ESS 201 Programming in Java
T1 2020-21
Lab 2
12 Aug 2020

Part A. (To be worked on during the Lab class)

1. We have a set of apartments and a set of owners. An Owner can own more than one apartment. For each apartment, we are given the owner and the rent. An owner has a name and age. Find the apartment with the highest rent and print the name of its owner.

You should have a class Apartment that contains the rent (int) and a reference to an Owner. The Owner class maintains the name (String) and age (int) of the owner. This class should have methods getName and getAge that allow its name and age to be queried.

The attached file Apartment.java has the skeleton of the code. Fill in details of the data members, as well as implementation of the methods, and then fill out the missing code in **main**.

2. Given a String that contains a sentence, print the words of the sentence in reverse order. That is, the sentence *"print the words of the sentence in reverse order*" should be printed out as

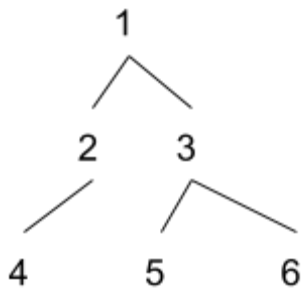
```
order reverse in sentence the of words the print
```

Explore the methods of the String class, and specifically the split() method. You can assume that the input method has only words and blanks, and no punctuations.

Create a class in a new file, with a main method as in earlier examples. Initialize a String variable to the value "the language Java the language is named after the island Java". Extract the words and print them in reverse order (to standard output) as described above.

3. A Tree data structure has a simple recursive structure. We can model this as a Java class, where each node has references to instances of the same Tree class.

Create a binary tree, class Tree, that has two children, leftChild and rightChild, both also of type Tree. The Tree class also contains data - an integer value. Note that either or both of the children can be null. Write a method *traverse* that will traverse the tree (recursively) and print out values in pre-order. The Tree class should have a constructor that takes two Tree objects as arguments - the left and right child - either or both could be null.



Construct the tree object in the figure, where the numbers at the nodes are the values being stored at that node. The numbers should be written out in one line, separated by a single space. Thus the output for this figure would be

1 2 4 3 5 6

The main should create each instance of the Tree nodes and create the tree structure by setting the appropriate children for each node.

Part B (To be submitted in one week)
The main intent of this exercise is to get used to the object oriented style of designing solutions, and modelling conceptually distinct aspects of the solution as different classes. Hence the focus is on the design and implementation of the classes, and it is important you follow the structure described below.

A book store keeps track of the year in which it purchased each book, the selling price of the book, and when it was sold. Books that are bought are sent to the Purchasing dept, which assigns a selling price to each book. The books automatically assign themselves a unique number (as described below). Books are sold by the Sales dept, which keeps track of each book and the year in which it is sold. However, the records of the sales department are maintained in a haphazard way, and it was difficult for them to figure out their annual revenues, and revenues corresponding to books purchased in a particular year. We have dug up the records for the last few years and now want to reconstruct the revenues of the store for the recent years.

Model this as a Java program with the following classes:

- Store:
  - has a Purchasing and Sales dept. Created when the Store is constructed
  - Maintains the list of all Books ever purchased
- Book:
  - Each time a new Book is generated, it is assigned a unique id by the class itself. the id is of the form yyyy-xxxxxx where yyyy is the year, and xxxxxx is a unique integer corresponding to the sequence number of the purchase in that year. This number is padded in the front with 0's to always be 6 digits. Thus, the first book in the year 2001 would be 2001-000001, the next would be 2001-000002 and so on, and the first book the next year would be 2002-000001 (ie the sequence number is reset every year).
  - The book also keeps track of its selling price - assigned by the Purchasing dept.
- Purchasing:
  - Has a method to add a new book, passing in the year and the purchasing price. It assigns a selling price to the book, which is 1.5 times the cost of the book. (this should be set up so that the factor can be changed easily by the Store)
- Sales:
  - Normally, just records the id of the book and when it was sold. We now need a new method that given the id of the book and the year in which it was sold, keeps track of the total revenue for books that were bought in that year, books bought the previous year, books bought 2 years earlier etc. (It can find the year of purchase of the book from the book id).
  - It has a method printSales that prints out in the following format:
    year1 <revenue from books bought in year1> <revenue from books bought in previous year> <revenue from books bought 2 years back>

year2 <revenue from books bought in year2> <revenue from books bought in previous year> <revenue from books bought 2 years back>
etc

- Test class - contains only main which does the following
    - Creates the Store class
    - reads the input, first for the purchase information. Calls methods on the Purchase class to add a new book with the year and price information. We can assume that the input about purchases are in chronological order
    - reads the input for sales. Calls method on the Sales object passing in the book id and year of sales.
    - When the input is complete, it calls the printSales method of the Sales dept.
- Create any other classes that could be helpful to modularize the code

The input has the following format (from standard input):
The first set of lines have 2 integers per line: the year of purchase and the cost of the book.
An input of 0 0 indicates that the purchase information is complete.
The next set of lines have the year of sale (integer) and a book-id (string)
A line with 0 0 indicates end of input

Notes:

- Explore the Java class ArrayList. ArrayList<Book> will help you maintain a dynamic array of Book objects. ArrayList<Integer> can help maintain a list of integer values
- One mechanism for generating unique ids is to use a **static** data member in the class. (Will be discussed in the next lecture)
- Data members of a class should not be directly accessible by other classes. These should be manipulated only through appropriate methods provided by the class.

Sample input
2017 400
2017 300
2017 200
2018 500
2018 800
2018 100
2018 200
2019 100
2019 400
0 0
2018 2017-000001
2019 2017-000002
2019 2018-000003
2018 2018-000002
2019 2019-000002

2018 2018-000004
0 0

Expected output
2018 1500 600 0
2019 600 150 450