CPS2004 – Object Oriented Programming Assessment

Assignment instructions (read carefully and thoroughly):

- This is an individual assignment and carries 100% of the final CPS2004 grade. Under no circumstances are you allowed to share the design and/or code of your implementation. You may not copy code from internet sources, you will be heavily penalized if you do so! The Department of Computer Science takes a very serious view on plagiarism. For more details refer to plagiarism section of the Faculty of ICT website https://www.um.edu.mt/ict/Plagiarism

- A demonstration of your work may be required by the Board of Examiners.

- While it is strongly recommended that you start working on the tasks as soon as the related material is covered in class, the firm submission deadline is **Monday 14th January 2019**.

- A submission including a soft-copy of the report and all the code (all source, make files, any scripts and instructions required to compile the code), should be uploaded to the VLE by the deadline. All files must be archived into a single file named: SURNAME_NAME_IDCARD.zip with each task located in a toplevel directory in the .zip file named task1, etc. **It is the student's responsibility to ensure that the uploaded zip file and all contents are valid.**

- You are to allocate 50-60 hours to complete this assignment.

- The first page in your assignment report must be the title of your assignment clearly showing your name, surname and study unit code. Reports that are difficult to follow due to low quality in the writing-style/organization/presentation will be penalized.

Assignment specification

Your code will be compiled and run on Linux (Ubuntu).

- C++ code will be compiled using the g++ compiler from the GNU Compiler Collection. This will be run using the command line g++ -Wall -std=c++11 *.cpp.
- Java submissions will be compiled and run using OpenJDK. Please make use of standard libraries only.

As a requirement, add bash scripts *compile.sh* and *run.sh* in each task directory to compile and run your task. In these bash scripts, include any command line arguments. Note that you should not have any absolute paths hardcoded in your programs/bash scripts.

Each task will be evaluated as follows:
40% - Correct application of OO principles including appropriate reasoning and diagrams for your design in the report.
30% - Working solution including use of design patterns, validation, exception handling, and critical evaluation of your solution in the report.
20% - Source code presentation, readability (through comments), organisation in packages/namespaces, coding standards, etc.
10% - Report quality and completeness

You are to complete all the following tasks.

Task 1 – C++

Implement a generic binary search tree[1] (hint: use templates and a "comparable" interface) with functions: *insert*, *find*, *remove*, and *pre-order/in-order/post-order* traversals; each with the appropriate parameters and return values. Make sure that the correct memory management techniques are used.

Also implement animal objects organized under mammals, reptiles, and birds. Each animal should have an attribute length (the size of the animal in centimeters).

Your system should be able to read commands from a textfile specified as a command line argument (if no file is specified, your test file should be used by default). These are the commands you should be able to support:

- `Insert mammal <Name> <Length> <AverageLitterSize>`
- `Insert reptile <Name> <Length> <venomous|non-venomous>`
- `Insert bird <Name> <Length> <can-fly|cannot-fly>`
- `Find <Name>`
- `Remove <Name>`

At the end of the command list, you are to print the tree using inorder traversal. When printing each animal it should be specified if it is a mammal, reptile, or bird.

[1] https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal.htm
https://www.tutorialspoint.com/data_structures_algorithms/binary_search_tree.htm

Task 2 – Java

Implement a system that manages food orders for multiple restaurants (all data is to be kept in memory). Your system should have the following elements:

- Restaurants - Some offer deliveries, take away, or both.
- Available menu items - Linked to a restaurant with prices.
- Orders - Limited to a single restaurant specifying whether it is by delivery or take away.
- OrderList – Able to calculate the price.

Use the linked list you implemented in your tutorial to store the ordered list and the observer design pattern to observe the orders being ordered and maintain an accumulation of total being spent and the restaurant which has the highest revenue.

Your system should be able to read commands from a textfile specified as a command line argument (if no file is specified, your test file should be used by default). These are the commands you should be able to support (hint: use an appropriate data structure to store restaurants such that they are easily retrievable by name):

- `BeginRestaurant <RestaurantName> <delivery|takeaway|both>`
- `Item <Name> <Price>`
- `EndRestaurant`
- `BeginOrderList`
- `BeginOrder <RestaurantName> <delivery|takeaway>`
- `OrderItem <ItemName>`
- `EndOrder`
- `EndOrderList`

At the end of each order list, you are to print the total price. At the end of the file, print the total price (of all order lists), the name of the restaurant with the highest revenue, and the revenue amount of this restaurant.