ITI 1121. Introduction to Computer Science II

Laboratory 4

Winter 2014

Objectives

- Creating a hierarchy of classes
- Further understanding of inheritance

Introduction

This laboratory consists of creating a hierarchy of classes for representing documents. The hierarchy of documents consists of a superclass called **Document**. All the Documents have a **name**, an **owner** and an **index** number. The class **Document** is specialized into **MediaDocument**. All MediaDocuments have a **duration**. The calculation of the rating of a document depends on the specific type of **MediaDocument**, namely **Movie** or **Audio**. Therefore, the implementation of the method **getRating()** is done in the classes **Movie** and **Audio**. The implementation of the classes must follow the principles of encapsulation presented in class.

1 Document

Implement a concrete class, called **Document**, to represent the characteristics that are common to all the documents. Namely,

1. All documents must have a unique identifier (of type **int**). The first document created will have the **id** 100, the **id** of the next one will be **101**, and so on. A newly created object has a unique **id** which is one (1) more than the id of the last object that was created;
2. All documents have a **name** and an **owner** (both of type **String**);
3. The class **Document** has one constructor. Its signature is as follows: **Document(String name, String owner)**. The constructor must initialise the **name** and **owner** of the document using the given parameters, as well as initialising the unique identifier of this **Document**;
4. **public int getIndexNumber()**: returns the unique identifier, here called index number, of this document;
5. **public String getName()**; returns the name of this document;
6. **public void rename(String name)**: changes the name of the document to **name**;
7. **public String getOwner()**: returns the owner of this document;
8. **public void changeOwner(String owner)**: changes the owner of the document to **owner**;
9. **public boolean equals(Document other)**: returns **true** if **other** designates a document, and that document has the same index number (unique identifier);
10. **public String toString()**: returns a **String** representation of this **Document**, consisting of the index number, document name and owner.

Files

- `Document.java`

2 MediaDocument

Implement a class, called **MediaDocument**, to represent the characteristics that are common to all the media documents. All the media documents have a method **int getRating()**, however the implementation of the method is specific to the kind of document.

1. A **MediaDocument** is a specialised kind of **Document**. Therefore, it must be a subclass of **Document**;
2. All media documents have a **duration** (of type **int**);
3. The class **MediaDocument** has a single constructor. Its signature is as follows: **public MediaDocument(String name, String owner, int duration)**. It initialises the characteristics that are common to all Documents, as well as the duration.

Files

- `Document.java`
- `MadiaDocument.java`

3 Movie

A **Movie** is a specialised **MediaDocument** that also has information about the **rating** of the story and **acting**. More precisely,

1. A **Movie** is a (concrete) subclass of **MediaDocument**;
2. It also stores information about the **rating** of the **story** and the **acting** (both of type **int**). This information (two numbers in the range 1 …10) is given as an argument to the constructor;
3. It has a single constructor and here is its signature: **public Movie(String name, String owner, int duration, int story, int acting)**. It serves to initialise all the characteristics that are common to all media documents, as well as the story and acting ratings. You can assume that the numbers, ratings, are valid;
4. **public int getStoryRating()**: returns the story rating;
5. **public int getActingRating()**: returns the acting rating;
6. It implements the method **public int getRating()**, which returns the average of the **story** and **acting** ratings rounded to the nearest integer.

Files

- `Document.java`
- `MadiaDocument.java`
- `Movie.java`

4 Audio

Audio is a specialised **MediaDocument** that also has information about the rating of this **Audio** document. More precisely,

1. Audio is a (concrete) subclass of **MediaDocument**;
2. It stores the rating (an **int**) of this document. The rating of an **Audio** document is a single number, the overall appreciation of the piece;
3. It has a single constructor. Its signature is: **public Audio(String name, String owner, int duration, int rating)**. It initialises the properties that are common to all media documents, as well as the rating of this document;
4. It implements the method **public int getRating()**, which returns the rating of this document. In the case of an **Audio** document, the rating is a single number, therefore, the method simply returns this number.

In a "real-world" application, the documents would contains additional attributes and methods (at least some content!). However, in the context of this laboratory, we will limit ourselves to those attributes and methods.

5 Test

Create a test program. In the main method, declare an array to store **MediaDocument** objects. Fill the array with **Movie** and **Audio** documents. Finally, write a loop that caclcutes the sum of all the ratings.

Files

- `Document.java`
- `MadiaDocument.java`
- `Movie.java`
- `Audio.java`
- `Test.java`

6 Quiz (1 mark)

- To invoke a parents constructor in a subclass, we use the _____ method.
    1. abstract
    2. construct
    3. parent
    4. super
    5. extends

    Answer:

Submit your answer using Blackboard Learn:

- https://uottawa.blackboard.com/

Last Modified: January 30, 2014