HW4 – Simple Webservices and More Inter-Process Communications

Due Date: *11/2/2012*
Estimated time: *20-24 hours*

Objectives

- Gain some experience with simple remote objects implemented as Webservices
- Become more familiar with inter-process communications and resource management concepts
- Become more familiar with unit testing techniques
- Become more familiar with logging techniques and a logging tool

Overview

In this assignment, you will continue to implement the inter-process communication protocols and resource management functionality for your distributed water-fight game. Specifically, you will need to complete about 60% of the functionality outlined in your requirements definition.

In addition, you will extend your functional requirements to include the ideas listed the next section and implement them by calling an instructor-provided SOAP Webservice.

New Functional Requirements

Step 1 Integrate the following new ideas into your functional requirements. Also, update your analysis to reflect your new understanding of the system and your design document to describe your revised architecture.

1. On start up, a fight manager needs to register itself with a *World-wide Water-fight Statistics System* (WFSS) by calling the *Register* method of a *WFStats* webservice object. The *Register* method needs the following parameters: a Global Unique Identifier (GUID), a name, operation name, and email address
   a. The URL for the *WFStats* webservice should be configuration parameter that the system operator can easily change before starting up the fight manager.
   b. Each instance of fight manager must uniquely identify itself with a GUID. That GUID should never change.
   c. Each fight manager should have a name, like “East Fight Manager”, that is defined in a configuration parameter. The system operator must be able to change this name before starting up the fight manager.

   d. Each fight manager should know the name and email address of its system operator. The operator's name and email address should be defined in a configuration parameter. For now, set these parameters to your own name and email address.

2. When a water fight is started, the fight manager should notify the *WFSS* of the new game by calling the *LogNewGame* method of a *WFStats* Webservice object. This method needs two parameters: the fight managers GUID and a game id.
   a. The game id only needs to be unique to the fight manager.
3. Periodically throughout a water fight and at a game's end, the fight manager needs to update the WFSS with the games current state by calling the *LogGameStats* method of a *WFStats* Webservice object. The *LogGameStats* method has the following parameters:

   - fight manager GUID
   - game id
   - A stats object that includes the following data members or properties
     - Timestamp
     - Current game status
     - Current number of players
     - Largest number of players to in the game at one time
     - Total number of balloon thrown as part of the game
     - Total amount of water contained in the thrown balloon
     - Total number of balloon that hit players
     - Total amount of water that hit players
     - Name of winner if the game has been won

Development

Step 2 Complete approximately 60% of your communication protocols and resource management functionality.

Step 3 Implement the new requirements that you integrated into your functional requirement during Step 1.

Step 4 Complete unit testing for the new functionality implemented in Step 3.

Step 5 Write a README.TXT file that describes what you completed during this assignment. Specifically, describe what to did, if anything, for the advanced requirements.

Submission Instructions

Put your documentation and entire solution into an archive file called CS5200_hw4_<*fullname*>.zip, where *fullname* is your first and last names. Then, submit the zip file to the Canvas system.

Grading Criteria

Basic Criteria (worth up to 85 points)	**Max Points**
Updated requirements definition, analysis, and design documents	5
A quality implementation of the new WFSS functionality	25
A quality implementation of approximately 60% of your communication protocols and resource management functionality	25
Test cases for the new WFSS functionality	20
Advanced Requirements (Worth up to 25 points)	**Max Points**
Additional using test cases (beyond what was completed for HW3) of communication protocols and resource management functionality. Thorough testing of a single non-trivial method, which has two or more parameters and non-sequential control flow, is worth up to 3 points. Thorough testing of a trivial method, which has zero or one parameters or a sequential control flow, is worth 1 point.	0 – 25
A new or improved user interface that allows a human to control a player in the game	0 - 25