

 [cs1302uga](#) / [cs1302-omega](#)

This document contains the description for the cs1302-omega project assigned to the students in CSCI 1302 at the University of Georgia.


 github.com/cs1302uga/cs1302-omega

☆ 0 stars  0 forks

☆ Star


 Watch ▾

<> Code

 Pull requests

 Actions

 Security

 Insights

 main ▾

...



mepcoterrell Update README.rst ...

3 days ago  4

[View code](#)

CSCI 1302 - Omega v2021.sp

Approved for **Spring 2021** Team Size **1**



This document contains the description for the **cs1302-omega** project assigned to the students in the Spring 2021 CSCI 1302 classes at the University of Georgia.

Please read the entirety of this file before beginning your project.

Contents

- [Deadlines](#)
- [Academic Honesty](#)
- [Updates](#)
- [Project Description](#)
- [Learning Outcomes](#)
- [Functional Requirements](#)
- [Non-Functional Requirements](#)
- [Final Project Policies](#)
- [How to Download the Project](#)
- [Submission Instructions](#)
 - [Deadline Option 1: SAT 2021-05-08 \(MAY 08\) @ 03:30 PM EST](#)
 - [Deadline Option 2: SUN 2021-05-09 \(MAY 09\) @ 03:30 PM EST](#)
 - [Deadline Option 3: MON 2021-05-10 \(MAY 10\) @ 03:30 PM EST](#)
- [Appendix](#)
- [FAQ](#)

Deadlines

This project has **three deadlines options**. Students who perform their final submission via the submit command before the date/times listed below automatically receive the associated Submission-Based (SB) extra credit. The amended late penalty that is described in the "[Final Project Policies](#)" section does not start applying until after the final date and time listed.

Bonus	Deadline Option
+10	SAT 2021-05-08 (MAY 08) @ 03:30 PM EST
+5	SUN 2021-05-09 (MAY 09) @ 03:30 PM EST
+0	MON 2021-05-10 (MAY 10) @ 03:30 PM EST

Academic Honesty

You agree to the Academic Honesty policy as outlined in the course syllabus and the University honesty website. Furthermore, you must adhere to the copyright notice and licensing information at the bottom of this document.

Updates

If any updates to this project are needed after it is released, then they will be announced on Piazza. Updates related to correcting typos will probably not be announced unless they change the meaning of some requirement.

Project Description

Your goal is to implement, from scratch, a GUI application in Java 11 using JavaFX 11 that incorporates a preponderance of the topics introduced in this course in a way that demonstrates that you have met the learning outcomes related to those topics.

To get started, you must **pick one** the following app categories for this project:

1. External API Tool; or
2. Arcade Game.

A brief description of each category is provided below.

External API Tool

README.rst

that they want. Your app needs to do more than just download and display responses from the external APIs, it needs to combine the responses in some meaningful way.

- Services like the [Open Library Search API](#), [TheCatApi](#), [TheDogApi](#), [PokeApi](#), etc. provide **free access** to their RESTful JSON APIs -- a RESTful JSON API is one that you can access with an `InputStreamReader` and parse with a JSON library like Gson. For this project, you may only use RESTful JSON APIs and no other kinds of APIs.
- Some of these API services do require you to register with them to gain access to an "API key" -- an API key is usually just a special string that is unique to you that must be incorporated into how you request the JSON response. For example, suppose you have an API key for [TheDogApi](#) stored in `API_KEY`, then you might use the following URL when requesting the JSON for a list of breeds (see [/breeds](#)):

```
"https://api.thedogapi.com/v1/breeds?apikey=" + API_KEY
```

- If you choose this app category, then you should read the "[Working with RESTful JSON APIs](#)" appendix section before you write any code.

Arcade Game

Find a classic [arcade game](#) that interests you and implement your own version of it. The visuals and game mechanics must be easily recognizable and consistent with traditional implementations of the game you chose. You are required to utilize either keyboard event handlers or mouse event handlers that aren't related to one or more buttons.

- We have included a simple example of a JavaFX component for a game in [cs1302.game.DemoGame](#) -- it's actually used in the starter code that's provided for this project. It utilizes the abstract parent class called [cs1302.game.Game](#) that provides some neat features like a main game loop and support for continuously holding down a key.
- You are not required to utilize [cs1302.game.Game](#) ; however, feel free to adapt it into your abstract parent class in the `cs1302.omega` package. All of the things that you have learned about JavaFX still apply, but it's likely that your arcade game will have less buttons and more moving images.
- If you choose this app category, then you should read the "[Creating Games in JavaFX](#)" appendix section before you write any code.

Now that you have chosen an app category from the list above, you still have a lot of flexibility with regard to the functionality and visuals of your app. So long as your app actually functions and you meet the other requirements, you are free to make the app look and feel however you want (keep it appropriate).

Remember, part of software development is being given a goal but not necessarily being given instruction on all of the details needed to accomplish that goal. For example, even though working with things like keyboard events, mouse events, or API keys have not been explicitly covered in class, you may need to are going to need to look up how to do these things in order to complete this project.

Learning Outcomes

Here are some of the learning outcomes for this project:

- Plan, design, implement, test, debug, and deploy a complete object-oriented software solution in Linux/Unix environment (1302-LO1).

- Utilize inheritance and polymorphism in a software project (1302-LO3-LO4).
- Develop a GUI for a software project (1302-LO7).
- Implement exception-handling in a software project (1302-LO8).
- Understand and apply language basics using an OOP language (1302-LO11).

Functional Requirements

A functional requirement is *added* to your point total if satisfied. This assignment is worth 100 points.

Primary Functions (90 points)

Your app will have some general requirements related to its functionality that depend on the app category that you chose.

Here are the category-specific requirements:

External API Tool:	For an External API Tool , this means that your app integrates two or more external RESTful JSON APIs based on user input and combines the responses in some meaningful / interesting way. It also means that users can do whatever it is you say your app can do in the description that you provide for your app as part of Deadline 3.
Arcade Game:	For an Arcade Game , this means that the visuals and game mechanics are easily recognizable and consistent with traditional implementations of the game you chose, and that you utilized either keyboard event handlers or mouse event handlers that aren't related to one or more buttons. It also means that users can do whatever it is you say your app can do in the description that you provide for your app as part of Deadline 3.

Multiple Uses per Execution (10 points)

After the application is started, your application should allow a user to perform the primary function provided by the app an arbitrary number of times without requiring them to exit and rerun the application. By arbitrary, we mean that there is no limit to how many times the user may do this.

Here are the category-specific requirements:

External API	For an External API Tool , this usually means that the user is able to query the API(s) more than once without restarting the program.
---------------------	---

Tool:	
Arcade Game:	For an Arcade Game , this usually means that when a game ends, the program does not terminate; instead, the user is able to start another game should they desire to do so.

Non-Functional Requirements

A non-functional requirement is *subtracted* from your point total if not satisfied. In order to emphasize the importance of these requirements, non-compliance results in the full point amount being subtracted from your point total. That is, they are all or nothing.

User-Friendly Experience (10 points)

The windows of your application should not exceed a pixel dimension of 1280 (width) by 720 (height). Additionally, except for reasonable delays resulting from X forwarding, your application should not hang/freeze or crash during execution.

NOTE:	If a grader encounters lag, then they will try to run your application locally after first checking that it compiles on Odin.
--------------	---

Local Assets / Resources (10 points)

All assets (e.g., images), except for assets discovered using an external API, need to be pre-downloaded and placed either in the `resources` (not `src/main/resouces`) or a directory under `resources`. **This will help make your app faster.** Here are some examples that illustrate the relationship between the path for a resource and the `file: URL` that you need to use in your code:

Resource	URL
<code>resources/icon.png</code>	<code>"file:resources/icon.png"</code>
<code>resources/foo/img.png</code>	<code>"file:resources/foo/img.png"</code>

Code Style Guidelines (20 points)

You should be consistent with the style aspect of your code in order to promote readability. Every `.java` file that you include as part of your submission for this project must be in valid style as defined in the [CS1302 Code Style Guide](#). All of the individual code style guidelines listed in that document are part of this single non-functional requirement. Like the other non-functional requirements, this requirement is all or nothing.

NOTE:	The CS1302 Code Style Guide includes instructions on how to use the
--------------	---

check1302 program to check your code for compliance on Odin.
--

In-line Documentation (10 points)

Code blocks should be adequately documented using in-line comments. This is especially necessary when a block of code is not immediately understood by a reader (e.g., yourself or the grader).

Attribution (10 points)

Proper attribution should be given for **all assets** (e.g., art, sound, music, etc.) that you include in your project, especially assets that you did not personally author. All such attributions needs to be placed in the `meta/ATTRIBUTION.md` file.

For each asset that you authored, please provide the following information:

- * Asset Name
 - ``resources/path/to/file``
 - Your Name. Year.

For each asset that you did not personally author, please provide the following information:

- * Asset Name
 - ``resources/path/to/file``
 - Author. Year.
 - URL
 - License

NOTE:	Don't forget to stage and commit your <code>meta/ATTRIBUTION.md</code> file after you update it!
--------------	--

Final Project Policies

Final Project == Final Exam

Per university policy, each student must be provided the opportunity to stand for a final examination as part of the completion of a full instructional term, and instructors have the authority to design and administer the final examination for a course in whatever manner is appropriate. In CSCI 1302 this semester, **the final project that described by this document will be treated as the final examination** since the grade and feedback that a student receives for this assignment is a summative evaluation of the entire term's work.

Final Submission Deadline

Please take care to note the date/time for final submission deadline, **Deadline 3**. In particular, the deadline time is 03:30 PM, which is earlier in the day compared to previous projects.

Amended Late Work Policy

For both logistical and policy-related reasons, the usual late work policy will not apply for this project, and **no late submissions will be accepted after 11:59:59 PM on MON 2021-05-10 (May 10)**. You can still submit late for partial credit, but late submissions will only be accepted between **03:30:01 PM -- 11:59:59 PM on MON 2021-05-10 (May 10)**; submissions received during that time frame will incur the standard penalty for one day late. Final submissions received after the acceptance window will not be graded.

Non-Discrimination and Anti-Harassment Policy

Since this project affords you more flexibility with respect to the content of your app, you are reminded that, as a UGA student, you must conduct yourself in accordance with the [Non-Discrimination and Anti-Harassment Policy](#).

Private GitHub-hosted Git Repository

Each student is required to setup a private GitHub-hosted Git repository for their project with the CSCI 1302 course instructors for this semester added as collaborators. **Instructions are provided later in this document.**

Working on a Local Machine

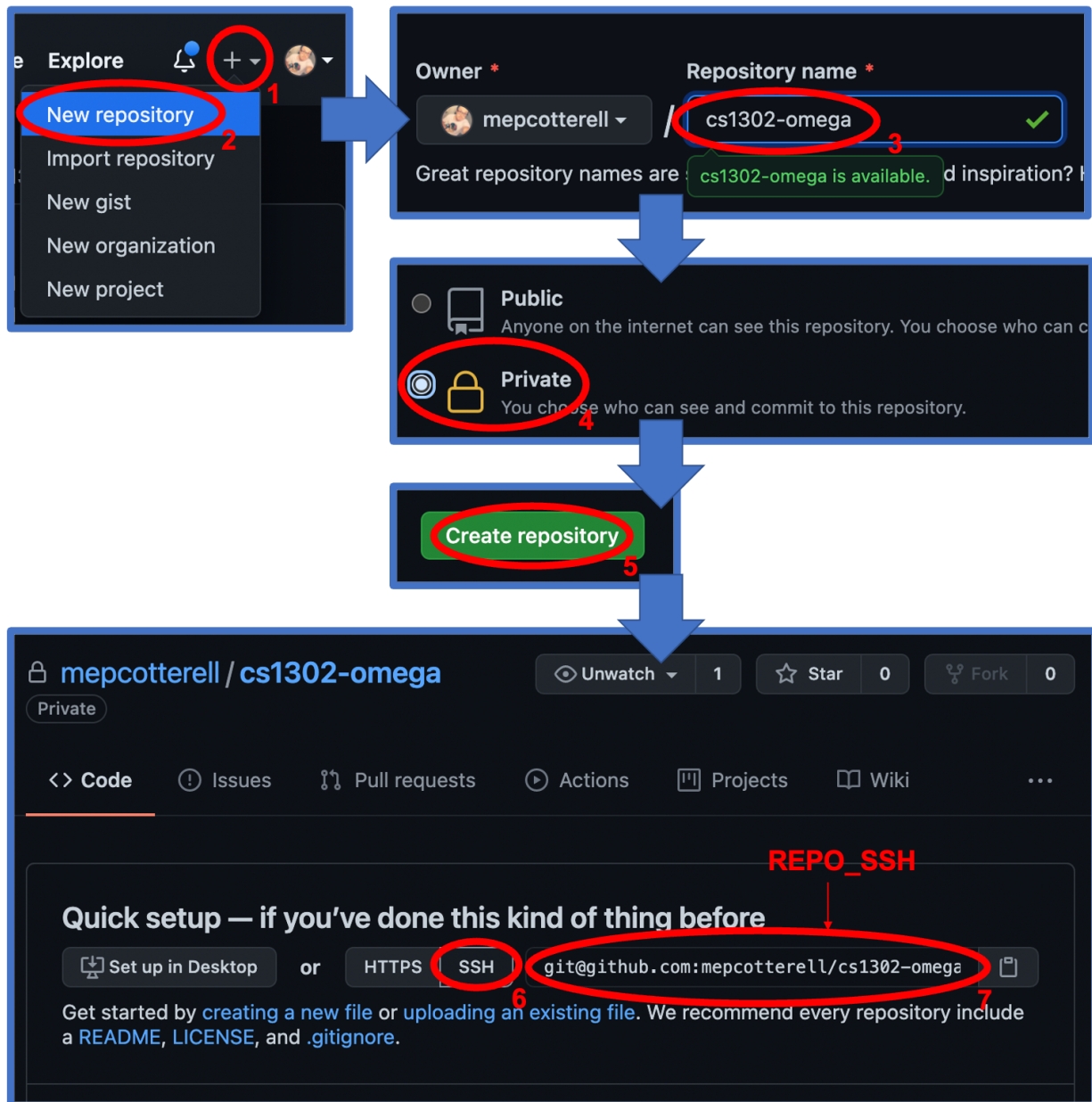
If you decide to work on part or all of the project on your local machine, then it's your responsibility to ensure that your environment is compatible with the versions of software on Odin. No technical assistance will be provided by the instructional staff to accommodate this beyond the information provided in this policy statement. Remember, **your code still needs to compile and run on Odin** per the "Development Environment" absolute requirement. That is, if your submission does not compile on Odin, then that will result in an immediate zero for the assignment. A list of the relevant software versions currently in use on Odin (at the time of this writing) is provided below for convenience.

- ***Apache Maven 3.8.1***
<https://maven.apache.org/>
- ***Java 11.0.10 (vendor: Oracle Corporation; not OpenJDK)***
<https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>
- ***OpenJFX 11.0.2 (note: should get handled by Maven)***
<https://gluonhq.com/products/javafx/>

How to Download the Project

Downloading the starter code for this project requires more steps compared to earlier projects. These instructions assume that you have completed the steps in "Setting up SSH Keys" to setup your public and private key pair on Odin and GitHub.

1. If you have not done so already, you should create a free GitHub-hosted private Git repository for your project under your GitHub account called `cs1302-omega` and note its SSH URL. Here is an example:



Remember to note the SSH URL!

NOTE: In the remaining instructions, `REPO_SSH` refers to the SSH URL for the private repository you created on GitHub.

2. Add the course instructors as collaborators to your private GitHub repository using the instructions provided in the [Setting up your own GitHub Account](#) reading. Here are the usernames for the course instructors.

- [mepcatterell](#) -- Dr. Cotterell
- [bjb211](#) -- Dr. Barnes

Be sure to add **both** instructors.

NOTE:	There may be a multi-day delay before one or both of the instructors accepts your collaboration invite. Don't panic; the instructors will accept the invites as soon as they can.
--------------	---

3. Clone your empty private repository to your Odin account.

```
$ git clone REPO_SSH cs1302-omega
```

You should now have a directory called `cs1302-omega` in your present working directory.

NOTE:	If you get an authentication error, then that means that you did not setup your public and private key pair on Odin and GitHub prior to following these instructions. Instructions for this are provided in the " Setting up SSH Keys " reading.
--------------	--

4. Setup a remote link the repository containing the starter code. A sequence of commands is provided below. You should make every effort to understand what each command is doing *before* you execute the command:

```
$ cd cs1302-omega
$ git branch -M main
$ git remote add starter https://github.com/cs1302uga/cs1302-omega.git
$ git pull starter main
```

If you followed these instructions correctly, then your present working directory (you should still be inside `cs1302-omega`) now contains the starter code and a `.git` directory.

5. You should think of the `cs1302-omega` directory on Odin as your local copy of the project. As you add, stage, commit, branch, etc., those changes are only local to that copy of the project -- they do not automatically appear on the GitHub page for your repository. To send changes to GitHub, follow these steps:

- i. Use `git status` to ensure that you are on the `main` branch and fully committed. If you're not, then take the necessary steps to make sure that you are.
- ii. Try to pull changes from GitHub (this may require you to manually merge in the case of a conflict; that's okay -- merge, commit, then continue):

```
$ git pull origin main
```

- iii. Push changes to GitHub:

```
$ git push origin main
```

In your browser, revisit your GitHub-hosted private Git repository. Instead of an empty repository, you should now see the starter code.

You can follow the steps above any time you want to send your local changes to GitHub.

NOTE:	If you have trouble getting any of this to work, then try asking on Piazza or see someone during office hours.
--------------	--

6. Clean, compile, and run the starter code using the provided `compile.sh` script:

```
$ ./compile.sh
```

Here is the expected output, which also shows the related Maven commands, should you wish to type them out manually:

```
+ mvn -q -e clean
+ mvn -q -e compile
+ mvn -q -e exec:java -Dprism.order=sw
```

By default, the project is setup to automatically run the `cs1302.omega.OmegaDriver` class. If you wish to run another driver class, then you can provide the `-Dexec.mainClass` option after the script name:

```
$ ./compile.sh -Dexec.mainClass=cs1302.api.PropertiesExample
```

Any other command-line options that you add after the script name will be added to the end of the `mvn` command that executes `exec:java`.

Submission Instructions

Deadline Option 1: SAT 2021-05-08 (MAY 08) @ 03:30 PM EST

NOTE:	Same instructions as the MON 2021-05-10 (MAY 10) @ 03:30 PM EST deadline.
--------------	---

Deadline Option 2: SUN 2021-05-09 (MAY 09) @ 03:30 PM EST

NOTE:	Same instructions as the MON 2021-05-10 (MAY 10) @ 03:30 PM EST deadline.
--------------	---

Deadline Option 3: MON 2021-05-10 (MAY 10) @ 03:30 PM EST

For this deadline, you're required to **include the your final project code** and **update to your deadline file**: `meta/DEADLINE.md`.

1. Update your project's `meta/DEADLINE.md`. Specific instructions for what to include in the update are contained in the file itself.
2. Merge all of your work in progress into to the `main` branch, then tag your `main` branch for this deadline as described below.
 - i. Ensure that whatever branch you are on is **fully committed** (i.e., `git status` says there is nothing to commit).
 - ii. Checkout the `main` branch.

```
$ git checkout main
```

iii. If needed, merge changes into `main` from the branch you were working on following the instructions provided in the "[Git Feature Branch Workflow](#)" appendix section.

iv. Tag your `main` branch by executing the commands below:

```
$ git tag -am "deadline" deadline
$ git push origin --all
$ git push origin --tags
```

NOTE:	Take special care to ensure that your fully-committed <code>main</code> branch reflects the project you wish to submit. Compare your log to the log on GitHub. If your GitHub repository does not have the most recent version of your project, then you may need to do a <code>git push origin main</code> while on your <code>main</code> branch.
NOTE:	If you need to make more commits and retag, then use an <code>a</code> , <code>b</code> , <code>c</code> , ... suffix in the tag names (e.g., <code>deadline-a</code> , <code>deadline-b</code> , etc.).
NOTE:	Evidence of branching and merging is encouraged for this deadline. When inspecting your Git log, the graders would like to see that you made proper use of <code>branch</code> , <code>checkout</code> , and <code>merge</code> to work on portions of your project prior to including those changes in your <code>main</code> branch. More detailed instructions are provided in the " Git Feature Branch Workflow " appendix section.

4. **CRITICAL:** For this deadline, you also need to submit on Odin! Use the `submit` command to submit your project on Odin for this deadline:

i. Check for style guide violations:

```
$ find cs1302-omega/src/main/java -type f -name "*.java" | xargs chec
```

NOTE:	If there are style guide violations, then checkout a new branch, fix your code, commit, test your program, potentially fix some more, commit, then checkout <code>main</code> and merge in the beautiful code from the branch you were just in. You should also retag and push your <code>main</code> branch as described elsewhere. Once you have no style guide violations, you can proceed to the next step.
--------------	---

ii. Perform your final submission:

```
$ submit cs1302-omega csci-1302
```

NOTE:	If you have any problems submitting your project, then please contact the CSCI 1302 Support Team by sending a private post to "Instructors" via the course Piazza as soon as possible.
--------------	--

Appendix

JavaFX

- [JavaFX 11 API Documentation](#)
- [JavaFX 11 Bookmarks](#)
- [CSCI 1302 JavaFX Tutorial](#)

Git

- [Git Feature Branch Workflow](#)

RESTful JSON APIs

- [Working with RESTful JSON APIs](#)

Games

- [Creating Games in JavaFX](#)

FAQ

Below are some frequently asked questions related to this project. You may also find the [cs1302-gallery FAQ](#) a useful resource as well.

1. May I use an API not mentioned in the project description?

RESTful JSON API

If you're asking about a RESTful JSON API that's not mentioned in the project description, then probably yes! Here are the requirements:

- the API and your use of the API does must not violate the UGA [Non-Discrimination and Anti-Harassment Policy](#); and

- the API must provide a JSON response based on a request to a URL that is pragmatically generated by your program.

If you're not sure about an API, then ask on Piazza.

Java API

If you're asking about a third-party Java API that is not included with Java 11, JavaFX 11, Gson 2.8.6, or the starter code, then the answer is no.

2. How do I add sound?

While JavaFX does support audio playback of various formats, this feature is not currently available over X11 forwarding from Odin. We're sorry to say this, but **you should not attempt to add audio to your application** for this project.

3. How can I generate my Javadoc using Maven?

For this project, a `site.sh` script is provided that will deploy a Maven site, including Javadoc, to your `~/public_html/cs1302-omega` directory when executed on Odin. The script will display the URL of the deployed site near the end of its execution. Here is the command:

```
$ ./site.sh
```

Feature Preparation Timestamps:

License CC BY-NC-ND 4.0

Copyright © Michael E. Cotterell, Bradley J. Barnes, and the University of Georgia. This work is licensed under a [CC BY-NC-ND 4.0](#) license to students and the public. The content and opinions expressed on this Web page do not necessarily reflect the views of nor are they endorsed by the University of Georgia or the University System of Georgia.

Contributors 2



mepcotterell Michael Cotterell



bjb211 Brad Barnes

Languages

● **Java** 97.5% ● **Shell** 2.5%