

rmiller

Create a working model of a small bank

| DiCe Rubric | | | |
|--|--|--|----------------------------|
| Documentation (15%) | Design Plan (3%) | Easy to understand, Logical sequence, Clearly defined sections. Provides general overview, etc. | 3% |
| | Testing Plan (3%) | Provides detailed steps to repeat test. Appropriate coverage of code. | 3% |
| | Project Writeup (3%) | Does the writeup document challenges and surprises and lessons learned encountered during the project? | 3% |
| | Grammar/ Spelling (3%) | Was the documentation free of grammatical and spelling errors and formatting inconsistencies? | 3% |
| | Code Formatting (3%) | Does the formatting of the code adhere to the common style as run through pycodestyle? | 3% |
| | | Are variables/functions named appropriately and enable code readability? | |
| | | Are doc strings (PEP257) and comments utilized? | |
| Are classes/modules/files named appropriately and enable code readability? | | | |
| Implementation (35%) | Version Control (5%) | Is a branch created to address each requirement or feature? | 5% |
| | | Is the history free of generated/artifact files? | |
| | | Are commit messages informative? | |
| | | Is the master branch free of direct work? | |
| | Architecture (25%) | Are classes and inheritance used effectively for the problem at hand? | 8% |
| | | Was the code designed and constructed in a modular fashion? | 4% |
| | | | |
| | | | |
| | | Were generally sound decisions made with regard to architecture? | 9% |
| | | Is the code SOLID & DRY? | 4% |
| Testing (5%) | Were comprehensive and robust test cases constructed to include but not limited to the test cases provided in this document? Are all tests repeatable? | 5% | |
| Execution(35%) | Safety (10%) | Does invalid input cause the program to crash? | 5% |
| | | Does invalid input cause the program to act inappropriately? | 5% |
| | Requirements (15%) | | |
| | | | Were all requirements met? |
| | | If multiple drones deployed to same map, they appropriately divide work | 3% |

| | |
|----|------------------------|
| 76 | Base Grade w/o Group |
| 85 | Final Grade with Group |

| Deduction | Reason |
|-----------|---|
| | Nice inclusion of screen shots and descriptions of tests for GUIs |
| 1 | programmically instead of programmatically |
| 3 | Doc string and type hint for Dashboard state it take a tkinter.Toplevel - when in reality it is taking a "root" Tk object passed by my code use of tkinter.Misc as a datatype in line 19 of map.py is not documented anywhere - even in your docstring where you claim it is _type_ Type hinting has to be consistently all or none for readability - line 60 of overlord.py does not specify return type as most other methods do |
| 5 | your definition of __eq__ in tile.py uses () to allow multiple lines -which means you should have used a simple if else stamen - AND the method should return True or False - not NotImplemented as the method is indeed actually implemented by you lines 177 to 187 never document or comment anything about the use of @overload in your code and why you are defining the same method 3 times with two of them that do nothing Absolutely no explanation as to why __repr__ was used instead of __str__ to return a string representation of the object - as that is what the __str__ is designed for. Not using the __repr__ in a nonstandard way without explaining why get method on line 187 of map.py does not use type hinting while rest of surrounding code does Your use of a datatype called "Context" in your action type hinting locks it into something it is not. When I call your action it is impossible that I pass you a datatype that was never defined in the API that was given to you. All you know is it is an object that contains 6 attributes |
| 1 | main.py at root of repo - is testing code and as such belongs in "test" even if it is not tdd. Also file is not pycodestyle compliant - but no points off this time for that part |
| 9 | Your Overlord has no health - though it is defined in Zerg's init - your code never calls it and has such has no _health (3) only 10 of available 5601 minerals were collected (.2%) steps() counted was only 54% accurate (239/440) (7) 9 unrecovered exceptions out of 9 runs |

| | | | |
|--|------------------------------|--|----|
| | Performance (10%) | Appropriate use of summary value in overall strategies | 3% |
| | | Drones take appropriate risks based on health | 3% |
| | | Other Performance Issues | 1% |
| | | | |

Documentation 15
Implementation 35
Execution 35
Total Points Available 85
Total Deductions

| | |
|----|--|
| 3 | 4 times drones deployed to summary of zero 15 times drone was deployed to map without highest summary first |
| 1 | Running with 30K ticks crashes at about 16k |
| 11 | |
| 29 | |
| 22 | |
| 62 | |
| 23 | |

| | | | |
|-------------------------------------|-----------------------|---|----|
| Suggested Features (15%) | Area | Feature | |
| | Documentation | Write a man(1) page for your program. | 2% |
| | Documentation | Provide a UML Diagram for the classes in the project | 2% |
| | Documentation | Submit your writeup in Tex or LaTeX in addition to the already required pdf format | 2% |
| | Implementation | Use TDD to write as many tests as possible that can be run automatically. Put tests in a separate subdirectory of the project named test . | 2% |
| | Execution | Employ the A* pathing algorithm | 8% |
| | Execution | Update the Dashboard with stats of each instantiated Drone and update info with each tick | 4% |
| | Execution | Enhance the map windows in the GUI to use more meaningful characters, colors, or images rather than the "#~*Z" characters to display the map | 4% |
| | Execution | Use the logging module from the Python standard library to log information about the drones to a file located within the mining package .Some examples of information to log may be information about the Drone when it is initialized or each time a drone dies. | 4% |
| | | | |

| | |
|----|---|
| 2 | |
| 0 | Not Implemented |
| 2 | |
| | 9 tests - 4 Errors |
| 1 | |
| 0 | Dijkstra's implemented - but not A* |
| | Not updated with each tick |
| 1 | |
| 4 | |
| | |
| 4 | |
| 14 | Points out of max 15 available for Features |
| 9 | |

Group Points 2

| | | |
|------------------------------------|--|-------------|
| Functional Requirements | Requirement | Area |
| | The project must run on the class Virtual Machine. | Execution |
| | Classes named Overlord, Drone, and Dashboard created as defined in project description | Execution |
| | All types of Zerg possess health and an action | Execution |
| | Overlord must use proper three argument constructor. | Execution |
| | All Drones instantiated during instantiation of Overlord | Execution |

| | | |
|---------------------|---|----------------|
| Requirements | Requirement | Area |
| | Design plan, test procedure, and writeup documents must be submitted with the project. | Documentation |
| | Test Cases used must be submitted with the project. | Implementation |
| | All source code and documentation must be submitted to the class version control system by 23:59 on the due date specified. | Implementation |
| | Basic Drone has all three - health capacity and moves | Implementation |
| | Total of refined_minerals used by instantiated Drones does not exceed those available to Overlord | Implementation |
| | Dashboard displays title of each top level window and id of map it displays | Implementation |

| | | |
|--|--|----------------|
| | Constraint | Area |
| | Make use of appropriate variable names. | Documentation |
| | All documentation must be in PDF format. | Documentation |
| | PEP-8 code style is required. | Documentation |
| | Docstrings must be used appropriately. | Documentation |
| | No third-party files/libraries may be used unless signed off by the Program Managers or Instructors. | Implementation |
| | Each logical portion or feature must be built in its own branch. | Implementation |
| | Merge (do not fast-forward) all commits to branch master and tag releases appropriately. | Implementation |

| | | |
|--------------------|---|----------------|
| Constraints | The default branch to clone should be master. | Implementation |
| | Code must be DRY when possible. | Implementation |
| | The project must be written in Python 3. | Implementation |
| | The project should be stored in your assigned VCS account, under the repository named <i>mining</i> . | Implementation |
| | The top level package for your project should be named <i>mining</i> . | Implementation |
| | | |
| | All types of Zerg possess at least health of 1 and an action that takes a map context as a parameter | Implementation |
| | Drones cannot be repaired at any point in their lifetime | Implementation |
| | Overlord exposes proper add_map and action methods | Implementation |
| | Timeouts do not occur due to 1 second Overlord constraint and 1 millisecond Drone constraint | Implementation |
| | Overlord exposes public <i>drones</i> instance variable that is a dictionary. | Implementation |
| | Overlords action returns one of three required behaviors | Implementation |