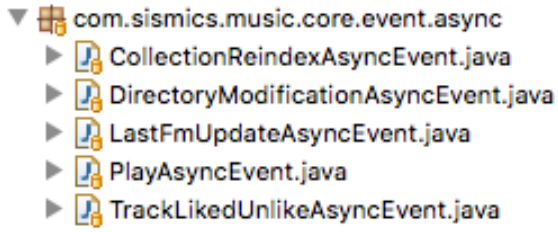
Task 3A

---

package com.sismics.music.core.event.async;

- After refactoring, the package looks like this.



- Number of classes has gone down from 9 to 5.
- DirectoryCreatedAsyncEvent and DirectoryDeletedAsyncEvent were two classes that contained the same code. Just the name of the class was different. Therefore those two were combined to make a single class whose name was DirectoryModificationAsyncEvent.
- Similarly the classes which where combined are -
  - LastFmUpdateLovedTrackAsyncEvent and LastFmUpdateTrackPlayCountAsyncEvent → LastFmUpdateAsyncEvent
  - PlayCompletedEvent and PlayStartedEvent → PlayAsyncEvent
  - TrackLikedAsyncEvent and TrackUnlikedAsyncEvent → TrackLikedUnlikeAsyncEvent
- These event classes were used in corresponding event buses and were consumed by corresponding listeners. Therefore the references in those classes have to be changed.
- This refactoring therefore involved changes in music-web package as well.

---

package com.sismics.music.core.service.collection;

CollectionService

- addDirectoryToIndex method was shortened by creating three private methods -
  - indexDirectory
  - deleteNonExistingTracks
  - cleanEmptyAlbums

```
public void addDirectoryToIndex(Directory directory)
{
    if (log.isInfoEnabled())
    {
        log.info(MessageFormat.format("Adding directory {0} to index", directory.getLocation()));
    }
    CollectionVisitor collectionVisitor = indexDirectory(directory);
    deleteNonExistingTracks(directory, collectionVisitor);
    cleanEmptyAlbums();
    ArtistDao artistDao = new ArtistDao();
    artistDao.deleteEmptyArtist();

    if (log.isInfoEnabled())
    {
        log.info(MessageFormat.format("Done adding directory {0} to index", directory.getLocation()));
    }
}
```

- readTrackMetadata - The code went down from 90 lines to 20 lines.

- handleMissingArtist() was created to handle the case when albumArtist was not specified.
- The initTrack() method was created to initialize other parameters of the track. This initTrack method calls other private methods to initialize some track parameters. These methods are -
  - setTrackAlbum
  - setTrackArtist
  - setTrackYear
  - setTrackOrder
  - handleMissingTag

```
public void readTrackMetadata(Directory rootDirectory, Path file, Track track) throws Exception
{
    Path parentPath = file.getParent();
    DirectoryNameParser nameParser = new DirectoryNameParser(parentPath);
    String albumArtistName = StringUtils.abbreviate(nameParser.getArtistName(), 1000).trim();
    String albumName = StringUtils.abbreviate(nameParser.getAlbumName(), 1000).trim();
    ArtistDao artistDao = new ArtistDao();

    AudioFile audioFile = AudioFileIO.read(file.toFile());
    Tag tag = audioFile.getTag();

    // The album artist can't be null, check is in the directory name parser
    Artist albumArtist = artistDao.getActiveByName(albumArtistName);
    if (albumArtist == null)
    {
        albumArtist = handleMissingArtist(albumArtistName, artistDao);
    }

    initTrack(rootDirectory, file, track, parentPath, albumName, artistDao, audioFile, tag, albumArtist);
}

private void initTrack(Directory rootDirectory, Path file, Track track, Path parentPath, String albumName,
        ArtistDao artistDao, AudioFile audioFile, Tag tag, Artist albumArtist) throws Exception
{
    if (tag == null)
    {
        handleMissingTag(file, track, albumArtist);
    }
    else
    {
        AudioHeader header = audioFile.getAudioHeader();

        track.setLength(header.getTrackLength());
        track.setBitrate(header.getSampleRateAsNumber());
        track.setFormat(StringUtils.abbreviate(header.getEncodingType(), 50));
        track.setVbr(header.isVariableBitRate());

        setTrackOrder(track, tag);
        setTrackYear(track, tag);
        // Track title (can be empty string)
        track.setTitle(StringUtils.abbreviate(tag.getFirst(FieldKey.TITLE), 2000).trim());
        setTrackArtist(track, artistDao, tag);
    }
    setTrackAlbum(rootDirectory, file, track, parentPath, albumName, albumArtist);
}
```

CollectionWatchService

- startUp() method was modularized as follows by creating createWatchService and watchAllDir private methods.

```
@Override
protected void startUp() {
    TransactionUtil.handle(() -> {
        createWatchService();
        watchAllDir();
    });
}

private void watchAllDir()

private void createWatchService()
```

- run() method was refactored by creating handlePath private method.

```
@Override
protected void run() throws Exception {
    while (isRunning()) {
        WatchKey watchKey = watchService.take();

        Path dir = watchKeyMap.get(watchKey);
        if (dir == null) {
            continue;
        }

        handlePath(watchKey, dir);

        cleanupOrphans();

        if (!watchKey.reset()) {
            watchKeyMap.remove(watchKey);
        }
    }
}

private void handlePath(WatchKey watchKey, Path dir) throws IOException
```

---

package com.sismics.music.core.service.importaudio;

ImportAudioService

- run() method is 84 lines long. It was refactored by creating the following private methods -
  - startYtDownload
  - updateImportStatus
  - finishProcess

```
@Override
protected void run() throws Exception
{
    while (isRunning())
    {
        // Wait for a new URL to import
        ImportAudio importAudio = importAudioQueue.take();

        try
        {
            log.info("Start importing a new URL: " + importAudio);
            Process process = startYtDownload(importAudio);

            // Reading standard output to update import status
            try (BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream())))
            {
                updateImportStatus(importAudio, process, reader);
            }
            finishProcess(importAudio, process);
        }
        catch (Exception e)
        {
            // Error while importing
            importAudio.setStatus(ImportAudio.Status.ERROR);
            importAudio.setMessage(importAudio.getMessage() + "\n" + e.getMessage());
            log.error("Error importing: " + importAudio, e);
        }
    }
}

private void finishProcess(ImportAudio importAudio, Process process) throws InterruptedException

private void updateImportStatus(ImportAudio importAudio, Process process, BufferedReader reader) throws IOException

private Process startYtDownload(ImportAudio importAudio) throws IOException
```

- importFile method was shortened by creating unzipFiles private method.

```
public void importFile(File file) throws Exception
{
    String mimeType = MimeTypeUtil.guessMimeType(file);
    String ext = Files.getFileExtension(file.getName()).toLowerCase();
    String importDir = DirectoryUtil.getImportAudioDirectory().getAbsolutePath();

    if (MimeType.APPLICATION_ZIP.equals(mimeType))
    {
        log.info("Importing a ZIP file");
        // It's a ZIP file, unzip accepted files in the import folder
        unzipFiles(file, importDir);
    }
    else if (Constants.SUPPORTED_AUDIO_EXTENSIONS.contains(ext))
    {
        // It should be a single audio track
        File outputFile = new File(importDir + File.separator + file.getName());
        log.info("Importing a single track: " + outputFile);
        Files.copy(file, outputFile);
    }
    else
    {
        throw new Exception("File not supported");
    }
}

private void unzipFiles(File file, String importDir) throws IOException, FileNotFoundException
```

---

package com.sismics.music.core.listener.async;

- The classes involved in the package had missing abstraction smell, because they did similar kind of task.
- For this AbstractAsyncListener was created. Following classes were made to extend this class -
  - CollectionReindexAsyncListener
  - DirectoryCreatedAsyncListener
  - DirectoryDeletedAsyncListener
  - LastFmUpdateLovedTrackAsyncListener
  - LastFmUpdateTrackPlayCountAsyncListener
  - TrackLikedAsyncListener
  - TrackUnlikedAsyncListener

```
public abstract class AbstractAsyncListener<T>
{
    private final Logger log = LoggerFactory.getLogger(this.getClass());

    private Stopwatch doBefore(T t)
    {
        if (log.isInfoEnabled())
        {
            log.info(t.getClass() + ": " + t.toString());
        }
        return Stopwatch.createStarted();
    }

    private void doAfter(T t, Stopwatch stopwatch)
    {
        if (log.isInfoEnabled())
        {
            log.info(MessageFormat.format(t.getClass() + " completed in {0}", stopwatch));
        }
    }

    protected abstract void handleInternal(T t);

    @Subscribe
    public void handle(final T t) throws Exception
    {
        Stopwatch stopwatch = doBefore(t);
        handleInternal(t);
        doAfter(t, stopwatch);
    }
}
```

- Every class first logs something, creates a stopwatch, performs some task and then finally logs again using the stopwatch object created.
- Therefore a handle() method was created in AbstractListenerAsync class. This method calls the abstract method - handleInternal method.
- The handleInternal method is an abstract method which needs to be overridden by every listener class that extends AbstractListenerAsync class.
- Ex -

```
public class DirectoryCreatedAsyncListener extends AbstractAsyncListener<DirectoryModificationAsyncEvent>
{

    @Override
    protected void handleInternal(DirectoryModificationAsyncEvent directoryModificationAsyncEvent)
    {
        final Directory directory = directoryModificationAsyncEvent.getDirectory();

        TransactionUtil.handle(() -> {
            // Index new directory
            CollectionService collectionService = AppContext.getInstance().getCollectionService();
            collectionService.addDirectoryToIndex(directory);

            // Watch new directory
            AppContext.getInstance().getCollectionWatchService().watchDirectory(directory);

            // Update the scores
            collectionService.updateScore();
        });

    }
}
```

---

package com.sismics.music.core.service.albumart;

AlbumArtService.java

- importAlbumArt method was refactored as follows -

```
public void importAlbumArt(Album album, File originalFile, boolean copyOriginal) throws
Exception {
        ImageUtil.FileType fileType = ImageUtil.getFileFormat(originalFile);
        if (fileType == null) {
            throw new Exception("Unknown file format for picture " +
originalFile.getName());
        }
        BufferedImage originalImage =
ImageUtil.readImageWithoutAlphaChannel(originalFile);

        String id = UUID.randomUUID().toString();
        for (AlbumArtSize albumArtSize : AlbumArtSize.values()) {
            importAlbumArt(id, originalImage, albumArtSize);
        }

        // Update the album
        album.setAlbumArt(id);

        if (copyOriginal) {
            // Copy the original file to the album directory
            Path albumArtPath = Paths.get(album.getLocation(), "albumart.jpg");
            File albumArtFile = albumArtPath.toFile();
            try {
                ImageUtil.writeJpeg(originalImage, albumArtFile);
            } catch (Exception e) {
                throw new NonWritableException(e);
            }
        }
    }
```

```
public void importAlbumArt(Album album, File originalFile, boolean
copyOriginal) throws Exception {
        ImageUtil.FileType fileType = ImageUtil.getFileFormat(originalFile);
        if (fileType == null) {
            throw new Exception("Unknown file format for picture " +
originalFile.getName());
        }
        BufferedImage originalImage =
ImageUtil.readImageWithoutAlphaChannel(originalFile);

        String id = UUID.randomUUID().toString();
        for (AlbumArtSize albumArtSize : AlbumArtSize.values()) {
            importAlbumArt(id, originalImage, albumArtSize);
        }

        // Update the album
        album.setAlbumArt(id);

        if (copyOriginal) {
            copyFile(album, originalImage);
        }
    }
```

package com.sismics.music.core.service.collection;

CollectionService.java

Long methods were found and were refactored. Those methods were -
- addDirectoryToIndex()
- readTrackMetaData()

Example -

REFACTORED CODE

```
public void readTrackMetadata(Directory rootDirectory, Path file, Track track) throws
Exception
	{
		Path parentPath = file.getParent();
		DirectoryNameParser nameParser = new DirectoryNameParser(parentPath);
		String albumArtistName = StringUtils.abbreviate(nameParser.getArtistName(),
1000).trim();
		String albumName = StringUtils.abbreviate(nameParser.getAlbumName(),
1000).trim();
		ArtistDao artistDao = new ArtistDao();

		AudioFile audioFile = AudioFileIO.read(file.toFile());
		Tag tag = audioFile.getTag();

		// The album artist can't be null, check is in the directory name parser
		Artist albumArtist = artistDao.getActiveByName(albumArtistName);
		if (albumArtist == null)
		{
			albumArtist = handleMissingArtist(albumArtistName, artistDao);
		}

		initTrack(rootDirectory, file, track, parentPath, albumName, artistDao,
audioFile, tag, albumArtist);

	}
```

OLD CODE

```
public void readTrackMetadata(Directory rootDirectory, Path file, Track track) throws
Exception {
        Path parentPath = file.getParent();
        DirectoryNameParser nameParser = new DirectoryNameParser(parentPath);
        String albumArtistName = StringUtils.abbreviate(nameParser.getArtistName(),
1000).trim();
        String albumName = StringUtils.abbreviate(nameParser.getAlbumName(),
1000).trim();
        ArtistDao artistDao = new ArtistDao();

        AudioFile audioFile = AudioFileIO.read(file.toFile());
        Tag tag = audioFile.getTag();

        // The album artist can't be null, check is in the directory name parser
        Artist albumArtist = artistDao.getActiveByName(albumArtistName);
        if (albumArtist == null) {
            albumArtist = new Artist();
            albumArtist.setName(albumArtistName);
            artistDao.create(albumArtist);
        }

        if (tag == null) {
            // No tag available, use filename as title and album artist as artist, and
guess the rest

track.setTitle(Files.getNameWithoutExtension(file.getFileName().toString()));
```

```
            track.setArtistId(albumArtist.getId());
            track.setLength((int) (file.toFile().length() / 128000));
            track.setBitrate(128);
            track.setFormat(Files.getFileExtension(file.getFileName().toString()));
            track.setVbr(true);
        } else {
            AudioHeader header = audioFile.getAudioHeader();

            track.setLength(header.getTrackLength());
            track.setBitrate(header.getSampleRateAsNumber());
            track.setFormat(StringUtils.abbreviate(header.getEncodingType(), 50));
            track.setVbr(header.isVariableBitRate());

            String order = tag.getFirst(FieldKey.TRACK);
            if (!Strings.isNullOrEmpty(order)) {
                try {
                    track.setOrder(Integer.valueOf(order));
                } catch (NumberFormatException e) {
                    // Ignore parsing errors
                }
            }

            String year = tag.getFirst(FieldKey.YEAR);
            if (!Strings.isNullOrEmpty(year)) {
                try {
                    track.setYear(Integer.valueOf(year));
                } catch (NumberFormatException e) {
                    // Ignore parsing errors
                }
            }

            // Track title (can be empty string)
            track.setTitle(StringUtils.abbreviate(tag.getFirst(FieldKey.TITLE),
2000).trim());

            // Track artist (can be empty string)
            String artistName = StringUtils.abbreviate(tag.getFirst(FieldKey.ARTIST),
1000).trim();
            Artist artist = artistDao.getActiveByName(artistName);
            if (artist == null) {
                artist = new Artist();
                artist.setName(artistName);
                artistDao.create(artist);
            }
            track.setArtistId(artist.getId());
        }

        // Track album
        AlbumDao albumDao = new AlbumDao();
        Album album = albumDao.getActiveByArtistIdAndName(albumArtist.getId(),
albumName);
        if (album == null) {
            // Import album art
            AlbumArtImporter albumArtImporter = new AlbumArtImporter();
            File albumArtFile = albumArtImporter.scanDirectory(file.getParent());

            album = new Album();
            album.setArtistId(albumArtist.getId());
            album.setDirectoryId(rootDirectory.getId());
            album.setName(albumName);
            album.setLocation(file.getParent().toString());
            if (albumArtFile != null) {
                // TODO Remove this, albumarts are scanned separately
                AppContext.getInstance().getAlbumArtService().importAlbumArt(album,
albumArtFile, false);
            }
            Date updateDate = getDirectoryUpdateDate(parentPath);
            album.setCreateDate(updateDate);
            album.setUpdateDate(updateDate);
            albumDao.create(album);
        }
        track.setAlbumId(album.getId());
```

```
    }
```

CollcetionWatchService.java

- The run() method of the class was shortened by creating a private method called handlePath.

```
    protected void run() throws Exception {
        while (isRunning()) {
            WatchKey watchKey = watchService.take();

            Path dir = watchKeyMap.get(watchKey);
            if (dir == null) {
                continue;
            }

            handlePath(watchKey, dir);

            cleanupOrphans();

            if (!watchKey.reset()) {
                watchKeyMap.remove(watchKey);
            }
        }
    }
```

---

package com.sismics.music.core.service.importaudio;

ImportAudioService.java

- The run method was shortened by creating private method -
  - startYtDownload
  - updateImportStatus
  - finishProcess
- The updated method looks like this. The length of the code went down from 84 to 28 lines.

```
@Override
protected void run() throws Exception
{
    while (isRunning())
    {
        // Wait for a new URL to import
        ImportAudio importAudio = importAudioQueue.take();

        try
        {
            log.info("Start importing a new URL: " + importAudio);
            Process process = startYtDownload(importAudio);

            // Reading standard output to update import status
            try (BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream())))
            {
                updateImportStatus(importAudio, process, reader);
            }
            finishProcess(importAudio, process);
        }
        catch (Exception e)
        {
            // Error while importing
            importAudio.setStatus(ImportAudio.Status.ERROR);
            importAudio.setMessage(importAudio.getMessage() + "\n" + e.getMessage());
            log.error("Error importing: " + importAudio, e);
        }
    }
}
```

- importFiles method was shortened by creating unzipFiles() private method. Details can be found in the code itself.

Task 3B

Application Metrics

Note: Further Application Statistics are available.

Logical lines of Code
5 724
0 (NotMyCode)
Estimated Dev Effort 144d

Types
284
3 Projects
63 Packages
1 415 Methods
485 Fields
205 Source Files
1 742 Third-Party Elements

Comment
16.14%
1 102 Lines of Comment

Debt
8.32%
Rating B 4d 6h effort to reach A
Debt 11d 7h
The technical-debt is incomplete because no coverage data specified.

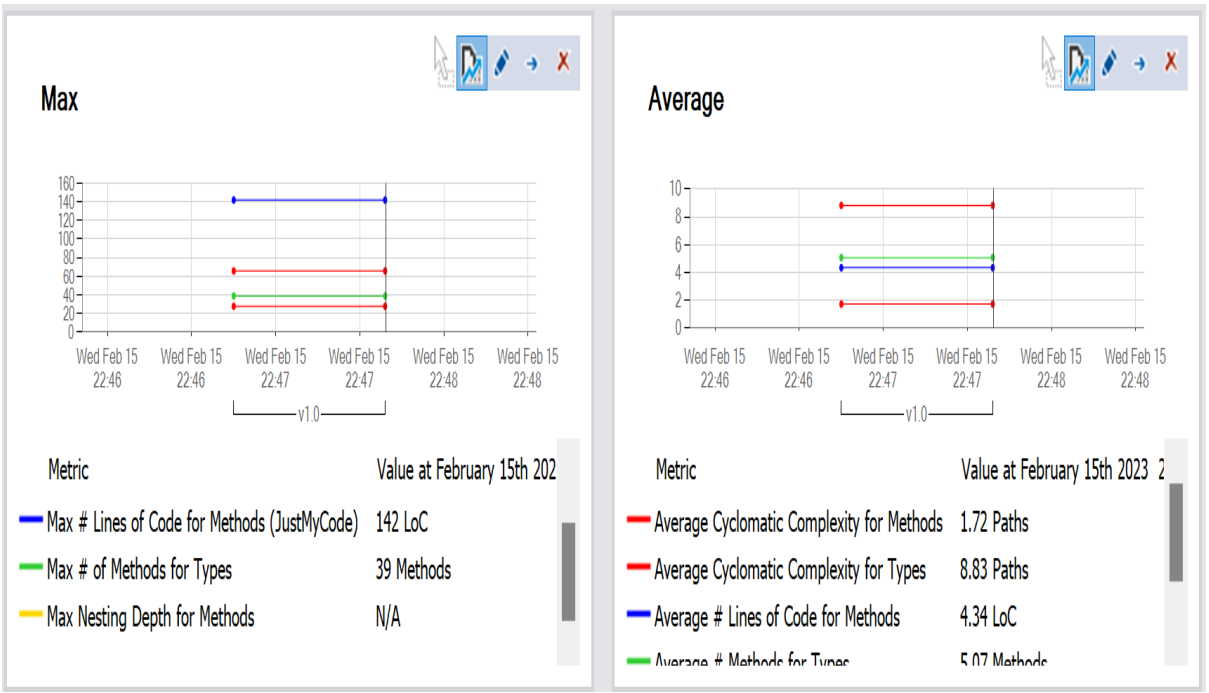
Coverage
N/A because no coverage data specified

Method Complexity
28 Max
1.66 Average

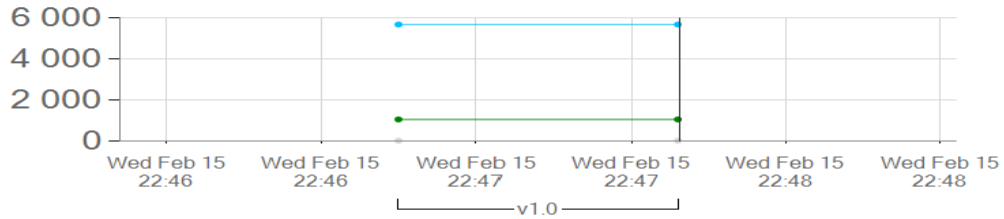
Quality Gates
Fail	2
Warn	0
Pass	3

Rules
Critical	1
Violated	61
Ok	145

Issues
All	1 232
Blocker	0
Critical	0
High	328
Medium	112
Low	792

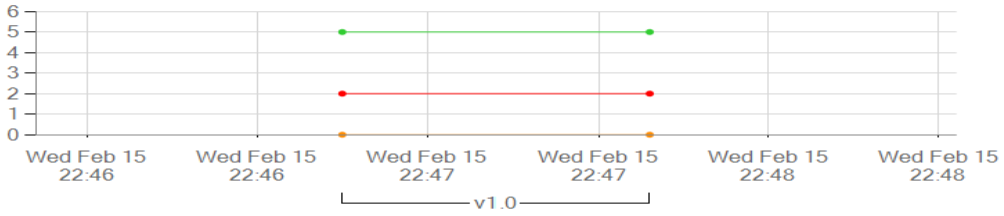


Lines of Code



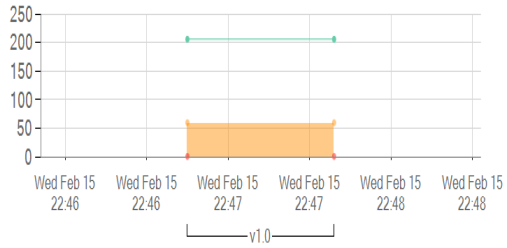
Metric	Value at February 15th 2023 22:47 (v1.0)
# Lines of Code	5 656 Loc
# Lines of Code (NotMyCode)	0 Loc
# Lines of Comments	1 038 Lines

Quality Gates



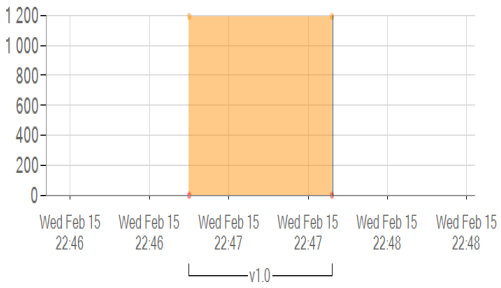
Metric	Value
# Quality Gates Fail	
# Quality Gates Warn	
# Quality Gates	

Rules Violated



Metric	Value
# Rules	
# Rules Violated	
# Critical Rules Violated	

Rules Violations



Metric	Value at February 15th 2023 22:47 (v1.0)
# Rules Violations	1 193 Violations
# Critical Rules Violations	2 Violations