Exam #3

Monday, April 29

5:00 - 7:00 pm

closed books and notes

updated coverage is posted!!

comprehensive exam

sample exam #3 is posted

COVERAGE FOR EXAM #3 - updated

CS 586; Spring 2024

Exam #3 will be held on **Monday, April 29, 2024,** between **5:00-7:00 p.m.**
Location: 104 Stuart Building
The exam is a **CLOSED books and notes** exam.

Coverage for the exam:

- OO design patterns: item description, whole-part, observer, state, proxy, adapter, strategy, and abstract factory patterns. [Textbook: Sections 3.1, 3.2; Section 3.4 (pp. 263-275); Section 3.6 (pp.339-343); Handout #1, class notes]
- Interactive systems. Model-View-Controller architectural pattern [Textbook: Section 2.4, pp. 123-143]
- Client-Server Architecture
  - Client-Dispatcher-Server [Section 3.6: pp. 323-337]
  - Client-Broker-Server Architecture [Textbook: Section 2.3; pp. 99-122]
- Layered architecture [Textbook: Section 2.2; pp. 31-51]
- Pipe and Filter Architecture [Textbook: Section 2.2; pp. 53-70]
- ~~Adaptable Systems:~~
  - ~~Micro-kernel architectural pattern [Textbook: Section 2.5, pp. 169-192]~~
- Fault-tolerant architecture [Handout #2]
  - N-version architecture
  - Recovery-Block architecture
  - N-Self Checking architecture
- Repository architecture [Textbook: Section 2.2; pp. 71-95]

Sources:

- Textbook: F. Buschmann, et. al., Pattern-oriented software architecture, vol. I, John Wiley & Sons.
- Class notes
- Handouts

Project Part #2.

Deadline: Thursday, April 25.

PART #2: PROJECT DESIGN, IMPLEMENTATION, and REPORT

CS 586; Spring 2024

Final Project Deadline: **Thursday, April 25, 2024**
Late submissions: 50% off
After **April 30** the final project will not be accepted.

Submission: The project must be submitted on Blackboard. The hardcopy submissions will not be accepted.

This is an **individual** project, not a team project. Identical or similar submissions will be penalized.

DESIGN and IMPLEMENTATION

The goal of the second part of the project is to design two *Gas Pump (GP)* components using the Model-Driven Architecture (MDA) and then implement these *GP* components based on this design using the OO programming language. This OO-oriented design should be based on the MDA-EFSM (for both *GP* components) that was identified in the first part of the project. You may use your own MDA-EFSM (assuming that it was correct) or you can use the posted sample MDA-EFSM. In your design, you **MUST** use the following OO design patterns:

- state pattern
- strategy pattern
- abstract factory pattern

In the design, you need to provide the class diagram, in which the coupling between components should be minimized and the cohesion of components should be maximized (components with high cohesion and low coupling between components). In addition, a sequence diagram should be provided as described on the next page (Section 4 of the report).

After the design is completed, you need to implement the *GP* components based on your design using the OO programming language. In addition, the driver for the project to execute and test the correctness of the design and its implementation for the *GP* components must be implemented.

The Report and Deliverables

I: REPORT

The report **must** be submitted as one PDF-file (otherwise, a **10% penalty will be applied).**

6

1. MDA-EFSM model for the *GP* components
   a. A list of meta events for the MDA-EFSM
   b. A list of meta actions for the MDA-EFSM with their descriptions
   c. A state diagram of the MDA-EFSM
   d. Pseudo-code of all operations of Input Processors of Gas Pump*:* *GP-1* and *GP-2*

may use the posted version.

0

2. Class diagram(s) of the MDA of the *GP* components. In your design, you **MUST** use the following OO design patterns:
   a. State pattern
   b. Strategy pattern
   c. Abstract factory pattern

one class diagram on several pages!!!

0

3. For each class in the class diagram(s), you should:
   a. Describe the purpose of the class, i.e., responsibilities.
   b. Describe the responsibility of each operation supported by each class.

6

4. Dynamics. Provide two sequence diagrams for two Scenarios:
   a. Scenario-I should show how one liter of gas is disposed in the Gas Pump *GP-1* component, i.e., the following sequence of operations is issued:
   Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(), StopPump()
   b. Scenario-II should show how one gallon of Premium gas is disposed in the Gas Pump *GP-2* component, i.e., the following sequence of operations is issued:
   Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt()

II: Well-documented (commented) source code

In the source-code you should indicate/highlight which parts of the source code are responsible for the implementation of the three required design patterns (**if this is not indicated in the source code, 20 points will be deducted**):

- state pattern
- strategy pattern
- abstract factory pattern.

The source-code must be submitted on the Blackboard. Note that the source code may be compiled during the grading and then executed. If the source-code is not provided, **15 POINTS** will be deducted.

III: Project executables

The project executable(s) of the *GP* components with detailed instructions explaining the execution of the program must be prepared and made available for grading. The project executable should be submitted on Blackboard. If the executable is not provided (or not easily available), **20 POINTS** will be automatically deducted from the project grade.

CS586 Project
Spring 2024

Sample Test Cases:

Gas Pump GP-1
Test #1
Activate(4), Start(), PayCredit(), Approved(), StartPump(), Pump(), StopPump()
Expected result at the last operation: Total=$4

Test #2
Activate(5), Start(), PayCredit(), Approved(), Activate(8), StartPump(), Pump(), Pump(), StopPump()
Expected result at the last operation: Total=$10

Test #3
Activate(5), Start(), PayCash(8), StartPump(), Cancel(), Pump(), Pump()
Expected result at the last operation: Total=$5

Gas Pump GP-2
Test #1
Activate(4.2, 7.2, 5.3), Start(), PayCash(10), Premium(), StartPump(), PumpGallon(), PumpGallon(), Receipt()
Expected result at the last operation: Total=$7.2; Returned cash=$2.8

Test #2
← no-op
Activate(4, 7, 5), Start(), PayCash(10), Premium(), Diesel(), StartPump(), PumpGallon(), Stop(), Receipt()
Expected result at the last operation: Total=$7; Returned cash=$3