# Technical test

# Context

A bookstore wants to develop a new computer tool to:

* Import information from its stock
* Get information about a particular book
* Calculate the price of a basket of books

## You are in charge of the development of this tool whose functionalities are detailed below. You are requested to implement the different methods exposed by the interface described in Appendix 4, following the specifications below.

## Exercices

### Import data stock

The tool must be able to import the stock data from the library from a JSON file whose format is specified in Appendix 1. It is not required to store the imported information in a database, but only to keep it in memory.

For this exercise, you must implement the following method:

void Import(string catalogAsJson)

An example of JSON is available in Appendix 2.

### Retrieving information

The bookstore manager must be able to consult, by a book title, the available stock. This would be made from the following method:

int Quantity(string name)

This method takes the title of a book as parameter and returns the number of copies available (i.e. the "Quantity" field in the JSON).

*Example for a call:*

*Store.Quantity("Ayn Rand - FountainHead"); Rеsult:*

*10*

### Calculation of the price of the baskets

The most important feature of the tool is the basket price calculation (i.e. a selection of books). This will be done by calling the following method:

double Buy(params string[] basketByNames)

The calculation of the price of the baskets must follow very specific rules which are described in Appendix 3.

*Example for a call:*

Store.Buy("J.K Rowling - Goblet Of fire", "Isaac Asimov - Foundation")*;*

*Rеsult:*

24.00

# Appendix 1 – JSON format

{

"type": "object",

"title": "The Root Schema", "required": [

"Category", "Catalog"

],

"properties": { "Category": {

"type": "array",

"title": "List of existing category with associated discount", "items": {

"type": "object",

"title": "one category with its discount", "required": [

"Name", "Discount"

],

"properties": { "Name": {

"type": "string",

"title": "The unique name of the category, it is a functionnal

key",

"default": "", "examples": [

"Fantastique"

],

"pattern": "^(.+)$"

},

"Discount": { "type": "number",

"title": "the discount applies when buying multiple book of this

category",

"default": 0.0,

"examples": [ 0.05

]

}

}

}

},

"Catalog": { "type": "array",

"title": "The Catalog of the store", "items": {

"type": "object",

"title": "a book in the catalog", "required": [

"Name", "Category", "Price", "Quantity"

],

"properties": {

"Name": {

"type": "string",

"title": "The unique Name of the book, it is a functionnal key", "default": "",

"examples": [

"J.K Rowling - Goblet Of fire"

],

"pattern": "^(.+)$"

},

"Category": { "type": "string",

"title": "The name of one the category existing in the Category root properties.",

"default": "", "examples": [

"Fantastique"

],

"pattern": "^(.+)$"

},

"Price": {

"type": "number",

"title": "the price of an copy of the book", "default": 0,

"examples": [ 8

]

},

"Quantity": { "type": "integer",

"title": "The Quantity of copy of the book in the catalog.", "default": 0,

"examples": [ 2

]

}

}

}

}

}

}

# Appendix 2 – JSON Example

{

"Category":[

{

"Name": "Science Fiction", "Discount": 0.05

},

{

"Name": "Fantastique", "Discount": 0.1

},

{

"Name": "Philosophy", "Discount": 0.15

},

],

"Catalog": [

{

"Name": "J.K Rowling - Goblet Of fire", "Category": "Fantastique",

"Price": 8,

"Quantity": 2

},

{

"Name": "Ayn Rand - FountainHead", "Category": "Philosophy",

"Price": 12,

"Quantity": 10

},

{

"Name": "Isaac Asimov - Foundation", "Category": "Science Fiction", "Price": 16,

"Quantity": 1

},

{

"Name": "Isaac Asimov - Robot series", "Category": "Science Fiction", "Price": 5,

"Quantity": 1

},

{

"Name": "Robin Hobb - Assassin Apprentice", "Category": "Fantastique",

"Price": 12,

"Quantity": 8

}

],

}

# Appendix 3 – Price calculation rule

In order to calculate the price of a basket several rules apply:

1. The purchase of a single book is paid at the price of the book provided in the catalog.
2. If the customer buys several books then a discount applies if these books belong to the same category

*Example: If the customer buys two different books in the Fantastic category a discount of 10% will apply.*

1. Only the first copy of each book has the right to the reduction

*Example 1 : if a customer buys a copy of JK Rowling - Goblet Of Fire and two of Robin Hobb - Assassin Apprentice, only Goblet of fire and one of the two copies of Robin Hobb will be entitled to the discount (which would give a price of 30 = 8 \* 0.9 + 12 \* 0.9 + 12).*

*Example 2 : if a customer buys a copy of Rand, the two works of Asimov (a copy of Isaac Asimov - Robot series and one of Isaac Asimov - Foundation), the book of Rowling in two copies and that of Hobb in two copies, he must pay 69.95 € = 12 + 5 \* 0.95 + 16 \* 0.95 + 8 \* 0.9 + 8 + 12 \* 0.9 + 12*

1. If a shopping cart is invalid because the catalog does not contain enough books (for example if the customer wants to buy two copies of Isaac Asimov Foundation), the owner expects to receive a NotEnoughInventoryException containing the list of books not found.

# Appendix 4 – Definition of interfaces and classes

public interface IStore

{

void Import(string catalogAsJson); int Quantity(string name);

double Buy(params string[] basketByNames);

}

public class NotEnoughInventoryException : Exception

{

public IEnumerable<INameQuantity> Missing { get; }

}

public interface INameQuantity

{

string Name { get; }

int Quantity { get; }

}