COMI 2510 Advanced Programming and Design
Assignment 1

You're going to write an application that displays information about movies, including movie ratings.

Create the following classes:

- `Rating`. `Rating` has two fields: `total` and `thumbsUp`. The `Rating` class should have a method that returns the percentage of total ratings that are thumbs-up ratings as well as a `toString` method.

- `Movie`. `Movie` has two `Rating` objects as fields: `viewerRating` and `criticRating`, and fields for the movie title and year of release. The `Movie` class should have a toString method that returns a String composed of the title, year of release, and both ratings.

Write these classes as designed. Do not add or remove fields or methods. Be sure to follow the guidelines outlined for appropriate class design in the textbook.

Write an application that creates the following movies and associated ratings objects, and then invokes the toString method of all movies and displays the results:

Title: Invasion of the Body Snatchers
Release year: 1978
Ratings:
Viewer: Total: 35,304; Thumbs-up: 26,125
Critic: Total: 48; Thumbs-up: 45

Title: Inherit the Wind
Release year: 1960
Ratings:
Viewer: Total: 9,492; Thumbs-up: 8,638
Critic: Total: 22; Thumbs-up: 20

Title: Erin Brockovich
Release year: 2000
Ratings:
Viewer: Total: 128,955; Thumbs-up: 94,137
Critic: Total: 143; Thumbs-up: 120

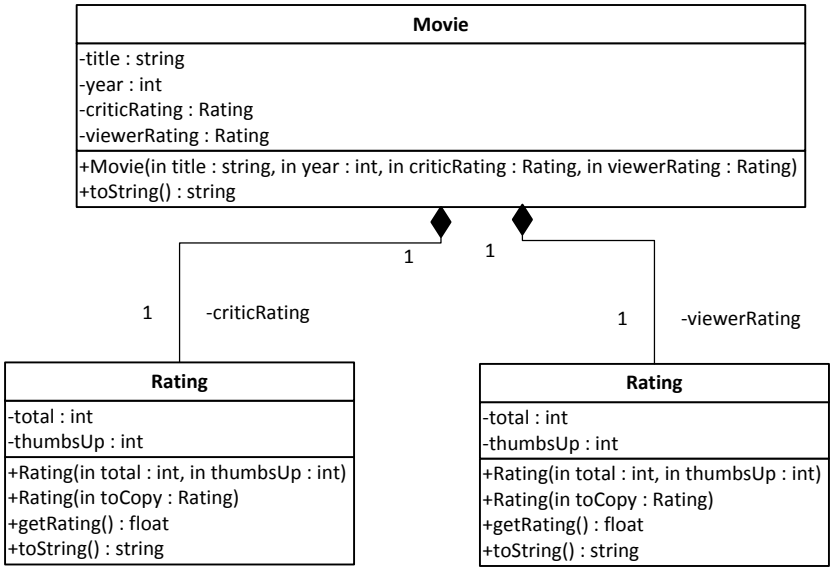
Turn in:

- Your three classes (`Rating`, `Movie`, and the class you design for your application) (.java files only).
- A sequence diagram of your program creating one movie (and its two ratings), invoking its `toString` method, and displaying output.
- Object diagrams for your program executing (show objects after instantiation) and the output of the program for that instance of execution.

Here are class diagrams for `Rating` and `Movie`.

Please note that the text shows, in its class diagrams, an aggregate relationship with the name of the field within the top part of the class diagram, but also as a separate class (e.g. Figure 8-14 UML diagram showing aggregation). I do not believe this is standard (e.g. Figure 13 here http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/).

This diagram shows the aggregate class twice (because there are two instances of the same class as part of the Movie class) and also within the Movie class diagram. Please don't hesitate to ask questions if this diagram isn't clear.



Assignment 1 Rubric

Component		Quality			
		Exceptional	**Acceptable**	**Amateur**	**Unsatisfactory**
	Run-time specifications 50%	50 pts: The program meets all of the run-time specifications, with no additional unspecified functionality.*	40 pts: There is additional unspecified functionality or the program produces incorrect results in no more than 5% of the customer's tests.	25 pts: The program produces incorrect results in no more than 10% of the customer's tests.	10 pts: The program produces incorrect results in more than 10% of the customer's tests.
	Design specifications 25%	25 pts: The program meets design specifications. No fields or methods are added or removed.	20 pts: Classes mostly meet specifications but are off by less than 10% (e.g. additional or missing field or method, incorrect type field or method).	12 pts: Classes mostly meet specifications but are off by less than 20%.	5 pts: Classes do not meet specifications more than 20% of the time.
	Diagramming techniques 12.5%	12.5 pts: Diagramming techniques are used appropriately and accurately reflect the static or dynamic state of the program.	10 pts: Diagramming techniques are used appropriately but with minor errors that don't affect readability and accurately reflect the static or dynamic state of the program or have minor errors that don't affect comprehension of the system state.	6.25 pts: Diagramming techniques are used appropriately but with several errors that somewhat affect readability or have minor errors that affect comprehension of the system state in less than 20% of the execution portrayed.	2.5 pts: Diagramming techniques have major errors that affect readability or do not accurately reflect the static or dynamic state of the program that affect comprehension of the system state. (e.g. wrong data in object diagram, objects not present in sequence diagram that are part of the execution sequence, missing messages)
	Documentation 12.5%	12.5 pts: The program contains comments including the programmer's name and date. Javadoc comments are included as shown in the text for all classes. There are block comments (as many as necessary) for each distinct block of code which accurately describe what the block is accomplishing.	10 pts: The header comment is incomplete but contains name and date, and/or the block comment(s) aren't clear. Javadoc comments are missing components less than 10% of the time.	6.25 pts: The documentation partially meets the exceptional guidelines with several poorly written comments and/or missing comments or missing components less than 25% of the time.	2.5 pts: 25% or more of the comments are missing or unhelpful.

*If you want to change the functional specifications of the program in any way, you must clear it with your customer (the instructor) in writing prior to making the changes. Include documentation of the specification changes when you submit your program.