

Midterm Exam Part 2 (level-1) – A Hand-on Exam to implement different requirements
50 points

Due (via your git repo) no later than 4:40 p.m., Wednesday, 31st March 2021.

Part A (25 points):

The key focus is to test your understanding on Computational Constructs and List Data Structure.



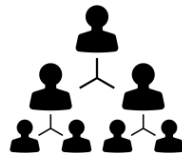
1. For simplicity, a starter code using the file named `dslist1.py` is provided in the exam repository. The starter code is incomplete. You are required to complete the implementation.
2. The starter code automatically fills out a list with a random set of values. The size of the list is random as well.
3. You are required to identify the total number of odd and even numbers in the list. The implementation for this part should be done by completing the code in the `evenNos` and `oddNos` methods.
4. Once implementation complete, executing the modified starter-code should correctly print the list, total count of odd nos, and even nos in the list.
5. An example output is displayed below for your reference:

```
amohan@amohanmacpro midterm-test % python3 dslist1.py
input list: [15, 12, 11, 89, 90, 91, 93]
count of odd nos: 5
count of even nos: 2
amohan@amohanmacpro midterm-test %
```

PS next page.

Part B (25 points):

The key focus is to test your understanding on Inheritance and Abstract Classes.



1. For simplicity, a starter code using the file named `worker.py` is provided in the exam repository. The starter code is incomplete. You are required to complete the implementation.
2. Create an abstract class called `worker` in the `worker.py` file.
3. Create an abstract method in the class `worker` called `duty`.
4. Create a non-abstract method in the class `worker` called `greet`.
5. Create four different workers namely: `doctor`, `attorney`, `carpenter`, `programmer`. These four workers should be implemented in their own individual classes inheriting the `worker` class. Implement all these classes in the `worker.py` file. Please note, I don't require you to create any additional `.py` file. All the implementation should be completed within the `worker.py` file.
6. Implement the `duty` method differently for different workers. For example:

A **doctor** should print:

```
I diagnose and treat medical conditions.
```


An **attorney** should print:

```
I advise and represent the legal rights of my clients.
```


Carpenter should print:

```
I construct, repair, and install building frameworks and structures.
```


Programmer should print:

```
I write code for computer programs and applications.
```
7. Implement the `greet` method with one parameter called `type` in the `worker` class. The `greet` method should then print a message based on the corresponding type. For example: **"I am a doctor"**
8. The above said `greet` method should be called from all the sub-classes. The method should be calling using the corresponding worker type. These calls should be implemented in the `duty` method. That is, before printing the duties for the individual worker, make a call to the `greet` method. The greeting message should be printed first. Next, the corresponding worker's duties should be printed.
9. Instantiate four objects for the four sub-classes in the `worker.py` file. Make a call to the `duty` method by using the four objects that are instantiated.

PS next page.

10. An example output is shown below for your reference:

```
amohan@amohanmacpro midterm-test % python3 worker.py
Hello, I am a Doctor:
    I diagnose and treat medical conditions.
Hello, I am an Attorney:
    I advise and represent the legal rights of my clients.
Hello, I am a Carpenter:
    I construct, repair, and install building frameworks and structures.
Hello, I am a Programmer:
    I write code for computer programs and applications.
amohan@amohanmacpro midterm-test %
```

To wrap up:

- Make edits to the honor-code.txt file. Here, read through the honor code statement and sign by replacing Student Name with your name. The honor-code is required to be signed for the work to be graded.
- Thanks for completing this exam. Share your experience about this exam by listing out your points and answering the questions in the reflections.txt file.

Submission Details

For this part of the exam, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. `dslist1.py` and `worker.py`.
2. A document to reflect your experience in this exam in a file named `reflections.txt`.
3. A document with the honor code pledge signed in a file named `honor-code.txt` document.
4. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is completed before the deadline. The honor code policy can be accessed through the course syllabus.