### Fill out this sheet and submit with your project as Name\_\_\_Brandon Nguyen\_\_ a Word document or PDF file. Total Points Expected\_\_\_\_\_\_\_\_\_/ 80

Grading Description & Rubric Program 8 Project Description—10 points

**High level description:**

FallingSand is a paint program that models different materials within a physical environment. It paints materials to a canvas, where each have different properties that interact with each other and the surrounding world.

**Elements added:**

More details can be found within the README file inside the project.

* Reset functionality
* Materials abstracted
* Objects use abstract Material class
* Certain objects implement Movable and/or constants class
* Ice, Gas, Cloud, Dirty Water, Dirty Sand materials added

**Code attempts that were abandoned and why:**

* **Using only one form of abstraction**

This was abandoned because not all materials are equal. Some moved and shared common functionality. Others stayed still and did nothing. However, they still share certain attributes (name, color) which means that multiple layers of abstraction were needed to avoid duplicate code but retain functionality.

* **Hardcoding values**

After a lot of trial and error between different coding decisions (Do I add a helper function here? Do I fundamentally change the display?) any point of the code that could be confusing was changed. A big part of this were actually using *final* values and not spreading them throughout different classes.

* **Using a Deque**

I was going to use a Deque for the names array in SandDisplay but it doesn’t simply support traversing through every value within it unless I used a nested loop and an iterator. ArrayLists were much easier to read in code.

* **Micro-optimization**

I spent way too much time looking into optimizing tiny things (e.g. using Arrays.copy vs a for-loop) that does not drastically affect the performance. Occasionally, this hampered code readability which was favored over miniature runtime improvements.

**Data structures used (ArrayList, LinkedList, HashTable, Queue, Stack, array, etc.):**

Array, ArrayList, List, HashTable(default implementation)

**Classes added after water, sand, and metal:**

Cloud, DirtySand, DirtyWater, Gas, Ice

**Approximate time spent on project:**

20 hrs.

**Anything else you want us to know as we evaluate your work:**

A lot of time was spent on ensuring code readability and good documentation. It is not perfect but I think that it is a lot better than uncommented code and random classes!

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Proficient** | **Competent** | **Novice** | **Unsatisfactory** |
| **Project Description**  **10 pts.** | Filled in completely. Accurately describes work done. | Filled in. Mostly describes work done. | Filled in. Sparse or inaccurate description of work done. | Information omitted and/or poorly description of work done. |
| **Documentation & Readability & Organization 5 pts.** | The documentation is well written and clearly explains code use. Code is exceptionally well organized and conforms to best practices. The main method has limited functionality. | The documentation is somewhat useful in understanding the code. The code is fairly easy to read and conforms to most best practices. | The documentation is minimal and/or the code is readable only to grader who has the assignment description. | The documentation is limited or non-existent and/or no modularization. |
| **Basic Features**  **Metal, Sand, Water**  **25 pts** | Code is complete, runs, and has no more than one minor flaw. | Code is complete, compiles. Has minor flaws. | Code has major flaw(s) but runs. | DOES NOT COMPILE  ZERO POINTS. NO PARTIAL CREDIT FOR PROJECT POSSIBLE. | |
| **Additional Features: three more elements**  **20 pts** | Code is complete, runs, and has no more than one minor flaw. Elements are distinct with unique actions. | Code is complete, compiles. Has minor flaws. Elements are distinct with unique actions. Omitted one element. | Code has major flaw(s) but runs. Elements are distinct with unique actions.  Omitted one or two elements. | DOES NOT COMPILE  ZERO POINTS. NO PARTIAL CREDIT FOR PROJECT POSSIBLE. | |
| **Advanced Features**  **Abstraction, Polymorphism**  **20 pts** | Code is complete, runs, and has no more than one minor flaw. Good organization. | Code is complete, compiles. Has minor flaws. Fairly good organization | Code has major flaw(s) but runs. Okay organization. | DOES NOT COMPILE  ZERO POINTS. NO PARTIAL CREDIT FOR PROJECT POSSIBLE. | |
| **Extra Credit**  **Complexity, creativity, etc.**  **10 pts max.** | Instructor’s discretion | Instructor’s discretion | Instructor’s discretion | Instructor’s discretion | |

### Grading Rubric Feature Definitions

Basic Completed Part One: Added elements sand and water. Behavior of elements is correct.

Moderate Basic plus has added 3 or more elements. No more than 2 of the elements has complex behavior.

Advanced Moderate plus has added complex behavior, modified code to have base element class and subclasses, added objects, modified design of canvas, or other modifications deemed by the instructor to be advanced.