CS212: Paradigms
Dr. Gross, Spring 2016

Project 1: Sonic the Hedgehog

1. In your **homework repo**, create a new package called `edu.blackburn.cs.cs212sp16.lastname.sonic`
2. Save a new UML file in WhiteStarUML called sonic-lastname.uml into this directory
3. Add and commit, add and commit, live it, breathe it, add and commit

Lucas and Jordan inspired this project. Because you all love the mp3s with the music and the rap music so much, we're going to make a simple media player that will play mp3s and other audio. The good news is that many people have written many libraries for decoding many different kinds of audio formats. The bad news is that many people have written many libraries for decoding many different kinds of audio formats.

Part 0: Initial Design (10 out of 100, due Mon, Feb 22 by 11:00am)

In your class diagram you should create a design that can handle the following:

- User interface
- File handling
- Saving and reading faves and replays
- Open, read, and play a file (which might be one of several *kinds* of formats, hint hint)

Part I: Libraries (20 out of 100, due Thu, Feb 25 by 5:00pm)

Do you want to test every library for every kind of audio file out there? I wouldn't, so I'm going to allow you to work as a class to test and review libraries. You **cannot share code**, but you can divide up all of the libraries. How you split them is up to you to solve as a class (or split into factions and compete).

Everyone must successfully test and review two libraries (have at least two people test each library, and put your test code in your **homework repo** in the package `edu.blackburn.cs.cs212sp16.lastname.sonic.testcode`). Java libraries are usually in files that end with .jar[1]. Include at least four kinds of audio formats. Upload the jars to the **lab repo** in Documents\Project1, along with instructions (in a text file) on how to configure the library into a project, and email me a brief review of each, grading it from A to F.

To be completely clear, you need to have:

- A jar file *in the lab repo* for each of two (or more) libraries for reading audio formats in the **lab repo**
- A file (text or PDF) alongside the jar explaining how to add it to the project
- One Runner in your testcode subpackage **in your homework repo** for each library that will play an audio file (this is test code, so you can use static all you want)

Want some files to test? Here's a suggestion: https://archive.org/details/TheWhisper
DO NOT PUT THESE IN ANY REPO (they're multiple MB)

[1] If you find something else, talk to Dr. Gross ASAP

Edit: How do you find libraries? (Question per Lucas Burdell)
Some critical search terms include: java, library, open-source and "open source", sound, music, format (and the name of the format), compression, decompression, decoder.

Part II: Start at the Very Beginning (20 out of 100, due Mon, Feb 29 by 11:00am)

I know you can do the next parts, so next you need to start implementing the core part of your design. If your design doesn't have inheritance in it, your design is wrong. You will need classes to talk to each kind of file format, but *you might want to guarantee that they all function in the same way*. The internal implementation of the decoder will differ depending on the file type and library. Assume that the file extension is correct.

Your code should take an argument of the path to a directory where there are multiple files of different types. It should read the contents of the directory open each file in sequence, play the file (using the correct decoder). Make sure your code implements your design or a modification. No static (other than main)!

You'll need to set a project default Main Class and arguments. Right click on your project, select Properties, select Run, the click Browse to select your class with a main method. Then add the appropriate path to Arguments. How do you read that in? It's in the args array passed to the main method. (You cannot Run File here because it will not pass that argument.) Don't forget to check if an argument is passed in!

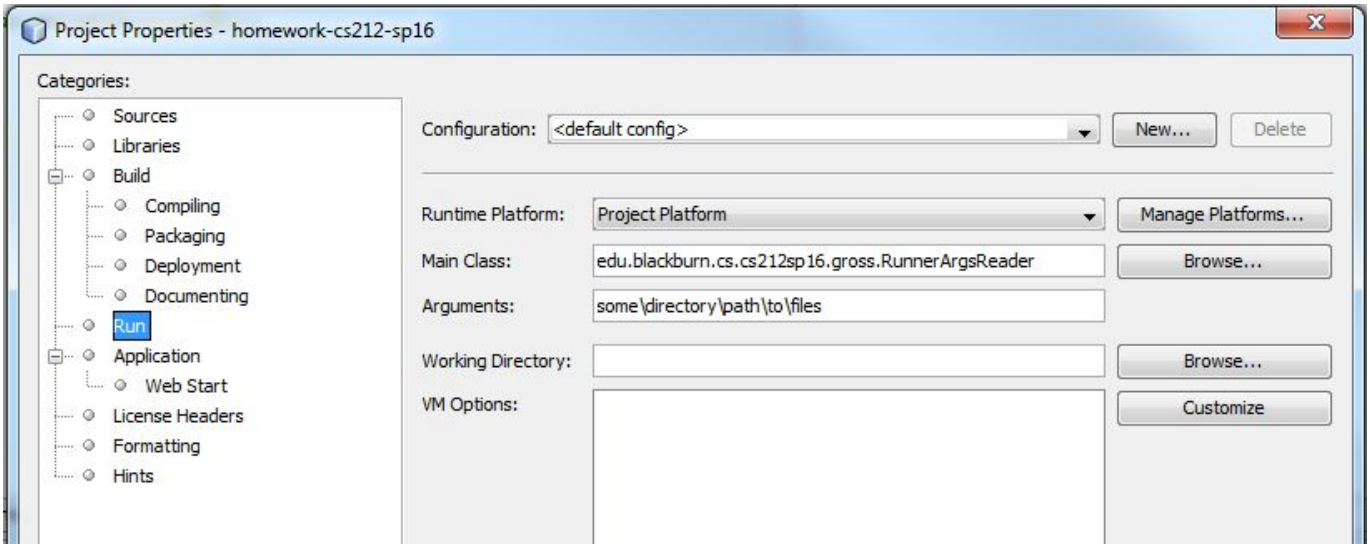

Figure 1. Setting up the project

Part III: Logan's Run (5 out of 100, due Friday, Mar 4 by 11:00am)

Now you need to write the rest of the classes. You should set up your directories with the topmost directory being artists, each subdirectory under artist be an album, and each file under an album should be a song. Your interface should start by reading in a path passed in from the main method. It should read in all filenames, and it should add the ones it can decode into two lists: Artists (all songs by a given artist), Albums (all songs in a given album). You will also need to read in files containing a list of favorites and a list of recently played songs. The interface should look like this:

```
No, I'm not naming this thing Music Player 3000, but welcome nonetheless

Type s for silent (don't actually play files), or hit return for normal play:

Please choose a number or 0 to quit:
1. Artists
2. Albums
3. Favorites
4. Recently played
> 1    [entered by the user]

Please select an artist or 0 to return to the prior menu
1. Acid Horse
2. Alanis Morissette
3. Alice in Chains
4. Alphaville
5. Billie Holiday
6. Bjork
…
> 4

Please select a song or 0 to return to the main menu:
1. Big in Japan
2. Forever Young
> 2

Please select an option or 0 to return to the main menu
1. Play Forever Young
2. Add Forever Young to Favorites
> 1

Playing Forever Young [then play it]
Done

Please select an option or 0 to return to the main menu:
1. Play Forever Young
2. Add Forever Young to Favorites
> 2

Added Forever Young to favorites
Please select an option or 0 to return to the main menu:
1. Play Forever Young
2. Add Forever Young to Favorites
> 0

Please choose a number or 0 to quit:
1. Artists
2. Albums
3. Favorites
```

```
4. Recently played
> 3

Please choose a favorite or 0 to return to the main menu:
1. No Name, No Slogan
2. Where Would I Be Without IBM
3. Forever Young
> 0

Please choose a number or 0 to quit:
1. Artists
2. Albums
3. Favorites
4. Recently played
> 4

Please choose a recently played song or 0 to return to the main menu:
1. Forever Young
2. Beds Are Burning
3. Satyagraha
… [the rest of the songs played, most recent first, least recent last]
> 0

Please choose a number or 0 to quit:
1. Artists
2. Albums
3. Favorites
4. Recently played
> 0

Goodbye!
```

Figure 2. Example output

Remember: Add. Commit. Commit. Push. Commit. Commit. Commit. Add. Commit. Push (you get the idea)

Rubric

Design (Class Diagram)	
All four areas covered	5
Good design using inheritance appropriately, at least twice	5
Libraries	
Library 1 working code	5
Library 1 shared jar and instructions	5
Library 2 working code	5
Library 2 shared jar and instructions	5
First Code Submit	
At least four libraries/formats used	5
Proper OO code	5
Good design	5
Inheritance/polymorphism used correctly	5
Final	
Standards/comments	5
Time estimate/accounts	5
Artist, Album, ArtistList, and AlbumList are correct	10
Inheritance/polymorphism used correctly	10
Favorites works	10
Recently Played works	5

Closing Thoughts:

1. No, you don't know how to do all of this. That's kinda the point. Or rather, that is the point.
2. If I am reminded (should I forget) via email, I will email one daily hint
3. Tim, Lucas, and I are available for help; plan to make use of us
4. Silent mode will help you out immensely; it's a gimme