## Problem 2 – MyTunes Shop

The local community is really happy by your implementation of the music shop. It is thriving and has a lot of customers. Your fame as a developer and musician has spread throughout the whole neighborhood. Now the people have assigned you another similar task. You've got to help build a system for selling songs and albums for a local shop called **MyTunes**. Someone has previously worked on it but didn't manage to finish it on time.

The system consists of **songs**, **performers**, and **albums**. Songs and albums are some types of **media**.

* A **media** is basically something which the shop can sell and has a **title** and **price**.
* A **song** is the smallest unit the shop can sell. Each song has a **title**, **price**, **performer**, **genre**, and **release year**. Songs may exist in **albums**.
* A **performer** has a name and some songs. It can be a **singer** or a **band**. Bands hold information about their members' names.

Some things in the shop **can be rated**. They contain information about the ratings they've got. A rating is simply a number between one and five.

The system currently supports the following functionality:

* **Song** is a type of media which contains basic information about its artist, duration, etc.
* **Singer** is a type of performer which has only one name.

You are given an engine which can execute commands. The commands this engine currently supports are:

* **insert:performer;singer;<name>** – Adds a new singer to the database
* **insert:media;song;<title>;<price>;<performer\_name>;<genre>;<year>;<duration>** – Adds a new song to the database (with zero initial quantity). If the performer does not exist, prints an error message
* **supply:song;<title>;<quantity>** – Increases the supply of the song by the provided quantity. If the song does not exist, prints an error message
* **sell:song;<title>;<quantity>** – Sells the specified quantity of the specified song. If the song does not exist, prints an error message
* **report:performer;singer:<name>** – Prints basic information about the specified singer, and all his / her songs, sorted alphabetically. If the singer does not exist, prints an error message
* **report:media;song:<title>** – Prints basic information about the specified song. If the song does not exist, prints an error message
* **rate** – The developer didn't have time to finish this command

The engine skips all invalid commands. It also handles invalid data passed as the command parameters. Commands can be separated by colons ("**:**") and / or semicolons ("**;**").

You are also given a simple printer utility which makes printing messages easier.

Your task is to study how the system currently works and to extend it, using the **best practices** of **Object-Oriented Design (OOD)** and **Object-Oriented Programming (OOP)**. You have to use **OOP principles** in order to **avoid code repetition** and **allow code reuse**.

**You are NOT allowed to change any of the provided classes or interfaces**. You may edit **only one line** in the **Main()** method. Additionally you are allowed to edit the **Song** class and **ISong** Interface.

You should implement a set of things, given below. You should also implement a way for the engine to recognize the new commands that will be available in the new, extended version of the system. Refer to the sample input and output to get more information about how the engine will have to work.

#### Rateable song

* Make the existing song **rateable**. This is like a normal song but can be rated.
  + The song must be rateable.  
    Throw an **ArgumentException** with the message "The song must be rateable."
  + Rating will always be a number between 1 and 5. There is no need to check it explicitly.

**rate:song;<name>;<rating>** **TODO**

* + In case of success, print **"The rating has been added successfully."**
* Implement reporting for rateable song. It should be in the following form:

|  |
| --- |
| **<title> (<year>) by <performer\_name>**  **Genre: <genre>, Price:$<price>**  **Rating: <average\_rating>**  **Supplies: <supplies>, Sold: <quantity\_sold>** |
| **// TODO** |

#### Album

* Implement an **album**.
  + Contains one or more songs.
  + It is a type of **media**, and has title, price, performer (singer or band), genre, year, and **collection of songs**.

**insert:media;album;<title>;<price>;<performer\_name>;<genre>;<year>.**

* + In case of success, print **"Album <name> by <performer\_name> added successfully**"
* Implement adding songs to albums.

**insert:****song\_to\_album;<album\_name>;<song\_name>**.

* + In case of error, print **"The album does not exist in the database."**or **"****The song does not exist in the database."**
  + In case of success, print  
    **"****The song <song\_title> has been added to the album <album\_title>."**
* Implement reporting for albums. It should be in the following form: TODO

|  |
| --- |
| **<title> (<year>) by <performer\_name>**  **Genre: <genre>, Price:$<price>**  **Supplies: <supplies>, Sold: <quantity\_sold>**  **Songs: / No Songs**  **<song\_1\_title> (<song\_1\_duration>)**  **<song\_2\_title> (<dong\_2\_duration>)...** |

* The songs should be in the order they have been added in the system.
* In case there are no songs, print "**No songs**".
* In case of error, print **"The album does not exist in the database."**

**Note:** Selling songs and selling albums are completely different – selling an album does not change the quantity of the 'standalone' songs in the system.

You should implement **supplying** and **selling** of albums (the commands are the same like for songs, but with **album** instead of **song**).

#### Band

* Implement a **band**.
  + A band is a type of **performer** and has **name** and collection of **members**

(Note: use strings for the member names, you don't need to have them in the database as singers).

**insert:performer;<name>.**

* Implement adding band members.

**insert:member\_to\_band;<band\_name>;<performer\_name>.**

* + Note that you need to check for existing band, but **NOT** for existing singer.
  + In case of non-existing band, print **"****The band does not exist in the database."**.
  + In case of success, print **"****The member <member\_name> has been added to the band <band\_name>."**
* Implement reporting for bands. It should be in the following form: TODO

|  |
| --- |
| **<name> (<member\_1, member\_2...>): <song\_1\_title>; <song\_2\_title> / no songs** |

* The members should be listed in order of addition to the system.
* If there are no members currently in the band do not print the parenthesis.
* In case the band does not exist, print **"The band does not exist in the database."**

### Constraints

* All prices are displayed with two numbers after the decimal point.
* All ratings are rounded to the nearest integer.
* Please take care to print the right messages.
* Take special care whether there is a full stop at the end of each message.
* The output will always be valid.

### Sample Input

|  |
| --- |
| insert:performer;singer;Sully Ernareport:performer;singer;Sully Ernainsert:media;song;Until Then;0.99;Sully Erna;Rock;2010;5.09sell:song;Until Then;20report:media;song;Until Thensupply:song;Until Then;50sell:song;Until Then;20insert:media;song;7 Years;0.99;Sully Erna;Rock;2010;8.40report:performer;singer;Sully Ernareport:media;song;Until Thensupply:song;7 Years;30sell:song;7 Years;10rate:song;7 Years;5rate:song;7 Years;4report:media;song;Until Thenreport:media;song;7 Yearsinsert:media;album;Avalon;10.99;Sully Erna;Rock;2010report:media;album;Avaloninsert:song\_to\_album;Avalon;Until Theninsert:song\_to\_album;Avalon;7 Yearsreport:media;album;Avaloninsert:performer;band;Godsmackreport:performer;band;Godsmackinsert:member\_to\_band:Godsmack:Sully Ernainsert:member\_to\_band:Godsmack:Shannon Larkininsert:member\_to\_band:Godsmack:Robbie Merrillinsert:member\_to\_band:Godsmack:Tony Rombolareport:performer;band;Godsmackinsert:media;song;Livin' In Sin;1.99;Godsmack;Hard Rock;2006;4.39insert:media;song;Speak;2.99;Godsmack;Hard Rock;2006;3.57insert:media;song;The Enemy;0.999;Godsmack;Hard Rock;2006;4.07insert:media;song;Voodoo Too;0.98;Godsmack;Hard Rock;2006;5.26insert:media;song;One Rainy Day;1.39;Godsmack;Hard Rock;2006;7.21insert:media;album;IV;15.99;Godsmack;Hard Rock;2006insert:song\_to\_album;IV;Livin' In Sininsert:song\_to\_album;IV;Speakinsert:song\_to\_album;IV;The Enemyinsert:song\_to\_album;IV;Voodoo Tooinsert:song\_to\_album;IV;One Rainy Daysupply:album;IV;30sell:album;IV;10report:performer;band;Godsmacksupply:song;Speak;30sell:song;Speak;10rate:song;Speak;5report:media;song;Speakreport:media;album;IVend |

### Sample Output

|  |
| --- |
| Singer Sully Erna added successfully  Sully Erna: no songs  Song Until Then by Sully Erna added successfully  There are not enough supplies.  Until Then (2010) by Sully Erna  Genre: Rock, Price: $0.99  Rating: 0  Supplies: 0, Sold: 0  50 items of song Until Then successfully supplied.  20 items of song Until Then successfully sold.  Song 7 Years by Sully Erna added successfully  Sully Erna: 7 Years; Until Then  Until Then (2010) by Sully Erna  Genre: Rock, Price: $0.99  Rating: 0  Supplies: 30, Sold: 20  30 items of song 7 Years successfully supplied.  10 items of song 7 Years successfully sold.  The rating has been placed successfully.  The rating has been placed successfully.  Until Then (2010) by Sully Erna  Genre: Rock, Price: $0.99  Rating: 0  Supplies: 30, Sold: 20  7 Years (2010) by Sully Erna  Genre: Rock, Price: $0.99  Rating: 5  Supplies: 20, Sold: 10  Album Avalon by Sully Erna added successfully  Avalon (2010) by Sully Erna  Genre: Rock, Price: $10.99  Supplies: 0, Sold: 0  No songs  The song Until Then has been added to the album Avalon.  The song 7 Years has been added to the album Avalon.  Avalon (2010) by Sully Erna  Genre: Rock, Price: $10.99  Supplies: 0, Sold: 0  Songs:  Until Then (5.09)  7 Years (8.40)  Band Godsmack added successfully  Godsmack:  no songs  The member Sully Erna has been added to the band Godsmack.  The member Shannon Larkin has been added to the band Godsmack.  The member Robbie Merrill has been added to the band Godsmack.  The member Tony Rombola has been added to the band Godsmack.  Godsmack: Sully Erna, Shannon Larkin, Robbie Merrill, Tony Rombola  no songs  Song Livin' In Sin by Godsmack added successfully  Song Speak by Godsmack added successfully  Song The Enemy by Godsmack added successfully  Song Voodoo Too by Godsmack added successfully  Song One Rainy Day by Godsmack added successfully  Album IV by Godsmack added successfully  The song Livin' In Sin has been added to the album IV.  The song Speak has been added to the album IV.  The song The Enemy has been added to the album IV.  The song Voodoo Too has been added to the album IV.  The song One Rainy Day has been added to the album IV.  30 items of album IV successfully supplied.  10 items of album IV successfully sold.  Godsmack: Sully Erna, Shannon Larkin, Robbie Merrill, Tony Rombola  Livin' In Sin; One Rainy Day; Speak; The Enemy; Voodoo Too  30 items of song Speak successfully supplied.  10 items of song Speak successfully sold.  The rating has been placed successfully.  Speak (2006) by Godsmack  Genre: Hard Rock, Price: $2.99  Rating: 5  Supplies: 20, Sold: 10  IV (2006) by Godsmack  Genre: Hard Rock, Price: $15.99  Supplies: 20, Sold: 10  Songs:  Livin' In Sin (4.39)  Speak (3.57)  The Enemy (4.07)  Voodoo Too (5.26)  One Rainy Day (7.21) |