← Project 1: Sets

Due by midnight, Thursday 9/27 (or late Friday)

Please read the assignment carefully! Many questions we will receive are likely answered on this page.

In this assignment, you are given two ADT interfaces, and you will implement them according to their specifications.

Starting point

Download and extract these materials. Contained are:

- `Driver.java`, the provided driver program.
- `Movie.java`, the things that will be held by a `MovieShelf`
- `SetFullException.java`, an exception type
- `SetInterface.java` and `MovieShelfInterface.java`, interfaces which you will implement in…
- `Set.java` and `MovieShelf.java`. **These are the files you will modify.**

Do not modify anything other than `Set.java` and `MovieShelf.java`. We will be testing these with unmodified versions of the other files.

Organization is good. Make yourself a folder for this project if you haven't already.

`Set` [65 points]

See the grading rubric further down to see the breakdown of the points.

You will implement the `Set` class. It implements `SetInterface`. **Read the doc comments for each method in `SetInterface` carefully.** In addition to the interface methods, you will also need to make some constructors, as detailed below.

Never write an entire program before testing it. No one writes a correct program in one shot. Save often. Test early and test often. Read the sections of the textbook starting at **Chapter 2.16.**

Details of your `Set` implementation

- It will store its data in an array.
- It will be dynamic capacity (like we talked about in Lecture 3).
- The `Set(int)` constructor will initialize the array to the given capacity.
    - It will check for an illegal capacity, and throw `IllegalArgumentException` in that case.

- The `Set()` constructor will initialize the array to a *reasonable default capacity*.
  - Reusing methods/constructors is a great habit! See section **D.10** of Carrano (page 877 in the 4th edition) for details.
- The `Set(E[])` constructor will initialize the set to contain the items in the array.
  - You **will not** make this array into the set's backing array. That violates encapsulation.
  - The array may contain duplicates and `null`s. You should ignore these instead of throwing an exception.
- Although you are given `SetFullException`, your `add` method will *never* actually throw it.
  - Since your implementation will dynamically resize its capacity, it can never be "full."

`MovieShelf` [30 points]

You will implement the `MovieShelf` class. It implements `MovieShelfInterface`. Again, read the doc comments in that file carefully.

`MovieShelf` is a *client* of your `Set`. That is, it will have an instance of `Set` and use it to hold its data. This means you are creating this class via *composition.*

*You are writing code that **uses your own code.** We call this "eating our own dogfood". It's kind of a gross term, but hey.*

Details of your `MovieShelf` implementation

- It will store its data in a `Set<MovieShelfItem>`.
- The `MovieShelf()` constructor will initialize that `Set` to be empty.
- The `MovieShelf(Movie[])` constructor will initialize that `Set` with that array of `Movie`s.
- A note on `printAll()`: you can use `Set.toArray()` to get the movies to print, but it is an array of `Object`. You will have to cast each item to a `Movie` to access its movie-specific getters.

Testing

We will be testing your code on the command line. You are welcome to use an IDE, but before submitting, please test that your Java files can be compiled and run on the command line without it.

You are given a simple driver program, **but you should write your own driver as well** to test your code more thoroughly. The driver provided does not test all the possible corner cases, and does not test `Set` directly.

Click here for an example of what the driver will output for a correct implementation.

DO NOT TURN IN CODE THAT DOES NOT COMPILE.

This is not acceptable in this class or in your following classes. Comment out the code that doesn't compile and put comments at the top of your Java files for the grader. You may receive some partial credit for such code.

Code that crashes may also receive partial credit.

Grading Rubric

- **[5 points]:** Submission
    - Incorrectly submitted projects will lose all 5 points.
    - Please follow the submission directions carefully. There's no reason not to.
    - *It's 5 free points, people.*
- **[65 points]:** `Set`
    - **[4]:** `Set()` constructor
    - **[4]:** `Set(int)` constructor
    - **[9]:** `Set(E[])` constructor
    - **[4]:** `int getCurrentSize()`
    - **[4]:** `boolean isEmpty()`
    - **[7]:** `boolean add(E)`
    - **[7]:** `E remove(E)`
    - **[6]:** `E remove()`
    - **[5]:** `void clear()`
    - **[6]:** `boolean contains(E)`
    - **[9]:** `Object[] toArray()`
- **[30 points]:** `MovieShelf`
    - **[3]:** `MovieShelf()` constructor
    - **[4]:** `MovieShelf(Movie[])` constructor
    - **[3]:** `add(Movie)`
    - **[3]:** `remove(Movie)`
    - **[8]:** `printAll()`
    - **[9]:** `borrowMovie()`

Submission

You will submit a ZIP file named `username_proj1.zip` where `username` is your Pitt username.

For example, I would name mine `jfb42_proj1.zip`.

Do not put a folder in the zip file, just the following files:

- `Set.java`
- `MovieShelf.java`
- Your custom driver Java file, if any

Do **not** submit any IDE project files.

Creating a ZIP file

If you've never created a ZIP file before, it's easy to do in modern operating systems.

1. Select all the files you want to add to the ZIP file (the Java files listed above).
2. **Right click on the files,** and...
    - **Windows:** do **Send To > Compressed (zipped) folder**. Then you can name it.
    - **macOS:** do **Compress *n* items**. Then you can rename the `Archive.zip` file.

    ◦ **Linux:** haha who knows

Submit to the Box folder at this link. Drag your zip file into your browser to upload. **If you can see your file, you uploaded it correctly!**

You can also re-upload if you made a mistake and need to fix it.