Soft8023- Distributed Systems Programming Projects.

On completion please zip up your files including any documents used for drawing the class diagram etc. Upload to Canvas. You must arrange a demo/screencast for week 7.

We have been thrown back in time and you are you are tasked with the job of helping to defend our beloved city Cork from attack by sea. The attack can come from either direction east or west. In Blarney we have a munition s factory but only just enough capacity to make enough bombs/shells to just fend off the invading force. If the attack comes from the west then the shells have to be sent to/deployed to fort Carlisle. If the attack comes from the east then the shells are sent to fort Camden. If the attack force splits up and attacks from both directions we want to divide our creation and distribution of bombs/shells.

We have 3 computers available and they can be networked that's all we could carry back with us. We have a sentry in Youghal and another one in Kinsale. When they see enemy ships coming then they immediately inform the store in blarney to send munitions to the appropriate site. Up to week 6 we are allowed to use local method calls to communicate events. These calls can't happen if the application really is using remote computers. You can't call a method on a different computer, you can't even call a method in another application running on the same computer. It can't happen even if we have two separate applications for the Blarney and fort sites. We improvise and pretend for the first few weeks using local references remembering we will have some work to do on this after week 6, good design here will make things a lot easier later. After week 6 we will be using a distributed technology for communications and we will make remote calls.

We first want to develop a prototype without using a distributed technology to get some good versatile software working that will help us in our eventual task.

Using **threads** and **design patterns**.

Develop the application classes that will be at both remote sites Kinsale and Youghal. These are the subjects/observables. Here we want a thread that mimics the arrival of a ship from the attacking fleet. It's a thread that sleeps for a random time. When it wakes the a ship has arrived, this is then signals to an appropriate class (in Blarney or local at the remote site probably better to keep it at remote site) detailing the type of ship.

When the observer in Blarney is made aware of the attack/s it creates a thread that is responsible for making and transporting the shells to the appropriate fort. This is mimicked by creating a "Dispatch" object that has the type of ship and remote location in a collection. There are 10 ships in the attacking fleet, we have capacity to make 10 shells/bombs. The attacking fleet can come in any numbers Max 10 from either or both directions. Make it more general for any ships if you like..

We will later add the distributed technology communications to this application.

Note if we design this well we will require less effort in the future when we add the distributed technologies/communication.

So design class responsibility as if it was a distributed system even though at the moment (up to week 6) we are only doing a mock up.

Classes

At remote site: observable pattern to signal sighting. Classes/threads to pass the information back to Blarney. An application to kicks off all that happens at Youghal and Kinsale and whatever other classes we feel will add to the flexibility of our software system. Maybe use a GUI here with buttons to signal arrival of ships.

In Blarney: A class/application that is capable of receiving the signal from the remote sites.

Maybe a shared area. And then something that creates the object/orders responsible for making and transporting of the shells to the appropriate destination.

The application so that Blarney is a factory for making shells, to cater for different types of attack ships. Develop a factory that can make different types of shells. Simulate this with a class hierarchy and the factory method pattern.

Shell types 1. Armor piercing 2. Torpedo 3. Blast bomb.

Attack ship types sailing, aircraft carrier or destroyer.

Blast bomb destroys sailing ship. Armor piercing destroys destroyer and torpedo destroys aircraft carrier.

If the attack is arriving from the west then the shells are sent to the opposite side of the harbour.

Project 40%.

Make two class hierarchies one for ships and one for bombs. Mimic a factory making bombs, with the factory method pattern. This factory is at the Blarney site. A thread at Youghal and Kinsale sites makes ships using a factory and a ship class hierarchy.

Observer pattern is to be used to signal sightings.

Singleton pattern to be used on any stores.

Dao pattern for a collection of Ship or Bomb objects plus other details (like destination ..).

Program to interfaces where it is possible to do so. It makes your code more versatile.

Week 6 deliverable the threads patterns etc 40% prepared with a view to distribution.

Design patterns:

https://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

https://www.tutorialspoint.com/design_pattern/observer_pattern.htm

https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm

https://www.tutorialspoint.com/design_pattern/factory_pattern.htm

or factory method pattern its better …..

Threads: Use of threads with sleep 5 marks.

Thread synchronization: Threads with shared area. Shared area probably in Blarney thread, updated by both remote sites hence the synchronization. 5 marks

Producer / consumer can be localized in the Blarney site, as only one bomb can be made at a given time. 10 marks

Design patterns. 20 marks 5 each.

Project 2 60%

Week 6 to 11 we will really distribute this application and use sockets and RMI for the communication (or GRPC if you want I'd be impressed). We could probably stream information from remote sites if we wanted.

Versatility of code 15 marks

Using sockets poll for the information from Kinsale site along with proxies 20 Marks

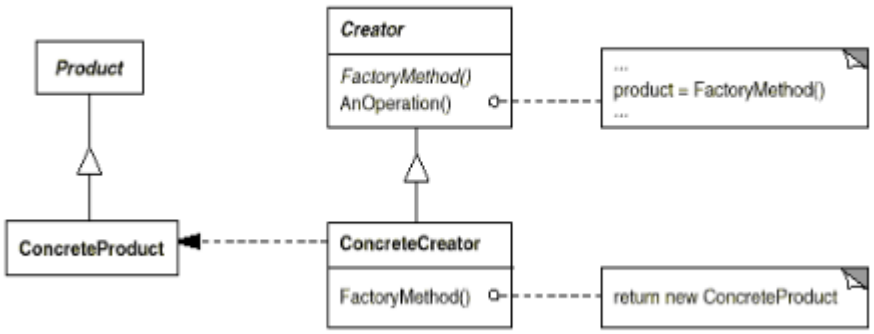
Using RMI/GRPC use call-back from Youghal site. Marks 25 marks

Due week 11 and demoed week 12.

Guide to developement

Use the factory method pattern to create Factory systems. One for ships and one for bombs:

Structure



Just mock them up. Ship abstract super class contain all common to ship. Subclasses contains details. Same with bombs, this is the product hierarchy. Now write the factory interface, next write the factory method classes. One factory for each product.

Next Create a main application. This kicks off everything. Definitely we make three treads to begin with:

Thread1 Blarney thread

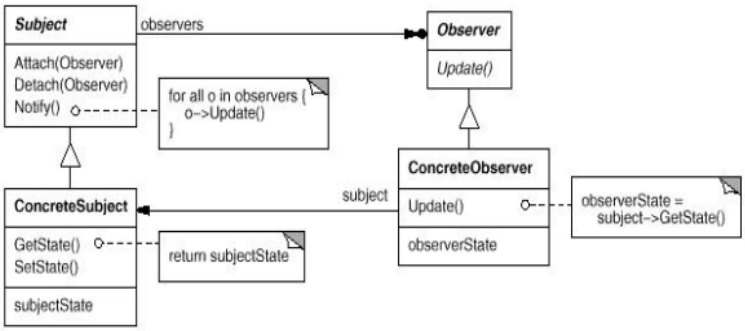
And two

Threads for the remote sites.

The remote sites will be implementing the observer pattern. This can be done with maybe just two threads. The remote thread creates a new thread called the observable thread whose responsibility is to create the signal when a ship arrives.

Remote thread maybe acts as an observer. It creates a thread which will be the observable. As per the following diagram.

- Structure



The observable thread has the following:

Notify observers.

This thread can also create a GUI with a button for each type of ship. The listener for the button calls the update method on observers via the notify method.

The observer thread will have an update method. The update method uses the ship factory to create a ship object and will send this to the blarney thread or Blarney shared area.

Dealing with the details at the Blarney site.

Remember that there are two remote sites. These will have to access one shared area to store the details of the ships as they arrive. They must be mutually excluded from accessing the shared area at the same time. Here we will make use of the singleton pattern and the word synchronized to allow the safe and lossless signaling of the arrival of ships, the ships are stored in the shared area.

Remember that a bomb system has its own class hierarchy and factory. So factory method pattern is used here also.

The Blarney site can only make one bomb at a time. So all else stops while the bomb is being manufactured. When the site is finished making the bomb. It then gets information about what it is to make next i.e. which ship arrived next, so the order of the arrival of the ships is

important. Work in a producer/ consumer scenario here. Producer/consumer can be local to the Blarney thread.

Store details of all work done at the Blarney site. Use the Dao for this an arraylist do as the store and persist this in a file of serializable objects.