Activity 6.3 - Writing Classes I (Task with enum)

Overview

This activity will be the first part of a larger ToDo List Manager program. You will write a class to represent a single Task and a driver class with a main method to test your Task class. We will practice using an enum class.

Instructions

Getting Started

1. Create a "`Activity6.3`" folder in VS Code.
2. Create a new class named `Task` and a new class named `TaskMaster`. Add a main method to your `TaskMaster` class.

`Task` Requirements

Category Enum

Define a public enum called `Category` that can have values: PERSONAL, WORK, SCHOOL, FUN, NONE.

Instance Variables

Every task will have the following instance variables (aka. attributes).

- A *`description`*.
- A *`priority`* (this will be any integer value).
- Whether or not the task is *`complete`*.
- A *`category`* that has one of the values of the Category enum

Constructors

To give the user flexibility in how they create a task, we will provide the following *overloaded constructors*. Make sure to initialize ALL instance variables in both methods. Think about the default values you will use if a value is not passed as a parameter.

- `Task(String description)`
- `Task(String description, int priority)`

toString Method

Write a `toString` method that will return a String representing the state of the task in the following format. If the task is complete, prepend "`[X]`" before the description, else prepend "`[ ]`".

Incomplete task output format:

```
[ ] description, priority, category
```

Complete task output format:

```
[X] description, priority, category
```

Getters/Setters (Accessors/Mutators)

Provide getter and setter methods for all of the instance variables.

- `getDescription`, `setDescription`
- `getPriority`, `setPriority`
- `isComplete`, `setComplete`
- `getCategory`, `setCategory`

`TaskMaster` Requirements

In your `TaskMaster` "driver" class (the class with the `main()` method), you will test the constructors and methods of your `Task` class by creating and calling methods for two `Task` instances.

Task 1

- Use the constructor that takes in a description to create a new `Task` object, named `task1`. Use "Finish Activity 6.3" as the task description. (See sample output below).
- Print `task1` to verify that your `toString` method works as expected.

- Set `task1` to complete using the `setComplete` method and verify the value was set correctly using your `isComplete` method.
- Print `task1` again to verify that your `toString` method still works as expected.
- Set `task1`'s category to SCHOOL using the `setCategory` method and verify the value was set correctly using your `getCategory` method.
- Print `task1` again to verify that your `toString` method still works as expected.

Task 2

- Use the constructor that takes in a description and priority to create a new `Task` object, named `task2` with priority `10`. Use "Give Tigger a bath" as the description of this task. (See sample output below).
- Print `task2` to verify that your `toString` method works as expected.
- Change the priority of `task2` to `20` using the `setPriority` method and verify the value was set correctly using your `getPriority` method.
- Print `task2` again to verify that your `toString` method still works as expected.

Sample Output

When you are done, your output should look something like the following:

```
Task 1
[ ] Finish Activity 6.3, 0, NONE
Task 1 is complete: true
[X] Finish Activity 6.3, 0, NONE
Task 1 category: SCHOOL
[X] Finish Activity 6.3, 0, SCHOOL


Task 2
[ ] Give Tigger a bath, 10, NONE
Task 2 priority: 20
[ ] Give Tigger a bath, 20, NONE
```

Terminology Identification

In your code add comments identifying examples of the following: object, object declaration, object initialization, method invocation. These should be identified with an @keyterm tag within the comment.

Code Review

When you are finished with this activity, pair up with a classmate and review each other's code to make sure it meets all the requirements.

Submission

After completing the assignment, use the assignment link in Canvas and follow the submission instructions there. You will upload your .java files and put your reflection in the "Comments" box.

Reflection Requirements

Write a one paragraph reflection describing your experience with this activity. The reflection should also include the name of your code review partner AND something interesting you found in their code. Please review the activity rubric for more details.