Assignment 8 - Railroad Cars

Due Apr 25 by 11:59pm **Points** 20 **Submitting** a file upload **File Types** h and cpp **Available** until Apr 28 at 12:01am

This assignment was locked Apr 28 at 12:01am.

ST.CHARLES COMMUNITY COLLEGE

CPT-182 - Programming in C++

Programming Assignment - Railroad Cars (20 Points)

(Number in Question Bank: Assignment **8.1**)

Program Overview

In this assignment, you are going to write four related classes (plus the **`main()`** program), **`Railroad_Car`**, **`Tank_Car`**, **`Box_Car`**, and **`Refrigerator_Car`**. Using class inheritance and polymorphism, you are going to write a well-organized object-oriented C++ program.

The `Railroad_Car` Class

- The **`Railroad_Car`** class is a pure abstract class.
- The **`Railroad_Car`** class has the following data field:

Variable Name	Data Type	Explanation
`length`	`double`	Stores the length of the railroad car.

- The **`Railroad_Car`** class has the following class-member functions:

Function Name	Function Argument	Function Behavior	Return Value
`get_length()`	**No argument**	Getter of `length`	The value of `length`
`set_length()`	`length`: `double`	Setter of `length`	`void`
`volume()`	**No argument**	Pure `virtual` function	Returns a `double`.

The `Tank_Car` Class

- The **`Tank_Car`** class is a derived class of the **`Railroad_Car`** class, which has a cylinder shape.
- The **`Tank_Car`** class has the following data field:

Variable Name	Data Type	Explanation
`radius`	`double`	• The end of a cylinder is a circle. • **`radius`** stores the radius of the circle.

- The **`Tank_Car`** class has the following class-member functions:

Function Name	Function Argument	Function Behavior	Return Value
`Tank_Car()`	**No argument**	Default constructor	**No return type**

`Tank_Car()`	`length`: `double`
	`radius`: `double`
		Constructor that initializes class data fields with the values passed in	**No return type**

`get_radius()`	**No argument**	Getter of `radius`	The value of `radius`
`set_radius()`	`radius`: `double`	Setter of `radius`	`void`
`volume()`	**No argument**	• Overrides the function in the base class. • Calculates the volume of the tank car.	The calculated volume

- Please make sure that you use the correct math formula to calculate the volume of a cylinder.

The `Box_Car` Class

- The **`Box_Car`** class is a derived class of the **`Railroad_Car`** class, which has a cuboid shape.
- The **`Box_Car`** class has the following data fields:

Variable Name	Data Type	Explanation
`width`	`double`	Stores the width of the box car.
`height`	`double`	Stores the height of the box car.

- The **`Box_Car`** class has the following class-member functions:

Function Name	Function Argument	Function Behavior	Return Value
`Box_Car()`	**No argument**	Default constructor	**No return type**

`Box_Car()`	`length`: `double`
	`width`: `double`
	`height`: `double`
		Constructor that initializes class data fields with the values passed in	**No return type**

`get_width()`	**No argument**	Getter of **`width`**	The value of **`width`**
`get_height()`	**No argument**	Getter of **`height`**	The value of **`height`**
`set_width()`	`width`: `double`	Setter of **`width`**	`void`
`set_height()`	`height`: `double`	Setter of **`height`**	`void`
`volume()`	**No argument**	• Overrides the function in the base class. • Calculates the volume of the box car.	The calculated volume

- Please make sure that you use the correct math formula to calculate the volume of a cuboid.

The `Refrigerator_Car` Class

- The **`Refrigerator_Car`** class is a derived class of the **`Box_Car`** class.
- The **`Refrigerator_Car`** class has the following data field:

Variable Name	Data Type	Explanation
`temperature`	`int`	Stores the internal temperature of the refrigerator car.

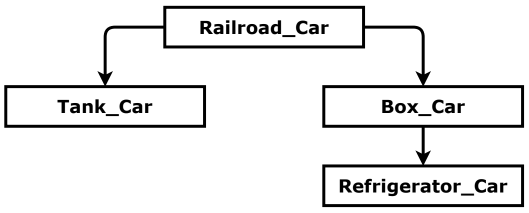
- The **`Refrigerator_Car`** class has the following class-member functions:

Function Name	Function Argument	Function Behavior	Return Value
`Refrigerator_Car()`	**No argument**	Default constructor	**No return type**

`Refrigerator_Car()`	`length`: `double`
	`width`: `double`
	`height`: `double`
	`temperature`: `int`
		Constructor that initializes class data fields with the values passed in	**No return type**

`get_temperature()`	**No argument**	Getter of **`temperature`**	The value of **`temperature`**
`set_temperature()`	`temperature`: `int`	Setter of **`temperature`**	`void`

Class Dependency Graph



The Input File

- The input file is a **plain text file** (filename: **`cars.txt`**).
- The first data field in each row of the input file is the type of railroad car (**`"Tank"`**, **`"Box"`**, or **`"Refrigerator"`**).
- If the first data field is **`"Tank"`**, then there are two more data fields followed the same row, which are the **length** and **radius** of the tank car (in that order).
- If the first data field is **`"Box"`**, then there are three more data fields followed the same row, which are the **length**, **width**, and **height** of the box car (in that order).
- If the first data field is **`"Refrigerator"`**, then there are four more data fields in the same row, which are the **length**, **width**, **height**, and **temperature** of the refrigerator car (in that order).
- You **cannot** assume (or guess) the number of railroad cars in the input file. In other words, no matter how many railroad cars are stored in the input file, your program should correctly process all of them.

- There are may be empty lines at the beginning, in the middle, and/or at the end of the input file. Your program should smartly skip those empty lines.
- Please refer to the **sample input file** to better understand the input file format.

The Output File

- The output file is a **plain text file** (filename: **`volumes.txt`**).
- After you read in a railroad car (either a tank car, box car, or refrigerator car) from the input file, your program calculates the volume of the railroad car and writes the volume to the output file.
- Each volume is a separate line in the output file.
- For each volume in the output file, please keep exactly **2** decimal places.
- Please refer to the **sample output file** to better understand the output file format.

The `main()` Program

- Before starting to read the input file, you need to create a **pointer** to **`Railroad_Car`** and initialize it to **`NULL`**. This should be the only **`Railroad_Car`** pointer you create in your program.
- During the reading of the input file, if the current railroad car you are reading in is a tank car, then you should instantiate the **`Railroad_Car`** pointer you created as a **`new`** **`Tank_Car`**. Same philosophy applies to **`Box_Car`** and **`Refrigerator_Car`** as well. This approach is called **polymorphism**.
- After you instantiate the **`Railroad_Car`** pointer as a **`new`** **`Tank_Car`**, **`Box_Car`**, or **`Refrigerator_Car`**, at the end of the current iteration, you call function **`volume()`** to calculate the volume of the railroad car. Based on the theory of overriding, the compiler will always find the correct function definition at runtime to correctly calculate the volume.
- Please note that the **`volume()`** function should only be called at the end of the iteration (one in each iteration), **not** ~~one in each **`if`** branch~~.
- Do **not** forget to delete the pointer at the end of each iteration.

Sample Input and Output Files (Click to Download)

Sample Input File 1 (https://drive.google.com/uc?export=download&id=1iyXaNRwJhyk30WhC9hcVSZvcNTTbbJTg) **Sample Input File 2 (**
Sample Output File 1 (https://drive.google.com/uc?export=download&id=1j-UFHPQjNkxcjwGhMvU_A9yNROpkl7Ex)Sample Output File 2

Assignment Submission and Grading (Please Read)

- Please upload all your **`.h`** (if any) and **`.cpp`** files (**not** the ~~entire Microsoft Visual Studio project folder~~) on Canvas.
- Before the assignment deadline, you can submit your work unlimited times. However, only your latest submission will be graded.
- At least **20%** of your code should be **comments**. All variable, function (if any), and class (if any) names should "make good sense". You should let the grader put **least effort** to understand your code. Grader will **take off points**, even if your program passes all test cases, if he/she has to put extra **unnecessary** effort to understand your code.
- Please **save a backup copy** of all your work in your computer hard drive.
- Your program will be graded (tested) using another valid input file (still named **`cars.txt`**) to check whether it can generate the expected (correct) output file (with correct format and correct output values in it). As long as the input file is valid, your program should generate a correct output file. In other words, your program should work for **any** valid input file, **not** just the sample input files provided in the assignment instructions.
- In this class, you can assume that the input file (input data) is always **valid** and **has correct format**. You do **not** need to deal with ~~invalid input~~ or ~~error handling~~.
- Your work will be graded after the assignment deadline. All students will receive their assignment grades at (almost) the same time.