CS5010, Fall 2019

Assignment 1

Tamara Bonaci
t.bonaci@northeastern.edu

- **This assignment is due by 9am on Tuesday, September 17 (notice the unusual AM deadline).**
- **To submit your solution, please:**
    - **Push your solutions to your designated repo within our course organization.**
    - **Submit your zipped up source code to Bottlenose, under submission for Assignment 1.**

Assignment objectives:

After completing this assignment, you should have:

- Refreshed your memory of Java syntax
- Refreshed your knowledge, and gained practice with object-oriented principles, such as abstraction, encapsulation, inheritance, information hiding and polymorphism
- Gained practice with unit testing
- Gained practice with UML class diagrams
- Gained practice with code documentation and Javadoc

Resources

You might find the following online resource useful while tackling this assignment:

- Java 8 Online Documentation (https://docs.oracle.com/javase/8/docs/api/)
- JUnit 4 Getting Started Online Documentation (https://github.com/junit-team/junit4/wiki/getting-started)
- UML Diagram Online Documentation (https://www.smartdraw.com/uml-diagram/)

Submission Requirements

Your repository should contain a package **`assignment1`**, and within that package, a separate package for every problem.

The package names should follow this naming convention: **`assignmentN.problemM`**, where you replace N with the assignment number, and M with the problem number, e.g., all your code for problem 1 for this assignment must be in a package named **`assignment1.problem1`**.

Additional expectations for every package:

- One `.java` file per Java class

- One `.java` file per Java test class
- One pdf or image file for each UML Class Diagram that you create
- All methods must have tests
- All non-test classes and non-test methods must have valid Javadoc

Your packages should **not** contain:

- Any `.class` files
- Any .html files
- Any IntelliJ specific files

Lastly, a few more rules that you should follow in this assignment ☺ :

- Your classes should have a `public` modifier
- Instance fields should have a `private` modifier
- Static field and methods are not allowed (except for constants)
- Use `this` to access instance methods and fields inside a class

Problem 1

You are interning with a startup company, developing a new transit card (such as Metro Transit Orca card) management system. More specifically, your company is developing an app that would allow a user to interact with, track status of, and deposit money to a transit card.

A transit card consists of:

- **A card owner**. A card owner is an individual with:
  - A first name
  - A last name
  - An address
  - An email address
- **A current card balance**, where the balance consists of:
  - An integer value for the dollar amount greater or equal to 0
  - An integer value for the cents amount, greater or equal to 0 and less than or equal to 99.

Your tasks:

1. Write a code to implement a transit card.
2. A transit card allows a customer to deposit money to the card. This functionality is implemented in method `depositMoney(Deposit newDeposit)`, that takes `Deposit` as an input argument. Deposit consists of:
   a. Deposit amount that is in the range [(5 dollars, 0 cents), (50 dollars, 0 cents)].
   b. The information about the first and the last name of the owner of the card that the money is being deposited to.

   For security and privacy reasons, the method checks that the provided information about card owner matches the information on file. Please implement the method `depositMoney(Deposit newDeposit)`.

3. Write tests for your class `TransitCard`, and all dependent classes.
4. Provide your final UML diagram that includes method `depositMoney(Deposit newDeposit)`.

Problem 2

A non-profit organization in Seattle is trying to organize a digital index of all artists who were born, and/or lived and created their art in Seattle. They have an idea how they'd like to go about it, but they need your help with implementation. Here's their idea:

An `Artist` can be:

- An `Actor`
- A `Musician`
- A `Painter`
- A `Photographer`
- A `Filmmaker`
- A `Dancer`
- A `Poet`

All `Artists` contain the following information:

- `Name`, containing information about an `Artist's` first and last name
- `Age`, which is an Integer in the range [0, 128], containing information an `Artist's` age
- `Genre`, containing information about an `Artist's` genres
- `Awards`, containing information about all awards that an `Artist` received

Additionally, all `Actors`, `Dancers` and `Filmmakers` keep track of the following information:

- `Movies`, containing information about all movies that they worked on (acted in)
- `Series`, containing information about all TV series that they worked on (acted in)
- `Other multimedia`, containing information about all other multimedia content that they worked on (acted in)

Your tasks:

3. Design Java classes to capture the above requirements and information.
4. Please write appropriate Javadoc documentation for all of your classes and methods.
5. Please write the corresponding test classes for all of your classes.
6. Please provide a final UML Class Diagram for your design.

Problem 3

You are a part of data analysis company that keeps track of scientific publications, and estimates their impact on the scientific community (you can imagine your company's system to be similar to Google Scholar). You are expected to develop a part of the system that estimates the **impact** of scientific publications.

The system **estimates the impact** of the following `Publications`:

- `Article`
- `Book`

An `Article` can be one of:

- `Conference proceeding`
- `Journal`

A `Book` can be one of:

- `Textbook`
- `Edited volume`

For every `Publication`, the system keeps track of the following:

- `Title`, represented as a `String`
- `Authors`, represented as an array of `Strings`
- `Publishing year`, represented as an `Integer`
- `Number of citations`, represented as an `Integer`.

Additionally, for every `Book`, the system also keeps track of:

- `Publishing company`, represented as a `String`
- `Number of pages`, represented as an `Integer`.

For every `Journal`, the system keeps track of:

- `Publisher`, represented as a `String`
- `Editors`, represented as an array of `Strings`

Finally, for every `Conference proceeding`, the system additionally keeps track of:

- `Conference name`, represented as a `String`.
- `Conference location`, represented as a `String`.

Your tasks:

1. Design Java classes to capture the above requirements and information.
2. The impact estimation system calculates the impact of a scientific publication according to the following rules:
    a. **Base impact:** the base impact of every scientific publication is estimated by dividing that publication's number of citations with an age of that publication.

b. **Age of publication:** the age of a publication is found as a difference between the current year (2019) and the publication's publishing year. If the publishing year is also 2019, the age of the publication is set to 1.
c. **Impact of journals:** historically, an impact of books and journals has been considered higher than an impact of conference proceedings. To account for that, a total impact of a journal is computed by multiplying the base impact by factor 2.
d. **Impact of books:** similarly, a total impact of a book is computed by multiplying the base impact by factor 4.
e. **Impact of newer publications:** additionally, newer publications are typically considered to be more impactful than older. To account for that, every publication younger than three years gets a **"freshness bump"**, where constant 100 is added to the current impact of that publication.
f. **Old publications' exceptions:** lastly, it has been shown that the presented rules do not provided meaningful estimation of impact for publications older than 250 years. So, if a publication is older than 250 year, the system should not try to estimate the impact according to the given rules. Instead, the system should throw a new `ImpactEstimationException`.

Design and implement method `estimateImpact()`, whose return type is `Double`. That means that, when we call method `estimateImpact()` on some `Publication`, it should return the impact of the given scientific publication as an `Double`.

Your tasks - continued:

3. Please write appropriate Javadoc documentation for all of your classes and methods.
4. Please write the corresponding test classes for all of your classes.
5. Please provide a final UML Class Diagram for your design.