Søren Hansen <sha@ase.au.dk>
V1.0

Message System

Introduction

Events and message queues present us ample room to create asynchronous systems that effectively are *reactionary* by nature. For numerous problems this approach presents an ideal solution, however when creating these systems *inter* knowledge of message queues and events are needed. In this lecture we will inspect two different design approaches that try and lower the coupling between *sender* and *receiver*. In the first design the *sender* knows who it wants to send it's event to, whereas in the second design it *does not care*. Secondly, in order to design/implement these two solutions several design patterns can be used. In this particular scenario we will look at 3 different design patterns. Last but not least, the second design

Content and reflection

Themes

- Message Designs
  - Specific receiver design[1]
  - Broadcasting design[1]
- GOF Design patterns
  - Singleton pattern[2][1]
  - Observer pattern[3][1]
  - Mediator pattern[4][1]
- Fundamental variable info
  - Static members in C++[5]

Do note that the two designs described in the slides can ONLY be found IN SAID slides - thus the slides are extremely important!

Questions

- Designs
  - Specific receiver
    - Which classes are involved and how
    - What is it that a *sender* needs to know to send an event to a *receiver*
    - In what way does this design reduce coupling
    - Which design patterns take part
  - Broadcast - receiver is *irrelevant*
    - Which classes are involved and how
    - What is it that a *sender* needs to know to send an event to a *receiver*
    - Which design patterns take part
    - *LOCAL* and *GLOBAL* IDs - whats this - *VERY IMPORTANT* to understand these two concepts

- Patterns
  - Singleton
    - What is the concept behind it
    - From an implementation point of view, how is this achieved
  - Observer
    - What is the concept behind it
    - How do the classes involved interact
    - What is the difference between *observer* & *publisher/subscriber* (last part requires a google search)
  - Mediator
    - What is the concept behind it
    - How do the classes involved interact conceptually

Material

Slides

[1] S. Hansen, *A message system*, Slides - see course repos.

Local repository

[2] R. J. Erich Gamma Richard Helm and J. Vlissides, *Gof singleton pattern*, Chapter, From the GOF book, found in https://redmine-server.ase.au.dk/courses/projects/i3isu/repository, 1994.

[3] ——, *Gof observer pattern*, Chapter, From the GOF book, found in https://redmine-server.ase.au.dk/courses/projects/i3isu/repository, 1994.

[4] ——, *Gof mediator pattern*, Chapter, From the GOF book, found in https://redmine-server.ase.au.dk/courses/projects/i3isu/repository, 1994.

Online

[5] Alex. (). Static member variables, [Online]. Available: `http://www.learncpp.com/cpp-tutorial/811-static-member-variables`.