Jeremy Aguilon
CS 4590

# Homework Assignment Writeup

**Quick note**: I found the mbrola voices way more pleasant than the default tts package. However, this requires extra installation. If this doesn't work, then feel free to remove the string in line 69 of notification_server_v1_1.pde (`tts = new TTS("mbrola_us1");`

## Architecture Summary: Life of a Notification

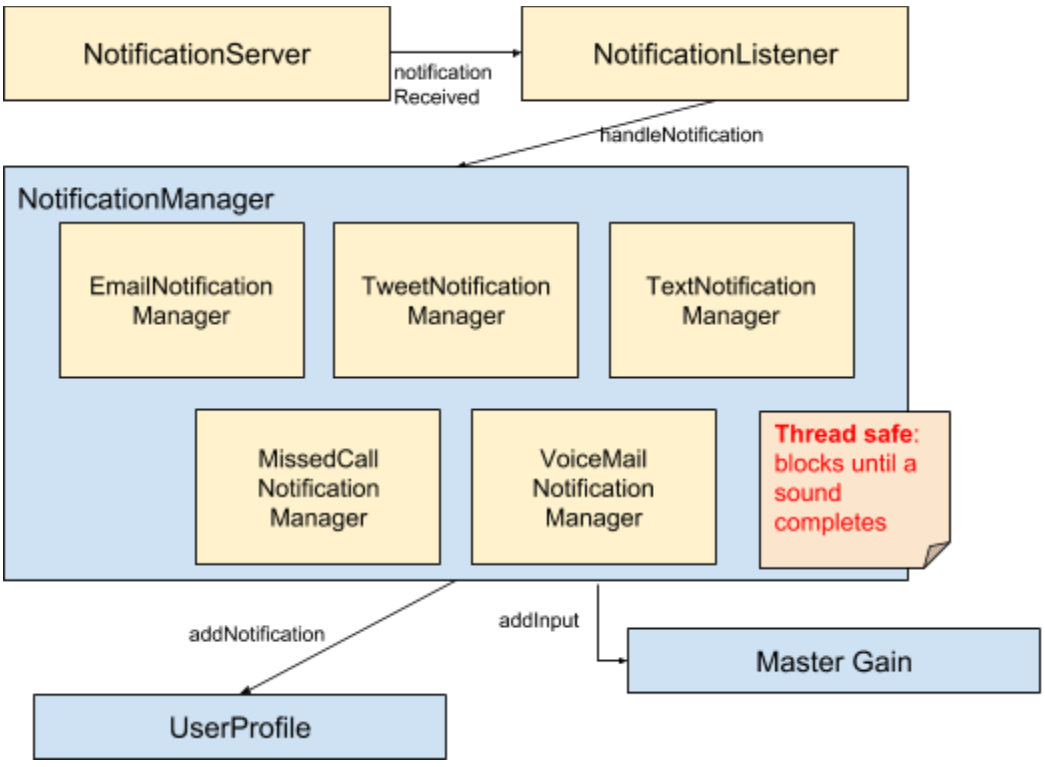



*Fig. 1: High-level lifespan of a notification*

When I was wrapping my head around this project, I came up with the following requirements:

- **Thread safety**: code should block until a sound is complete. This way, if an important notification comes while a sound is completing, they won't clutter the user's sound.
    - The **notification report** should similarly block all other notifications
    - The **device status sounds** (i.e. low battery) shouldn't overlap with notifications either
- **Abstraction**: A different class should handle each type of notification, which separates out policies
- **Temporal accessibility**: Important notifications (i.e. those that resulted in a sound) should be saved in a user profile. These sounds will be used for the notification report

With these policies in mind, I created the architecture in fig. 1. From a high level:

1. The notification listener receives a notification. If a notification is being processed already, then the thread blocks until it is available
2. A NotificationManager singleton is retrieved according to the type of the notification, and the notification is passed to this singleton
3. The manager takes into account the context of the environment, the notification, and the sender to decide what sound to play (if at all).
    a. If a sound plays, then the notification is saved in a user profil

# Notification Sound Descriptions

Clearly, some notification types are normally more important than others. For example, most tweets are less important than a missed call. However, some *instances* of a notification type also vary. For example, a missed call from a parent is usually more urgent than a missed call from a telemarketer. Thus, I found it useful to gather urgent noises for certain notification types.

Notification types with an urgent noise option follow a consistent sound profile. **Default noises** are shorter with a lower pitch while **urgent noises** last longer and have a climbing pitch. This lets the user easily associate both default and urgent noises to the same notification type.

To make the sounds more learnable, I made sure that both urgent and default variations of a notification type are easy to associate. To test the notifications, I played my peers an example stream and found that they could accurately understand the notifications after a few run-throughs.

*Table 1: Description of notification noises*

| **Notification Type** | **Sound category** | **Default Noise Description** | **Urgent Noise Description** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Email | Auditory Icon | A gentle chime of a medium-pitched bell | A doorbell “ding dong” recording |
| Tweet | Auditory Icon | A brief bird chirp sample | |
| Missed Call | Earcon | An 8-bit tune that falls in pitch | A louder 8-bit tune that rises in pitch |
| Text | Earcon | An abstract but familiar “swiping” earcon | |
| Voicemail | Earcon | A pleasant rising chime, reminiscent of chat apps | |

# Notification Policy

The implementations of NotificationManager instances decide what kind of sound to output to the user. When a NotificationManager receives a Notification, it decides whether to play a sound and what sound to play. From a high level, I used the following parameters to make decisions:

- **Context**: the environment of the user
- **Priority**: how urgent the notification appears to be
- **Is Best Friend**: whether the sender is a “best friend” (heuristically set to “Sister,” “best friend,” and “mom”)
- **Retweets**: how viral a tweet is
- **Content summary**: the sentiment of the notificatoin
- **Spamminess**: Whether an email has spammy characteristics

Overall, there are three different sounds that may play for a given notification:

1. A synthesized sine wave beep: Used for **urgent** friends from best friends, regardless of any other parameter
2. A text-to-speech commentary: Used for emails, texts, and voice mail for appropriate contexts
3. An auditory icon/earcon (see table 1).

Tables 2-6 dictate the specific heuristics for each NotificationManager.

*Table 2: Email Notification Manager*

| **Context** | **Notification Predicate Pseudocode** |
|---|---|

| | |
|---|---|
| Party | `if (not spammy and priority = 1) then  default email notification` |
| Lecture | `if (not spammy and priority = 1) then default email notification` |
| Jogging | `if (not spammy and priority <= 2 and is best friend) then urgent email notification`<br><br>`if (not spammy and priority = 1) then default email notification` |
| Public Transit | `if (spammy) then text-to-speech “Spam warning”`<br><br>`if (priority level <= 2) then urgent email notification`<br>`    * Also include a tts “Good news” if appropriate`<br><br>`else default email notification` |

*Table 3: Tweet Notification Manager*

| **Context** | **Notification Predicate Pseudocode** |
|---|---|
| Party | `if (priority = 1 and retweets > 1000) then tweet notification` |
| Lecture | `N/A` |
| Jogging | `if (priority <= 2 and is best friend) then tweet notification`<br><br>`if (priority = 1 and retweets > 1000) then tweet notification` |
| Public Transit | `if (is best friend) then tweet notification`<br><br>`if (priority level <= 2) then tweet notification` |

*Table 4: Text Notification Manager*

| **Context** | **Notification Predicate Pseudocode** |
|---|---|
| Party | `if (priority <= 2) then default text notification` |
| Lecture | `N/A` |
| Jogging | `if (priority <= 2 or is best friend) then default text notification`<br>`    * Also include a tts “Good/bad news” if appropriate` |
| Public Transit | `if (true) then default text notification` |

*Table 5: Missed Call Notification Manager*

| **Context** | **Notification Predicate Pseudocode** |
|---|---|

| | |
|---|---|
| Party | `if (priority <= 2 and is best friend) then urgent missed call notification`<br><br>`if (priority = 1) then default missed call` |
| Lecture | `N/A` |
| Jogging | `if (priority <= 2 and is best friend) then urgent missed call`<br><br>`if (priority = 1) then default missed call` |
| Public Transit | `if(priority <= 2) then urgent missed call`<br><br>`else default missed call` |

*Table 6: Voice Mail Notification Manager*

| **Context** | **Notification Predicate Pseudocode** |
|---|---|
| Party | `if (priority <= 2 and is best friend) then voicemail chime` |
| Lecture | `N/A` |
| Jogging | `if (is best friend or priority <= 2) then voicemail chime` |
| Public Transit | `if (is best friend or priority <= 3) then voicemail chime`<br>`* Also include a tts "Good/bad news" if appropriate` |

# Notification Summary

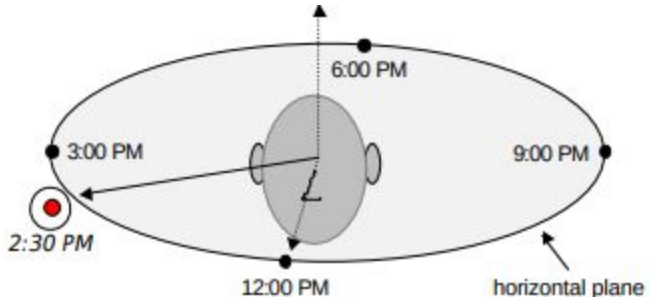
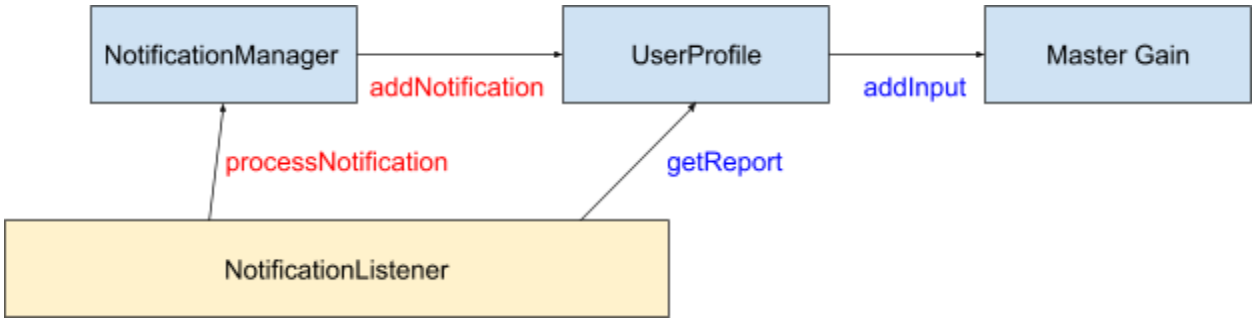



*Fig 2: Nomadic radio notification scheme for*

According to the Nomadic Radio Paper by Nitin Sawhney and Chris Schmandt, they map "incoming message on a chronological 12-hour spatial audio display. I wanted to have a similar temporal feature to this approach. Unlike the Nomadic Radio example, I had the following limitations:

- The audio files are not binaural, and Beads does not seem to support simulating this feature.
- There is no concept of "real-world" time on the JSON files; instead, there are simply milliseconds since the file was read.




Given these limitations, I settled from a gradual pan from left to right. As notification sounds are processed by the NotificationManager, I save the notifications in a UserProfile instance. This user profile has a `getReport` method that sonifies the earliest messages on the left ear, and notifications gradually move towards the right.

Since the summary takes a lot of concentration to understand, I block all incoming notifications until the summary is complete. Due to thread safety, the notifications are processed cleanly afterward.

# Appendix

## Used Files

| **Sound** | **OcenAudio Modifications** |
|---|---|
| Party Context Track | Created a seamless loop |
| Subway Context Track | Created a seamless loop |
| Jogging Context Track [track one, track two] | Overlaid both tracks in a seamless loop to create the impression of running by a road |
| Twitter Chirp | |
| Default Email Bell | |
| Urgent Email Bell | Pitch-shifted so that the notes would climb from low to high |

| | |
|---|---|
| Default Text Earcon | |
| Default Missed Call | |
| Urgent Missed Call | |
| Battery Notifications | Heavily edited to create a “critical,” “low,” “medium,” and “high” notification with different pitches |
| Connection Notifications | Heavily edited to create a “low,” “medium,” and “high” notification with different pitches |