ECS40 Spring 2018 2018-05-01

Homework 2 due Thu 2018-05-10 at 23:59

Use the handin directory hw2 to submit your work

Part 1: The All or Nothing Restaurant

Welcome to the one-of-a-kind "All-or-Nothing-Restaurant", where if guests are lucky enough to get seated, then they can eat as much as they want. But, there's a catch. The arriving group can only be seated all together at one table. If the current seating arrangement cannot accommodate them, they need to try coming back another night. Each table in the restaurant can only be used once in the entire evening.

Given the restaurant's popularity, there is large queue of groups waiting to be seated everyday. The restaurant however only has 3 types of tables - Large (which can accommodate 10 guests, Medium which can accommodate 7 guests and Small which can accommodate 5 guests). Currently, the restaurant has only 1 Large sized table, 2 Medium sized tables and 3 small sized tables. The owner of the restaurant has hired you to write a program to assign guests to tables.

Description

The input of the program consists of the size of the groups, in the order in which they appear in the queue. The size of the first group to be seated appears first. The table assignment program must first print a list of available tables in the restaurant. It should then read the sizes of each group to be seated and report on attempts to assign the group to a table. Finally, it should print a summary of seat assignments. See the example input and output files for details. If a group size is not valid (i.e. not a positive number) an exception should be thrown and an error message printed before the program proceeds with the next group.

Assignment

The program **`assigntables.cpp`** is provided, together with the header files **`Restaurant.h`** and **`Table.h`** . You should not modify these files. You will implement the classes **`Restaurant`** and **`Table`** in the files **`Restaurant.cpp`** and **`Table.cpp`** respectively. You will create a **`Makefile`** that includes a target **`assigntables`** (that builds the executable) and a target **`clean`** (that removes the executable and all object files). Use the test input files to verify that your program reproduces *exactly* the example output files. Use the Unix diff command to compare your output to the example output.

Specification of the Restaurant class

The **`Restaurant`** class constructor creates the list of tables using dynamic allocation of the (private) pointer array **`tableList`**. The tables should appear in the list in order of decreasing size. The **`Restaurant`** constructor takes three arguments (the number of large, medium and small tables in the restaurant). The **`size`** data member is the total number of tables in the restaurant. The **`Restaurant`** class has a member function **`addGuests`** that tries to assign a group of guests to a table by inspecting the list of tables and by assigning the group to the first table that can accommodate the group, starting at the beginning of the **`tableList`** array. The **`Restaurant`** class also has a member function **`printSummary`** that prints a list of tables with their current and maximum occupancy. Empty tables should not be printed in the summary. See the examples of output files for details on the output format.

Specification of the Table class

The **`Table`** class constructor takes one argument (the size of the table). It has accessor member functions **`maxOccupancy`** and **`currentOccupancy`** returning the maximum and current number of guests respectively. The **`addGuests`** member function attempts to add a group of guests to the table and modifies the occupancy accordingly. It returns **`true`** if the operation is successful and **`false`** otherwise.

All compilations should complete without warnings (use the **`–Wall`** option). Submit your project using:

```
$ handin cs40 hw2 restaurant.tar
```

Part 2: Aircraft Maintenance System

The regular maintenance of aircraft engines is critical to ensure the high level of safety of air travel. Maintenance activities are scheduled for each engine of an aircraft after specific periods of use, which depend on the aircraft and engine model. In this assignment, you are asked to write a program that determines the maintenance schedule of aircraft engines. The program must be able to process different types of aircrafts (having different numbers of engines), requiring maintenance at different intervals. In this assignment, we will only consider two types of aircraft: Airbus A380 (four engines, requiring maintenance every 750 hours) and Boeing 737 (two engines, requiring maintenance every 500 hours).

Description

The input of the program is read from standard input. Each input line consists of a description of an aircraft, including the model (encoded as one character), the aircraft name (or "tail code"), and the number of hours recorded since the last maintenance check for each of its engines. The number of hours of use may differ among the engines of a given aircraft due to the fact that not all engines are subjected to maintenance at the same time.

Assignment

Aircrafts are represented as instances of the classes **`A380`** and **`B737`** that are derived from a base class **`Aircraft`**. The main program **`maintenance.cpp`** is provided, together with the header file **`Aircraft.h`** . Another program **`testAircraft.cpp`** is provided that tests the functionality of the **`Aircraft`** derived classes. You should not modify these files. You will implement the base class **`Aircraft`**, and the derived classes **`A380`** and **`B737`** in the file **`Aircraft.cpp`** . You will create a **`Makefile`** that includes a target **`all`**, which builds the executables **`maintenance`** and **`testAircraft`**. The **`maintenance`** program reads a list of aircraft descriptions from standard input and prints a description of the aircraft and its maintenance schedule. See the example input files and corresponding output files for details. Note that if an engine's maintenance is past due, a special output message is printed instead of a time. If an unknown aircraft code is used, the program prints an error message and exits (see example files).

Example: the input line:

```
A VH-OQF 630 550 470 690
```

describes an Airbus A380 named VH-OQF whose four engines have 630, 550, 470 and 690 hours of use since their last maintenance. Given this input, the maintenance program will print the following output:

```
Aircraft: VH-OQF type: Airbus A380 has 4 engines
engine 1: 630 hours
engine 2: 550 hours
engine 3: 470 hours
engine 4: 690 hours
Maintenance schedule for VH-OQF
engine 1: maintenance due in 120 hours
engine 2: maintenance due in 200 hours
engine 3: maintenance due in 280 hours
engine 4: maintenance due in 60 hours
```

If the input contains multiple lines, the program should process them sequentially until the end of file is reached in input.

Specification of the Aircraft classes

The **`Aircraft`** base class is an abstract class from which the classes **`A380`** and **`B737`** are derived. The base constructor takes two arguments: the number of engines and the aircraft name. In addition, the following member functions must be defined:

`virtual const std::string type(void) const`
A pure virtual function returning a string (**`A380`** or **`B737`**).

`virtual const int maxHours(void) const`
A pure virtual function returning the maximum duration of use of an engine before maintenance is due.

`const std::string name(void) const`
A member function returning the name of the aircraft.

`int numEngines(void) const`
A member function returning the number of engines of the aircraft.

`void setHours(int i, int h)`
A member function used to set the current number of hours of use for engine **`i`** .

`void print(void) const`
A member function that prints the description of the aircraft on standard output (see first part of the above example).

`void printSchedule(void) const`
A member function that prints the maintenance schedule of the aircraft on standard output (see above example).

`static Aircraft* makeAircraft(char ch, std::string name_str)`
A static factory function that creates an **`Aircraft`** object of the appropriate type (determined by the character **`ch`**) and returns a pointer to it. If **`ch`** is '**`A`**', an **`A380`** must be created. If **`ch`** is '**`B`**', a **`B737`** must be created.

All compilations should complete without warnings (use the **`–Wall`** option). Submit your project using:

```
$ handin cs40 hw2 aircraft.tar
```