Mechatronic Engineering

Object Oriented Programing and Software Engineering
Laboratory instruction 12
C++ introduction

AGH Kraków, 2020

Materials created for educational purposes.
Dedicated for students attending Software Engineering course.
Author would apreaciate any feedback regarding errors of any kind found in the instruction script.
Please report those to the following email address: danielt@agh.edu.pl

Contents

1 Object oriented design

As you may have noticed through the years, there are many approaches to programming. Depending on an used language, programer should choose the right one to meet the key points of the approached problem. If it comes to creating high complexity programs object oriented projecting and programing comes in handy. It is because the use of objects and classes allows you to create software models corresponding to the reality.

The following section shows also the steps of evolution in programing methods.

1.1 Programing techniques

In this paragraph I'll briefly explain basic programing techniques, their principles and limitations.

1.1.1 linear programing

The principal of linear programing is to write the code so that individual instructions are implemented one after another. Programs writen is such way usually use global variables. Frequent use of *go to* instructions makes the program less readable to other programmers.

1.1.2 Procedural programing

This technique provided programers with a very efective tool: **functions**. Implementing functions into programing allowed the programer to use additional variables called local variables. The general idea of procedural programing is to divide the target problem into simple solvable tasks. Tasks are solved by the created functions.

1.1.3 Programming with data hiding

This is a tehnique that allows programmer to use data modules. General idea is to pack different data into one module. Such data can be treated as a group. It comes down to the use of structures (in C++ it can be also achieved by using classes to store only public data). The main difference to previous method is that structures can be used to send all the necesary data at once to a function.

1.1.4 Object based design

This tehnique adds member functions (methods) to the previously grouped data. The main principle is that the data structuralized into a module (object) doesn't know **how** to do something, but it knows **what** to do. Member functions are responsible for knowing how to do something.

1.1.5 Object oriented programing

object oriented programming has some features that distinguish them from other programming techniques:

- Reusability – enables the secondary use of previously written code. It is possible by implementing of **inheritance** into programming
- Extensibility – facilitates subsequent modifications of the program.
- Real world modeling – Object oriented programing allows to transfer of real life models into software representations.

1.2 Designing an object oriented program

Designing an object oriented program requires to follow a plan. You could compare it to building a house. It is possible, to build a house ad hoc, it will even fullfill the basic functionality of a house, but if you plan your house earlier, building it will be easier. When it comes to comlicated systems it is far more efficient to plan your work before you start programming.

Planing steps:

1. Problem research – Fragment of reality that the program is supposed to address is called a system. The main idea is to recognise the propperties of a considered system before you start coding. In this step you get to know what should your program do.
2. Program designing – In this step you need to figure out how to realize tasks that your program should do.
3. Implementation – After you created your theoretical plan it is time to do the codding
4. Testing – In this steps all the errors and misbehavings of our program should be found. This step doesn't finish the planing process. It often leads to some corrections in the planing and/or coding.

Table 1: System behaviors scenarios [1]

Who	Action	Whom	Result
Player	Throws	Dice	Random number from 1 - 6
...	...	...	...
...	...	...	...

1.2.1 Designing steps

1. Identification of system behaviors
2. Identification of objects in the system
3. Objects classification
4. determination of the mutual dependencies of object classes
5. Assemling the model

1.2.1.1 Identification of system behaviors

This step is responsible for systematization of knowledge about the problem. The whole knoledge about the system should be written in the form of scenarios. It's not obligatory to arrange all the scenarios in a chronological order, but it will save you a lot of work when determining the sequence of objects and theit life cycles.

Scenarios should be prepared in a simplest possible form, for example in a table (tab. 1)

1.2.1.2 Identification of objects in the system

In this step, you need to find objects that work in the system that you want to program. Programmer needs to find objects that naturally reflect real system behavior. It can be easily obtained by going through the table 1 and pick columns **Who** and **Whom**. Those columns contain information about names of the classes for the objects of our system.

Modeling cards are a very usefull tool when it comes to object identification in the system. Modeling cards store information about objects properties.

Note, that in this step we are only identifying objects and their dependencies, implementation details ar not yet important.

Table 2: Modeling cards for object identification in the system

Class	(class name)		
Duties:			Asociates:
....			....
	Visable propperties:	....	

Table 3: Example of modeling cards for object identification in the system

Class	Dishwasher		
Duties:			Asociates:
...			...
spining sprinklers			motors
drying			heater
water pumping			water pump
...			...
	Visable propperties:	ongoing program	
		water level	
		humidity level	
		...	

1.2.1.3 Objects classification

The next step is to analyze the previously developed classes and objects. The next task of the programmer is to check and find cinnections (inheritances) between the designed classes and determine their hierarchy. This step is crucial to see if there is a chance for reusability of the designed set of classes and objects.

Firstly a programer should proceed with prioritization of designed classes – it is crucial to properly define abstract classes to reuse it by other classes of the system. One method to propperly build inheritances into designed program is to declare a statement: **class A object is a special type of a class B object**. For example: a barn is a special type of a building. If its true it means that class A is a derived class to class B. To check if there are classes including objects of other classes a statement should be declared: **The object of the A class consists, among others, of an object of the B class**. For example: a barn consists, among others, of a gate.

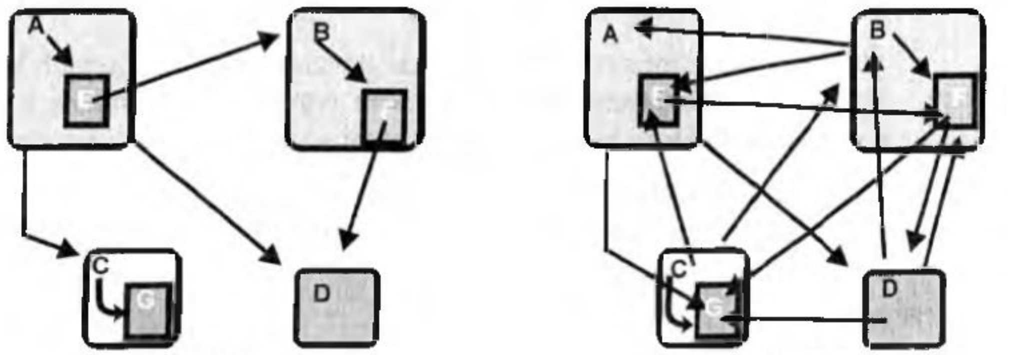
1.2.1.4 Mutual dependencies of classes

In this step is it necessary to decide the way your classes corelate to each other. There are two tools to help a programer to achive that. First one

Table 4: Example of a list of relations for a dishwasher system

Object	Dependency	Whom?
previous object	...	...
	forces to open	water valve
dishwasher programmer	forces to shutdown	water valve
	checks if cover is locked	cover sensor
	...	...
next object	...	...

Figure 1: Example of a graph of object cooperations.On the left a propperly designed system. On the right an overcomplicated system [1].



need to make a list of the relations between the objects and secondly draws a graph of object cooperations. The list can be easily prepared with the use of scenarios (tab. 1) and modeling cards (tab. 3). Example of a list of relations in table 4

Graph of object cooperations is a tool that clearly presents comunication lines between objects. Each rectangle on a graph is an object and lines represent dependecies between objects. Another useful thing that can be clearly seen by looking at the graph is the comlpicicity of designed sytem. There are many aproaches, but one object shouldnt directly cooperate with more than three objects (fig. 1).

Graph can also be a usefool tool to easily divide designed system into **subsystems**. Subsystem is a group of classes dealing with the fulfillment of one, very generally understood role.

1.2.1.5 Determining the sequence of objects and theit life cycles

After splitting the system into subsystems, you can go to the next stage. It is the assembly of the model. To this end, the scenario of our system should be used (tabela 1). The next step is to determine the sequence of actions of the objects of each class. To do this, you can use the following wording: *As soon as object A receives a signal to do it - it issues the following command to object B and ...*
The result of this step will be useful when defining the bodies of individual member functions of a given class.

1.3 Implementation

Implementation is to create a class definition. In this step, you should first develop: visible class properties - constituting member data and class behaviors - which will be treated as member functions. Methods should be created as declarations first. Definitions should be made when the whole system is sufficiently designed and all necessary elements are declared.

2 Task

Go through your program built for puproses of previous laboratories and check if u have propperly build it to meet the object oriented standard. Correct your program if its necessary.

Bibliography

[1] J. Grebosz, Symfonia C++ standard ISO, 2014.