Workflow and Documentation

# End-Product

## Directions

The more technical the topic, the harder it is to make a relatable assignment.

Sooo... how about lifeboats on the Titanic.  Except in this version it's in the Caribbean and there are no sharks.  So everyone splashes around for a while and nobody dies.  Then they all go to Papas and Beer.

Classes: Lifeboat that holds a list of Passengers.  Crew that hold a rope (shared\_ptrs) to a Lifeboat.  Captain that is looking (weak\_ptrs) at all the lifeboats.

At start: Main has a single queue of passengers.  A Passenger has an age and a gender, both are randomized.  The queue starts with 50 of them.  Two Lifeboats exist.  Each have a capacity of 20 and each have two Crew holding on to them with shared pointers.

For each person in line, "Women and children first".  If the Passenger is a woman or less than 10, they get the first boat, or the second if the first is full.  Men try the second boat first, and can only try the first boat if it is their second try (so all women have had a shot).  If you can't get on a boat, get back in the queue.  After someone gets on a boat, there is a 5% chance that one of the Crew will let go of their shared pointer accidentally.  When the Captain (who had weak pointers to the boats) sees the boats are gone, the program ends.

Passenger: Gender, Age

Captain: list of weak pointers to Lifeboats

Crew: Strong pointer to Lifeboat

Lifeboat: List of Passenger pointers

Main: Two Lifeboats, queue with 50 Passengers, two Crew per Lifeboat, 1 Captain

Main outline

Set up objects

While the queue isn't empty and the captain says there are still boats

   If woman or child

      Try boat 1

   If not in a boat

      Try boat 2

   If still not in a boat

      Back to end of queue

   If someone loaded

      Check if a Crew drops a boat

Output:

Lifeboat's destructor should print how many people got on it.  It get destroyed when the last Crew object lets go

After Captain detects all boats are gone, output how many people will be swimming to shore because they were left behind

## Notes

Passengers don't do anything

Captains look on the edge seeing how many lifeboats are left (weak\_ptr)

Crew hold lifeboats (shared\_ptr) all crew have their own shared\_ptr

If guys get to front of line twice let him on boat

When the crew let's go of boat, boat gets deleted

# Workflow

Steps:

1. Outlay and understand the project
2. Create the documentation of possible implementation
3. Transfer documentation to UML diagram
4. Code implementation
5. Revise

# Documentation

## Captain.h

* Observes how many boats are left

std::list<weak\_ptr<Lifeboat>> mSightedBoats

* Sight of lifeboats

bool BoatsHere()

* If no boats here, returns false

## Crew.h

* Hold a boat and can accidentally drop it

constant float DROP\_CHANCE

* Constant drop chance, 5%

shared\_ptr<Lifeboat> mBoat

* Lifeboat connected to Crew

void Drop()

* Method to possibly accidentally drop boat

## Lifeboat.h

* Two instances, gets destroyed when crew shared\_ptr is lost

constant int CAPACITY

* Constant cap of boats, 20

std∷list<shared\_ptr<Passenger>> mPassengers

* Passengers on the boat

void AddPassenger(shared\_ptr<Passenger>)

* No logic, simply adds Passenger to mPassengers

~Lifeboat()

* Prints number of mPassengers when destroyed

## Passenger.h

* People who need to get on boats

enum gender

* Types of gender, male and female

int age

* Below CHILD\_MAX means is a child

const int CHILD\_MAX

* Max age to be considered a child, 10

gender mGender

bool mTried

* Set to true if failed first try

Passenger()

* Randomizes creation of age and gender

int GetGender()

int GetAge()

bool HadTried()

# UML Render

Diagram

Description automatically generated