CMSC 256 – Project 1

Programming Assignment 1

Note: When you turn in an assignment to be graded in this class, you are making the claim that you neither gave nor received assistance on the work you turned in (except, of course, assistance from the instructor or teaching assistants).

Program: **Inheritance Implementation**

Part 1 – DigitalMedia Super Class

Define a super class to model digital media. Sound, images, text and video are all available in digital format for use on a computer system. The class **DigitalMedia** is a generalization of these kinds of files. It has four fields: **name**(String) is the file name that includes the file extension, **size**(long) is the file size in bytes.

Design the class so that individual instances never contain null fields. The parameterized constructor must validate both parameters before creating the instance.

All classes will be graded using a JUnit test suite, so the method headings for all methods need to match the test class **exactly**. Refer to the UML diagram to confirm the method headings of your code.

Implement accessor (getter) and mutator (setter) methods for each of the instance variables using the standard naming conventions. In addition, override the following methods from the Object class:

- **public boolean equals(Object other)**
- **public String toString()**

The **equals** method parameter must be of type **Object**. Two **DigitalMedia** objects are equal if they have the same state as represented by the instance variables. Guidelines for overriding the **equals** method:

- Use the == operator to check if the argument is a reference to itself.
- Use the **instanceof** operator to check if the argument has the correct data type– DigitalMedia in this case.
- Cast the argument to the same data type as the class – DigitalMedia in this case.
- For each significant data member, test for equality.
- Test these three properties:
    - Is it symmetric?
    - Is it transitive?
    - Is it consistent?

Your source code file name should be **DigitalMedia.java**.

Part 2 – Song and Image Subclasses

Write the following Java subclasses of the DigitalMedia class from Part 1:

- **Song** to represent a song in a music collection.
    - In addition to the inherited instance variables from the super class, there are two additional data members of the Song class. These are String objects representing the artist's name (**`artist`**), and the album where the song can be found (**`album`**). These instance variables for the class are to have private access modifiers so you will need to write the appropriate methods to allow a client to access and modify the data values.
    - In accordance with good programming practice, you are to override the equals and toString() methods inherited from the DigitalMedia class.
        - A Song's title is the same as the name instance variable in the super class without the extension.
        - Two Song objects are considered equal if they have the same title, artist, album, and size.
- **Image** to represent a digital image file.
    - An image has the additional instance variables **width** and **height**, both of type **int**.
    - As for the Song class, implement getters, setters, equals, and toString() for the Image class. Two images are considered equal if they have the same name, size, width, and height

CMSC 256 – Project 1

Part 3 – MediaList Class

Implement a class called **MediaList** that contains a main method that reads in a file name **via the command line**. The file is a listing of songs and images. In the file, information about each media item is on a single line.

For Songs, the line is formatted with the letter **s**, song file name, the artist name, the album, and size in which each element is separated by a colon (:) and varying amounts of white space. For example,

S : Mykonos.mp3 : Fleet Foxes :Mykonos - Single: 9507904

S : He Doesn't Know Why.mp3 : Fleet Foxes :Fleet Foxes : 7080711

For Images, the line is formatted with the letter **i**, the image file name, the width, the height, and size in which each element is separated by a colon (:) and varying amounts of white space. For example,

I : Beach.jpg : 2048 : 1536 : 1163207

I : IMG_3658.JPG: 2448 : 3264 : 1384318

The **MediaList** program is to:

- Read in the input file creating Song and Image objects from the data contained in the lines of the file. Failure to create individual Song/Image objects will result in severe point penalties.
- Store the Songs and Images in a single ArrayList of type DigitalMedia. Using multiple ArrayLists is not permitted.
- Prompt the user to enter either S or I to see a display of all of the DigitalMedia of that type.
- If "I" is entered, display a formatted list of all the images by calling the toString() method for each Image object.
- If "S" is entered, display a formatted list of all the songs by calling the toString() method for each Song object.

Make sure your program is well documented - including the comment block header at the top of all of the source code files with your name, the course number and name, the project name, the program purpose. Be sure to include appropriate comments throughout your code, choose meaningful identifiers, and use indentation as shown in your textbook and in class.

It is expected that your program will be well documented and you are required to include a private helper method in **MediaList.java** called **printHeading** that outputs the following information to the console in an easy-to-read format: your name, the project number, the course identifier, and the current semester. You will call this method as the first statement in your **main** method.

You will upload the project source code files, **DigitalMedia.java, Song.java, Image.java,** and **MediaList.java** to the Assignment link in Blackboard.

CMSC 256 – Project 1

Name: _____

Part 1 - DigitalMedia class:

Class elements (instance variables, constructor, accessor methods) are correct (5 pts.) _____

equals method written correctly (5 pts.) _____

toString() method written correctly (5 pts.) _____

Class design includes appropriate code to uphold encapsulation (5 pts.) _____

Part 2 – Song and Image Subclasses

Song class:

Class elements (instance variables, constructor, accessor methods) are correct (5 pts.) _____

equals method written correctly (5 pts.) _____

toString() method written correctly (5 pts.) _____

Class correctly implemented as a subclass, using principle of inheritance (5 pts.) _____

Image class:

Class elements (instance variables, constructor, accessor methods) are correct (5 pts.) _____

equals method written correctly (5 pts.) _____

toString() method written correctly (5 pts.) _____

Class correctly implemented as a subclass, using principle of inheritance (5 pts.) _____

Part 3 – MediaList Class

printHeading() method included and called from main() (5 pts.) _____

Reads command line argument for file name (5 pt.) _____

Accurately reads data from file to create objects as specified(10 pt.) _____

All classes:

Code written to anticipate and handle improper data and other errors (10 pt.) _____

Files are named and submitted as specified (5 pts.) _____

Appropriate use of comments (including header comments on all files) (5 pts.) _____

Total (100 pts.) _____

Grading Comments: