Meme Magic: Project in Sprints

John R Hott
University of Virginia
Charlottesville, Virginia, USA
jrhott@virginia.edu

Derrick Stone
University of Virginia
Charlottesville, Virginia, USA
dstone@virginia.edu

Nada Basit
University of Virginia
Charlottesville, Virginia, USA
basit@virginia.edu

Daniel Graham
University of Virginia
Charlottesville, Virginia, USA
dgg6b@virginia.edu

Course Data Structures
Programming Language Java
Knowledge Unit Software Development Methods
CS Topics Object-Oriented Programming, Inheritance, Software Development and Design

SYNOPSIS

Meme Magic is a series of six assignments intended to provide progressive exposure to programming in Java using a popular and recent concept: Memes. Memes utilize an image conveying a concept or feeling with a caption provided by the Meme author. The series of assignments, designed as sprints in the context of a larger project, begin with the design and scaffolding of Java classes needed to write a program to produce text-based Memes and end with a fully-functional graphical user interface. For a detailed list of learning goals, please see the Learning Goals section. In the first sprint, students depict the overall project structure of a text-based meme application using Unified Markup Language (UML) and write method stubs in Java. In each of the next two sprints, students implement half of the specified functionality and integrate those components to a fully working application. Students are asked to add Comparators to sort memes to their application in sprint 4 and to unit test all of their code using JUnit in sprint 5. In the final sprint, students extend the functionality once more to a graphical user interface to experience event-driven programming. Once the full sequence is completed, students will be able to generate and save graphical memes. Steps and learning concepts include designing the project structure using UML diagrams, implementing that design, unit testing with JUnit[1], and event driven programming using Swing.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Student assessment**.

KEYWORDS

Software Design, Object-Oriented Programming, Unit Testing, Data Structures, Event-driven programming, Graphical User Interface

ACM Reference Format:
John R Hott, Derrick Stone, Nada Basit, and Daniel Graham. July 2022. Meme Magic: Project in Sprints. In *ACM EngageCSEdu*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3519933

1 ENGAGEMENT HIGHLIGHTS

Meme Magic engages the learner with content relevant to a modern student by leveraging a recent social media phenomenon: the meme. Students have the opportunity to create memes and are encouraged to share them at the culmination of the series of assignments. This series utilizes a modern technique of software development: breaking a large, complex problem into smaller deliverable sections intended to simulate an Agile approach. Each section of the project, or sprint, has key learning goals emphasized in the work assigned, affording the learner an opportunity to understand the object-oriented approach one layer at a time. This approach also helps the learner to complete a more advanced project that might otherwise be accomplished by students in a CS2-level course.

Given the prevalence of memes across most social media platforms, and their familiarity of and use by students from different backgrounds, supplementary assignments to this series can leverage existing interest to incorporate class discussions, presentations, or activities on ethical or societal

[1]https://junit.org/junit4/

issues surrounding the creation and dissemination of memes. Relevant topics may include:

- Copyright considerations: including discussions on the implications of using images that are not your own in creating a meme, providing appropriate attribution, etc;
- Women (and other prominent figures) in computer science: encouraging students to research prominent computer scientists and use their stories and images for the memes they create, as well as asking students to give a short presentation about these prominent figures of computer science using their memes;
- Inclusivity: discussing the use of inclusive or contentious images and language in memes, with an opportunity to bring in a guest speaker from the liberal arts or the school of education;
- Art and design: coordinating with instructors or practitioners in the Arts to facilitate discussions or create activities about selecting an image, color schemes, choice of font, composition, graphic design, etc; and
- "Netiquette": including activities or discussions on the impact certain images have on different individuals and groups in society.

2 RECOMMENDATIONS

This series of assignments has been developed for students with some exposure to programming, though students do not necessarily need prior exposure to the Java programming language. It is therefore recommended that this project follows some introductory material on Java. It may also be helpful to provide instructions on (i) designing classes using Unified Markup Language (UML), and (ii) good coding practices such as writing thorough comments, including the use of JavaDoc style commenting.

Given that each sprint in the series builds on prior sprints (to work on assignment x, the students must have completed assignment $x - 1$), instructors should help students avoid falling behind. One strategy to avoid having an individual student fall behind is to have them work with a group of 3-4 other students. An approach that preserves individual work is to have the students discuss their approach but not share code until a sprint has been completed. This ensures everyone has a fully working version before the start of the next sprint. Another approach might be to allow students to fully collaborate, but it may be hard to ascertain if all students are internalizing the learning objectives. If each sprint is assessed independently from prior submissions, this provides a mechanism for students to be prepared for the next sprint while continuing to assess individual student contributions.

3 LEARNING OBJECTIVES

Throughout the Meme Magic project in sprints, the following learning objectives are addressed:

Sprint 1. Creating UML (Unified Modeling Language) diagrams, implementing class design consisting of method stubs, utilizing style guidelines for increased readability and consistency, and including appropriate comments for clarity and readability;

Sprint 2. Implementing default and overloaded constructors, implementing getter and setter (accessor and mutator) methods, utilizing an array to store reference objects, experiencing inheritance from `Object`: overriding `toString()` and `equals()` methods, and incorporating main-method testing to check correctness (i.e., behavior matches specifications);

Sprint 3. Utilizing `ArrayList` to store and organize objects, integrating newly implemented classes into an existing code base, and incorporating main-method testing to check correctness (i.e., behavior matches specifications);

Sprint 4. Implementing the `Comparable` interface to produce a natural ordering of objects, implementing the `Comparator` interface to produce additional orderings of objects, sorting lists of objects utilizing the `Comparable` and `Comparator` interfaces, using `TreeSet` to store unique objects defined by `hashCode()` and `equals()`, and accessing and referencing Java API for objects;

Sprint 5. Writing unit tests for Java using JUnit and calculating code coverage of unit testing; and

Sprint 6. Implementing graphical user interfaces (GUIs) using event-driven programming, organizing GUI content using Java Swing containers and components, and handling exceptions with try/catch blocks.

4 PITFALLS

The major pitfall to avoid is the case in which a student does not complete any one assignment. Subsequent assignments will be harder or impossible without the preceding work, thus putting the student at a disadvantage. Along with encouraging students to collaborate and share code after each sprint has completed, other possible countermeasures to this approach are:

1. Providing a working solution at the end of each sprint,
2. Having Instructor or Teaching Assistant-led reviews of each sprint's requirements, and
3. Utilizing pair programming or assigning the sprints to small groups of students.

One possible pitfall with assigning teams to work on the sprints is the possibility of some students not fully contributing. To mitigate this, student groups may be asked to submit a short breakdown of the contributions of each member for each assignment. Alternatively, peer evaluations can be conducted via an online form at the end of each sprint or after the completion of the series. The instructor should make it clear at the start of the course that peer evaluations will factor into sprint grades.

While pair programming and group environments provide many benefits, further issues may arise if collaboration policies are unclear. In an employment situation, the most important result is the outcome of the assignment. In contrast, the most important result in a learning environment is individual student comprehension and retention, which is not easily measurable without some degree of identifiable individual work.

5 MARKING GUIDELINES

We encourage a mixture of automated grading and manual inspection. Since the first sprint comprises the design stage, creating UML diagrams for classes in the project as well as writing class and method stubs, we suggest manually grading the diagrams for completion and correctness. We additionally provide unit tests to check that all classes and methods stubs have been correctly implemented. This process facilitates the implementation of future sprints, since the next two sprints encompass completing the designed functionality.

Depending on course size, for the next three implementation-heavy sprints, we encourage an 80-20 score breakdown in which 80% is provided by an auto-grader and 20% is manual inspection. In these sprints, we limited the number of submissions to the automated platform to encourage students to plan their implementations and test their code using main method testing before submission. We inspected their tests manually, assigning 15% of the grade based on testing. Lastly, 5% was reserved for code readability.

In sprint 5, students are asked to write JUnit tests for their entire code base. We encourage scoring this based on the amount of code coverage their tests employ. We provide an example auto-grader that assigns a score proportional to code coverage, requiring at least 50% coverage. An additional 10% is reserved for manual inspection to verify that students are still following good coding practices.

For the final sprint, since GUIs are difficult to automatically grade, we encourage assigning the entire grade manually. To facilitate this process, we asked all students to create a video describing their implementation and showing their code, launching their program multiple times, and walking through using their GUI to create a meme.

6 EXTENSIONS

We envision a number of extensions to Meme Magic that can be applied based on differing course learning objectives or instructor preferences. By using Java as the underlying programming language, the final product may easily be adapted to other platforms. Here are some examples:

(1) Replace sprint 6 to produce an Android app instead of a desktop-based application by replacing `MemeMagic.java` scaffolding with an Android-based implementation;
(2) Update sprint 6 to utilize the JavaFX framework instead of Java Swing;
(3) Encourage creativity in sprint 6 through asking students to modify the GUI by redesigning the overall GUI layout or including colors, additional elements, or text in different locations in the produced meme image;
(4) Increase difficulty of the last assignment by tasking students with creating their own GUI based on functional requirements of generating graphical memes without the use of scaffolding code;
(5) Expand the capabilities of the GUI to organize and view previously-created memes by incorporating the functionality of the `Feed` and `User` classes;
(6) Ask students to present their final GUI in a class expo to give them practice presenting in front of others; or
(7) Facilitate community among students by encouraging them to publish their memes on a course discussion site.

7 MATERIALS

We provide instructions, code scaffolding, JavaDoc documentation, and relevant examples to supplement the assignment. The following student-focused content is provided in the `Student` directory:

- `instructions` - a directory containing the instructions for each sprint, including a combined version in both PDF and Word document formats and an expanded rubric for each of the sprints based on the auto-graders available below;
- `resources` - a collection of resources that includes:
  – `style-guide.pdf` - a coding style guide,
  – `CS2Style.xml` - an Eclipse configuration file to match the included style guide,
  – `Method Stubs.html` - a short overview of writing method stubs,
  – `UML in Text Files.html` - a short overview of writing UML as text files, and
  – `pineapple_pizza` - an Eclipse project that includes an example GUI application using Swing; and
- `scaffolding` - code scaffolding for `sprint6` along with example main method testing for `sprint4`.

Additionally, example solutions and auto-grading tools are available for instructors on request. That material includes:

- `Syllabus.pdf` - the course syllabus utilizing Meme Magic;
- `Course Schedule.pdf` - the schedule of topics that include Meme Magic;
- `javadoc` - a directory containing the HTML documentation for all classes and methods for `sprint2`, `sprint3`, and `sprint4`;
- `solutions` - example solutions for each sprint; and
- `autograders` - auto-grader code for each of the sprints as JUnit tests formatted for use with Gradescope[2], but which may be modified for other JUnit-based graders.

[2]https://gradescope.com