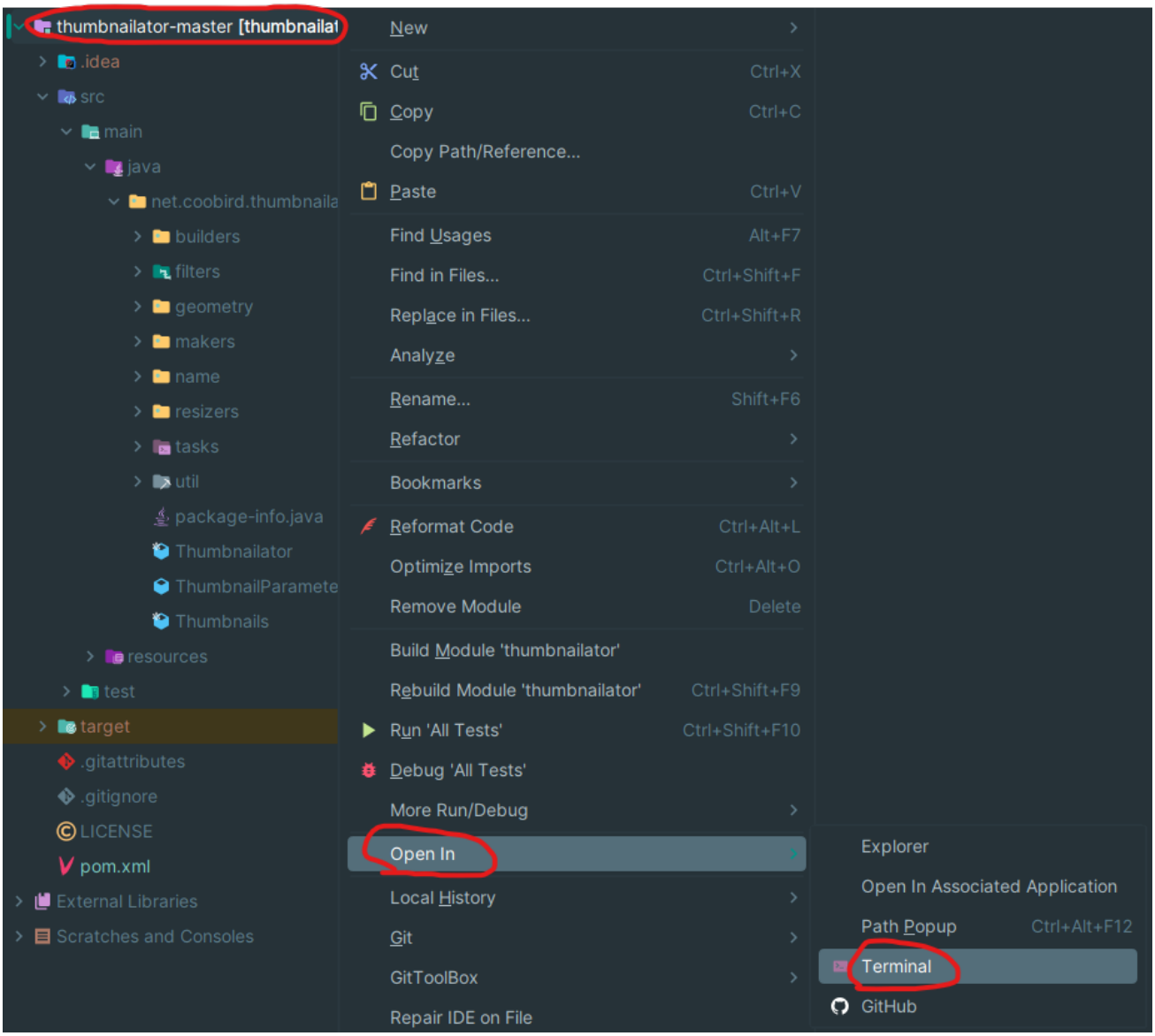


2. Import the project to your IDE of choice. This guide will show instructions for IntelliJ IDEA.

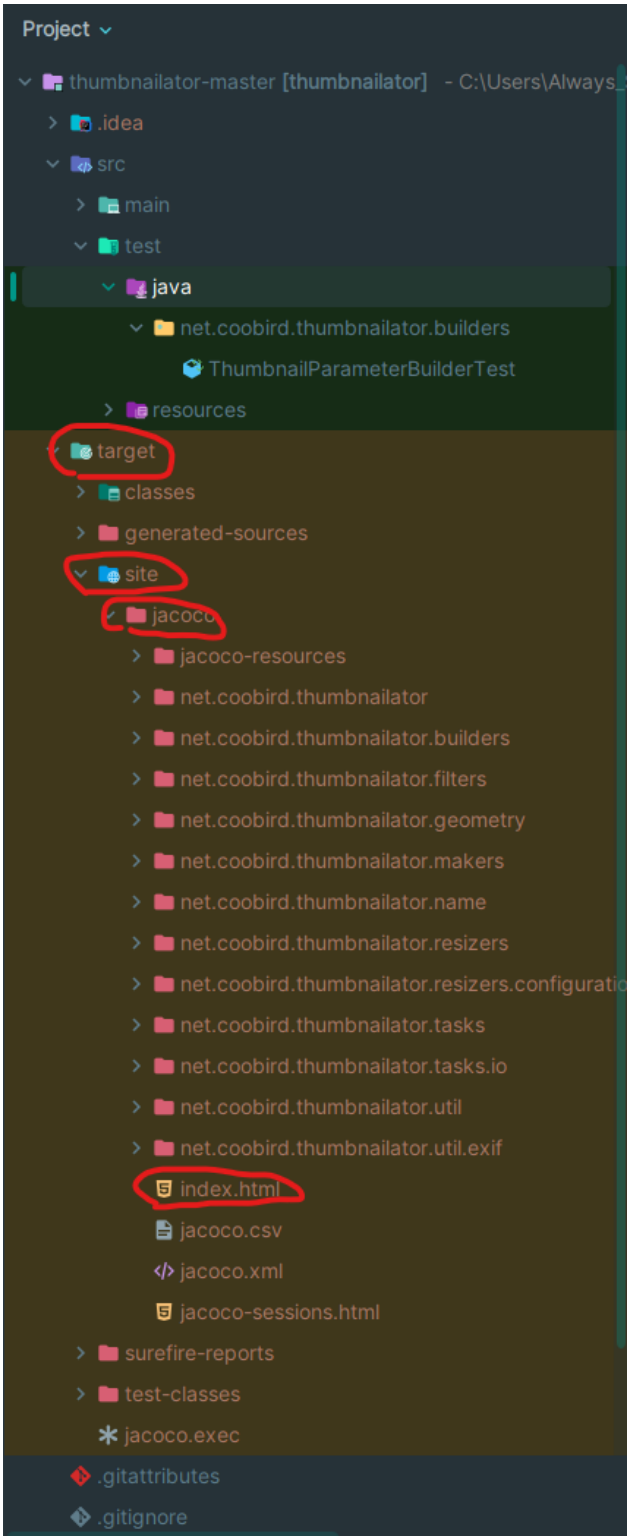
3. IntelliJ IDEA: File → Open →Archive to import the project.



4. Right click parent directory of project → click Open In → Select Terminal. Next, execute the command: **mvn clean test jacoco:report** to download all dependencies and run the existing default tests for the first time to confirm your project builds correctly. As well, this will ensure your JaCoCo code coverage report is generated in the right folder



5. Next, navigate to Target → site → jacoco → index.html and open this html file in your internet browser of choice to view the JaCoCo testing report for coverage analysis by package, class, and method.



6. You will use this jacoco dashboard to measure and improve your coverage as you work on your project. The top-level view is by package. You can click on any package to get to sub packages/classes and click on any classes to get to methods. If you click on a method, you will see a line-by-line breakdown of coverage for the test code. You only need to review the coverage for the methods your group is responsible for implementing tests for.

thumbnailator

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
net.coobird.thumbnailator		5%		5%	264	281	674	719	135	151	17	18
net.coobird.thumbnailator.tasks.io		0%		0%	193	193	498	498	66	66	12	12
net.coobird.thumbnailator.filters		0%		0%	73	73	244	244	41	41	12	12
net.coobird.thumbnailator.util.exif		0%		0%	62	62	151	151	24	24	5	5
net.coobird.thumbnailator.geometry		0%		0%	47	47	94	94	33	33	14	14
net.coobird.thumbnailator.makers		0%		0%	49	49	132	132	27	27	4	4
net.coobird.thumbnailator.resizers		30%		2%	31	48	77	118	11	28	0	8
net.coobird.thumbnailator.name		0%		0%	30	30	64	64	28	28	8	8
net.coobird.thumbnailator.util		0%		0%	28	28	72	72	11	11	3	3
net.coobird.thumbnailator.tasks		0%		0%	27	27	56	56	22	22	5	5
net.coobird.thumbnailator.resizers.configurations		0%		n/a	17	17	40	40	17	17	5	5
net.coobird.thumbnailator.builders		56%		61%	27	43	50	93	18	26	1	2
Total	9,375 of 9,889	5%	811 of 848	4%	848	898	2,152	2,281	433	474	86	96

7. All tests should be implemented in the directory **src/test/java** organized by package and by class. There is an example test class in this directory for your reference. Use this as a basis for how to write your tests.

Appendix B: Group Assignment

Group A

Package: net.coobird.thumbnailator.filters

Canvas.java
```
public BufferedImage apply(BufferedImage img)
```

Caption.java
```
public Caption(String caption, Font font, Color c, float alpha, Position position, int insets)
public BufferedImage apply(BufferedImage img)
```

Colorize.java
```
public Colorize(Color c, int alpha)
public BufferedImage apply(BufferedImage img)
```

Pipeline.java
```
public BufferedImage apply(BufferedImage img)
```

Package: net.coobird.thumbnailator.geometry

AbsoluteSize.java
```
public AbsoluteSize(Dimension size)
public AbsoluteSize(int width, int height)
public Dimension calculate(int width, int height)
```

Group B

Package: net.coobird.thumbnailator.name

ConsecutivelyNumberedFilenames.java
```
public ConsecutivelyNumberedFilenames(File dir, String format, int start) throws IOException
```

Package: net.coobird.thumbnailator.resizers

BilinearResizer.java
```
public void resize(BufferedImage srcImage, BufferedImage destImage)throws NullPointerException
public BilinearResizer(Map<RenderingHints.Key, Object> hints)
public void resize(BufferedImage srcImage, BufferedImage destImage)throws NullPointerException
```

DefaultResizerFactory.java
```
public Resizer getResizer(Dimension originalSize, Dimension thumbnailSize)
```

ProgressiveBilinearResizer.java
```
public void resize(BufferedImage srcImage, BufferedImage destImage)throws NullPointerException
```

Group C

Package: net.coobird.thumbnailator.tasks.io

BufferedImageSink.java
```
public BufferedImage getSink()
```

FileImageSink.java
```
public FileImageSink(File destinationFile, boolean allowOverwrite)
public FileImageSink(String destinationFilePath)
public FileImageSink(String destinationFilePath, boolean allowOverwrite)
private static boolean isMatchingFormat(String formatName, String fileExtension) throws
UnsupportedFormatException
private static String getExtension(File f)
public void write(BufferedImage img) throws IOException
```

Group D

Package: net.coobird.thumbnailator.util.exif

ExifFilterUtils.java
```
public static ImageFilter getFilterForOrientation(Orientation orientation)
```

Package: net.coobird.thumbnailator.geometry

Region.java
```
public Region(Position position, Size size)
public Rectangle calculate(int outerWidth,int outerHeight, boolean flipHorizontal, boolean flipVertical,
boolean swapDimensions)
```

RelativeSize.java
```
public RelativeSize(double scalingFactor)
public Dimension calculate(int width, int height)
```

Package: net.coobird.thumbnailator.makers

ScaledThumbnailMaker.java
```
public ScaledThumbnailMaker(double widthFactor, double heightFactor)
public ScaledThumbnailMaker scale(double widthFactor, double heightFactor)
```

FixedSizeThumbnailMaker.java
```
public FixedSizeThumbnailMaker(int width, int height, boolean aspectRatio, boolean fit)
public FixedSizeThumbnailMaker size(int width, int height)
```

Group E

Package: net.coobird.thumbnailator.tasks.io

FileImageSource.java
```
public FileImageSource(File sourceFile)
public FileImageSource(String sourceFilePath)
public BufferedImage read() throws IOException
```

Package: net.coobird.thumbnailator.tasks

FileThumbnailTask.java
```
public FileThumbnailTask(ThumbnailParameter param, File sourceFile, File destinationFile)
```

SourceSinkThumbnailTask.java
```
public SourceSinkThumbnailTask(ThumbnailParameter param, ImageSource<S> source, ImageSink<D> destination)
```

StreamThumbnailTask.java
```
public StreamThumbnailTask(ThumbnailParameter param, InputStream is, OutputStream os)
```

Package: net.coobird.thumbnailator.util.exif

ExifUtils.java
```
public static Orientation getExifOrientation(ImageReader reader, int imageIndex) throws IOException
```

Group F

Package: net.coobird.thumbnailator.util

ThumbnailatorUtils.java
```
public static List<String> getSupportedOutputFormats()
public static boolean isSupportedOutputFormat(String format)
public static List<String> getSupportedOutputFormatTypes(String format)
public static boolean isSupportedOutputFormatType(String format, String type)
```

Package: net.coobird.thumbnailator.builders

BufferedImageBuilder.java
```
public BufferedImageBuilder(Dimension size)
public BufferedImageBuilder(Dimension size, int imageType)
public BufferedImageBuilder(int width, int height)
public BufferedImageBuilder(int width, int height, int imageType)
public BufferedImage build()
```

Group G

Package: net.coobird.thumbnailator.util.exif

ExifUtils.java
```
public static Orientation getOrientationFromExif(byte[] exifData)
```

IfdStructure.java
```
public IfdStructure(int tag, int type, int count, int offsetValue)
public int hashCode()
public boolean equals(Object obj)
public String toString()
```

Package: net.coobird.thumbnailator.util.exif

BufferedImages.java
```
public static BufferedImage copy(BufferedImage img)
public static BufferedImage copy(BufferedImage img, int imageType)
```

Group H

Package: net.coobird.thumbnailator.builders

BufferedImageBuilder.java

```
public BufferedImageBuilder imageType(int imageType)
public BufferedImageBuilder size(int width, int height)
public BufferedImageBuilder width(int width)
public BufferedImageBuilder height(int height)
```

ThumbnailParameterBuilder.java

```
public ThumbnailParameterBuilder size(Dimension size)
public ThumbnailParameterBuilder size(int width, int height)
public ThumbnailParameterBuilder scale(double scalingFactor)
public ThumbnailParameterBuilder scale(double widthScalingFactor, double heightScalingFactor)
public ThumbnailParameterBuilder region(Region sourceRegion)
```

Group I

Package: net.coobird.thumbnailator.builders

ThumbnailParameterBuilder.java

```
public ThumbnailParameterBuilder keepAspectRatio(boolean keep)
public ThumbnailParameterBuilder quality(float quality)
public ThumbnailParameterBuilder format(String format)
public ThumbnailParameterBuilder formatType(String formatType)
public ThumbnailParameterBuilder filters(List<ImageFilter> filters)
public ThumbnailParameterBuilder resizer(Resizer resizer)
public ThumbnailParameterBuilder resizerFactory(ResizerFactory resizerFactory)
public ThumbnailParameterBuilder fitWithinDimensions(boolean fit)
public ThumbnailParameterBuilder useExifOrientation(boolean use)
```

Group J

Package: net.coobird.thumbnailator.builders

ThumbnailParameterBuilder.java

```
public ThumbnailParameter build()
```

Package: net.coobird.thumbnailator

Thumbnails.java

```
private static void validateDimensions(int width, int height)
private static void checkForNull(Object o, String message)
private static void checkForEmpty(Object[] o, String message)
private static void checkForEmpty(Iterable<?> o, String message)
public static Builder<File> of(String... files)
public static Builder<File> of(File... files)
public static Builder<URL> of(URL... urls)
public static Builder<? extends InputStream> of(InputStream... inputStreams)
```

Group K

Package: net.coobird.thumbnailator

Thumbnails.java

```
public static Builder<BufferedImage> of(BufferedImage... images)
public static Builder<File> fromFilenames(Iterable<String> files)
public static Builder<File> fromFiles(Iterable<File> files)
public static Builder<URL> fromURLs(Iterable<URL> urls)
public static Builder<InputStream> fromInputStreams(Iterable<? extends InputStream> inputStreams)
public static Builder<BufferedImage> fromImages(Iterable<BufferedImage> images)
```

Group L

Package: net.coobird.thumbnailator.tasks.io

InputStreamImageSource.java

```
public InputStreamImageSource(InputStream is)
public void setThumbnailParameter(ThumbnailParameter param)
public BufferedImage read() throws IOException
private BufferedImage readImage(ImageReader reader) throws IOException
private Rectangle calculateSourceRegion(int width, int height, Orientation orientation, Region region)
```

Group M

Package: net.coobird.thumbnailator.tasks.io

OutputStreamImageSink.java

```
public OutputStreamImageSink(OutputStream os)
public void write(BufferedImage img) throws IOException
private void setCompressionModeExplicit(ImageWriteParam writeParam)
private boolean isJpegOrBmp(String formatName)
private boolean isPng(String formatName)
private boolean isDefaultPngWriter(ImageWriter writer)
private boolean isJava9OrNewer()
public OutputStream getSink()
```

Group N

Package: net.coobird.thumbnailator

ThumbnailParameter.java

```
private ThumbnailParameter(Dimension thumbnailSize, double widthScalingFactor, double heightScalingFactor,
Region sourceRegion, boolean keepAspectRatio, String outputFormat, String outputFormatType, float
outputQuality, int imageType, List<ImageFilter> filters, ResizerFactory resizerFactory, Boolean
WithinDimensions, boolean useExifOrientation)

private void validateThumbnailSize()
private void validateScalingFactor()
public Dimension getSize()
public double getWidthScalingFactor()
public double getHeightScalingFactor()
public int getType()
public boolean isKeepAspectRatio()

public ThumbnailParameter(Dimension thumbnailSize, Region sourceRegion, boolean keepAspectRatio, String
outputFormat, String outputFormatType, float outputQuality, int imageType, List<ImageFilter> filters, Resizer
resizer, boolean fitWithinDimensions, boolean useExifOrientation)
```