Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Computer Programming

Organizer

author: Maciej Krzyżowski
instructor: dr inż. Michał Marczyk
year: 2020/2021

Table of contents

1 Project's topic

A program should create lists (e.g. home, work, shopping) and enable the user to add tasks to the given list. The user should also remove the task from the list to mark the task as done.

2 User Interface

The program will display available lists with a number of tasks within given lists and prompt the user to input list's number to enter specific list. The user will also be given an option to exit the program. Upon choosing a list, its content will be shown with a short manual below containing commands that can be used. The tasks in the lists will be numbered and will have checkboxes next to them to display the task's status.

The user will be able to add, update or remove tasks by their index or remove all or finished tasks. Besides, there will also be an option to print the list again, print the list to file, rename the list, show the manual or change the current list.

3 Classes and Data Structures

3.1 Classes

The program contains four classes in total, two base and two derived classes, where Product inherits from class Task and ShoppingList inherits from class Organizer. Class Task represents a task which is stored in a list. It has a name, a description and a status. Class Product inherits the name and status from class Task but instead of description it has a cost.

Class Organizer contains a vector of objects of type Task, which acts as a list, and a name of the list. Class ShoppingList is an inherited class from the Organizer class with its vector storing objects of type Product.

Inheritance diagrams and methods' descriptions are in the Doxygen part of the report.

3.2 Data Structures

The data structure being used in the program is the vector, which stores objects being either tasks or products, depending of list's type. Using vectors allows having a data structure with variable size and easy adding, modifying and removing elements.

4 Algorithms

The program uses simple and not complicated algorithms. To make the program faster, it takes advantage of std::string_view instead of std::string in cases, where the string of text wouldn't be modified (e.g. when setting a new name for a task). This is because the std::string_view doesn't allocate memory as std::string does.

Exemplary method, which removes finished tasks:

```
void Organizer::removeFinished()
{
    int i = 0;
    for (auto x : this->list)
    {
        if (x.status)
        {
            this->list.erase(this->list.begin() + i);
        }
        i++;
    }
}
```

Exemplary method, which uses std::string_view:

```
void Product::set(std::string_view _name, double _cost)
{
    this->name = _name;
    this->cost = _cost;
    this->status = false;
}
```

The rest of the methods as well with overloaded functions and operators are also in the Doxygen part of the report. The program doesn't use global variables.

5 Program Overview

The program begins by prompting the user to choose a list from three available options. The user is then shown, which commands can be used in the program. The possible commands are:

- adding a new task or product
- updating its status
- removing it from the list
- printing the list either on the screen or to the file
- renaming the list
- printing list of available commands
- quitting the list

The program ends when while choosing a list the user inputs incorrect choice, which is told to the user at the beginning of the program.

The user is given the flexibility to use the program in many ways and in quick pace, with short and intuitive commands. The manual is easily accessible from within the program and gives detailed description of how the program can be used.

6 External Specification

The program has an option to output a chosen list to a text file. The text file has a name corresponding to the given list. The list in the file looks the same as in the console.

7 Testing

The program has been tested with various types of inputs. Incorrect ones are detected and an error message is printed. The program prevents the user from inputting a word where a number is required, which would cause the program to fail, and has protections needed with output file operations.

8 Conclusion

The program takes advantage of basic characteristics of object oriented programming, such as inheritance or polymorphism, as well as overloading. It also uses uncomplicated algorithms and focuses on fast execution of the code.

github link: `https://github.com/maciekrz/organizer`

Organizer

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

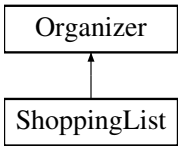
Class Documentation

3.1 Organizer Class Reference

Base class used as a list of tasks to do.

`#include <organizer.h>`

Inheritance diagram for Organizer:



Public Member Functions

- virtual void removeAll ()

  A method for removing all tasks.
- virtual int countTasks ()

  A method for calculating number of tasks in the list.
- std::string getName ()

  A method for getting a name of the list.
- void setName ()

  A method that sets a new name for the list.
- void interactiveMode ()

  A method that calls other methods depending on user's input.
- Organizer (std::string_view _listName)

  One-argument constructor.
- Organizer ()

  Default constructor.

Protected Member Functions

- virtual void printAll ()

  A method for printing the list.
- virtual void printListToFile ()

  A method for printing the list to file.
- virtual void addTask ()

  A method for pushing a new task on the list.
- virtual void updateTask ()

  A method for changing task's status.
- virtual void removeTask ()

  A method for removing a specific task from the list.
- virtual void removeFinished ()

  A method for removing finished tasks.
- void printHelp ()

  A method that prints available options for the user.

Protected Attributes

- std::string **listName**

Private Attributes

- std::vector< Task > **list**
- Task **_task**

3.1.1 Detailed Description

Base class used as a list of tasks to do.

Parameters

list	A vector of objects of type Task
listName	Name of the list
_task	An object that is being modified with its copy being pushed on the list

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Organizer() [1/2]

```
Organizer::Organizer (
            std::string_view _listName )
```

One-argument constructor.

Parameters

_listName	A name for the list

3.1.2.2 Organizer() [2/2]

```
Organizer::Organizer ( )
```

Default constructor.

3.1.3 Member Function Documentation

3.1.3.1 printAll()

```
void Organizer::printAll ( )  [protected], [virtual]
```

A method for printing the list.

Reimplemented in ShoppingList.

3.1.3.2 printListToFile()

```
void Organizer::printListToFile ( )  [protected], [virtual]
```

A method for printing the list to file.

Reimplemented in ShoppingList.

3.1.3.3 addTask()

```
void Organizer::addTask ( )  [protected], [virtual]
```

A method for pushing a new task on the list.

Reimplemented in ShoppingList.

3.1.3.4 updateTask()

```
void Organizer::updateTask ( )  [protected], [virtual]
```

A method for changing task's status.

Reimplemented in ShoppingList.

3.1.3.5 removeTask()

```
void Organizer::removeTask ( )  [protected], [virtual]
```

A method for removing a specific task from the list.

Reimplemented in ShoppingList.

3.1.3.6 removeFinished()

```
void Organizer::removeFinished ( )  [protected], [virtual]
```

A method for removing finished tasks.

Reimplemented in ShoppingList.

3.1.3.7 printHelp()

```
void Organizer::printHelp ( )  [protected]
```

A method that prints available options for the user.

3.1.3.8 removeAll()

```
void Organizer::removeAll ( )  [virtual]
```

A method for removing all tasks.

Reimplemented in ShoppingList.

3.1.3.9 countTasks()

```
int Organizer::countTasks ( )  [virtual]
```

A method for calculating number of tasks in the list.

Returns

Number of tasks

Reimplemented in ShoppingList.

3.1.3.10 getName()

```
std::string Organizer::getName ( )
```

A method for getting a name of the list.

Returns

Name of the list

3.1.3.11 setName()

```
void Organizer::setName ( )
```

A method that sets a new name for the list.

3.1.3.12 interactiveMode()

```
void Organizer::interactiveMode ( )
```

A method that calls other methods depending on user's input.

The documentation for this class was generated from the following files:

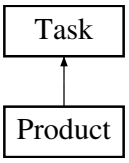
- organizer.h
- organizer.cpp

3.2 Product Class Reference

Derived class from class Task that becomes a product for purchase. Methods and overloaded operator work the same as in base class but are slighty changed.

`#include <task.h>`

Inheritance diagram for Product:



Public Member Functions

- void **print** ()
- void **set** (std::string_view _name, double _cost, bool _status)
- void **set** (std::string_view _name, double _cost)
- double **getCost** ()

Private Attributes

- double **cost**

Friends

- std::ostream & **operator**<< (std::ostream &os, Product const &prod)

Additional Inherited Members

3.2.1 Detailed Description

Derived class from class Task that becomes a product for purchase. Methods and overloaded operator work the same as in base class but are slighty changed.

Parameters

cost	A cost of the product

The documentation for this class was generated from the following files:

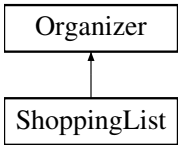
- task.h
- task.cpp

3.3 ShoppingList Class Reference

Derived class from class Organizer that is a shopping list. Instead of vector of objects of type Task, it holds objects of type Product. Its methods work the same as in class Organizer but were changed to work with Products and not Tasks.

`#include <organizer.h>`

Inheritance diagram for ShoppingList:



Public Member Functions

- **ShoppingList** (std::string_view _listName)
- void removeAll ()

  A method for removing all tasks.
- int countTasks ()

  A method for calculating number of tasks in the list.

Private Member Functions

- void printListToFile ()

  A method for printing the list to file.
- void addTask ()

  A method for pushing a new task on the list.
- void printAll ()

  A method for printing the list.
- void removeTask ()

  A method for removing a specific task from the list.
- void removeFinished ()

  A method for removing finished tasks.
- void updateTask ()

  A method for changing task's status.

Private Attributes

- Product **_product**
- std::vector< Product > **list**

Additional Inherited Members

3.3.1 Detailed Description

Derived class from class Organizer that is a shopping list. Instead of vector of objects of type Task, it holds objects of type Product. Its methods work the same as in class Organizer but were changed to work with Products and not Tasks.

Parameters

list	A vector of objects of type Product
_product	An object that is being modified with its copy being pushed on the list

3.3.2 Member Function Documentation

3.3.2.1 printListToFile()

```
void ShoppingList::printListToFile ( )  [private], [virtual]
```

A method for printing the list to file.

Reimplemented from Organizer.

3.3.2.2 addTask()

```
void ShoppingList::addTask ( )  [private], [virtual]
```

A method for pushing a new task on the list.

Reimplemented from Organizer.

3.3.2.3 printAll()

```
void ShoppingList::printAll ( )  [private], [virtual]
```

A method for printing the list.

Reimplemented from Organizer.

3.3.2.4 removeTask()

```
void ShoppingList::removeTask ( )  [private], [virtual]
```

A method for removing a specific task from the list.

Reimplemented from Organizer.

3.3.2.5 removeFinished()

```
void ShoppingList::removeFinished ( )  [private], [virtual]
```

A method for removing finished tasks.

Reimplemented from Organizer.

3.3.2.6 updateTask()

```
void ShoppingList::updateTask ( )  [private], [virtual]
```

A method for changing task's status.

Reimplemented from Organizer.

3.3.2.7 removeAll()

```
void ShoppingList::removeAll ( )  [virtual]
```

A method for removing all tasks.

Reimplemented from Organizer.

3.3.2.8 countTasks()

```
int ShoppingList::countTasks ( )  [virtual]
```

A method for calculating number of tasks in the list.

Returns

Number of tasks

Reimplemented from Organizer.

The documentation for this class was generated from the following files:

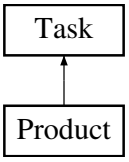
- organizer.h
- organizer.cpp

3.4 Task Class Reference

Base class that represents a task.

`#include <task.h>`

Inheritance diagram for Task:



Public Member Functions

- virtual void **print** ()
- virtual void set (std::string_view _name, std::string_view _desc, bool _status)

  A three-argument method that changes object's values.
- virtual void set (std::string_view _name, std::string_view _desc)

  An overloaded two-argument method that changes object's values. Works the same as its original version, but sets status value as false by default.

Public Attributes

- std::string **name**
- bool **status**

Private Attributes

- std::string **desc**

Friends

- std::ostream & operator<< (std::ostream &os, Task const &task)

  An overloaded operator for outputting object's values.

3.4.1 Detailed Description

Base class that represents a task.

Parameters

desc	A description of the task
name	A name of the task
status	A status of the task represented by a boolean value

3.4.2 Member Function Documentation

3.4.2.1 set() [1/2]

```
void Task::set (
            std::string_view _name,
            std::string_view _desc,
            bool _status )  [virtual]
```

A three-argument method that changes object's values.

Parameters

_name	The new name for the task
_desc	The new description for the task
_status	The new status of the task

3.4.2.2 set() [2/2]

```
void Task::set (
            std::string_view _name,
            std::string_view _desc )  [virtual]
```

An overloaded two-argument method that changes object's values. Works the same as its original version, but sets status value as false by default.

3.4.3 Friends And Related Function Documentation

3.4.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            Task const & task )  [friend]
```

An overloaded operator for outputting object's values.

Returns

A string that is ready for being put either on screen or to file

The documentation for this class was generated from the following files:

- task.h
- task.cpp

Index