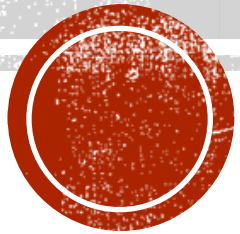
FINITE STATE AUTOMATA

Treasure Hunt



THE ELEVENTH PLANET

Adithya K. L. N -Section 4 - 009306752

Tanmay Bhatt -Section 4 - 011499072

Siddharth Daftari -Section 4 - 011457004

Gagan Jain -Section 4 - 011504441

Rushikesh Pawar -Section 4 - 011447722

WHAT ARE FINITE STATE AUTOMATA?

- **Finite State Automaton** (singular: Automaton) is a model of behavior composed of
  - Finite number of states
  - Transitions between those states
  - Rules
- **State** is something for which a true or false statement can be made
- **Rule** is a way to transition from one state to other

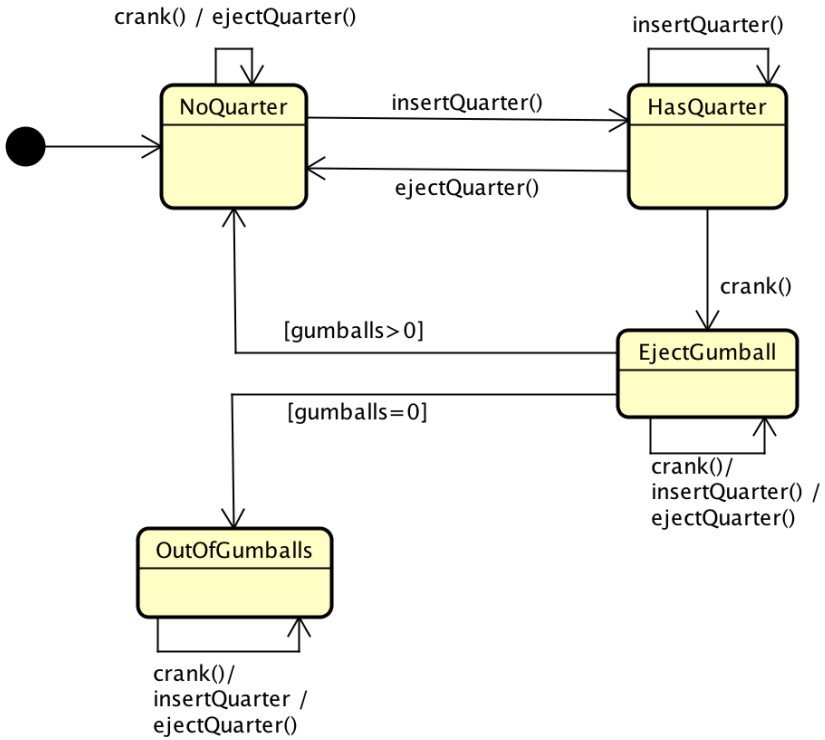
WHY SHOULD YOU CARE ABOUT FINITE STATE MACHINES?

- They represent real behaviors and systems
- They solve many problems like *string matching*, *control of systems*, *digital circuit design*
- It's easier to think about complicated systems visually

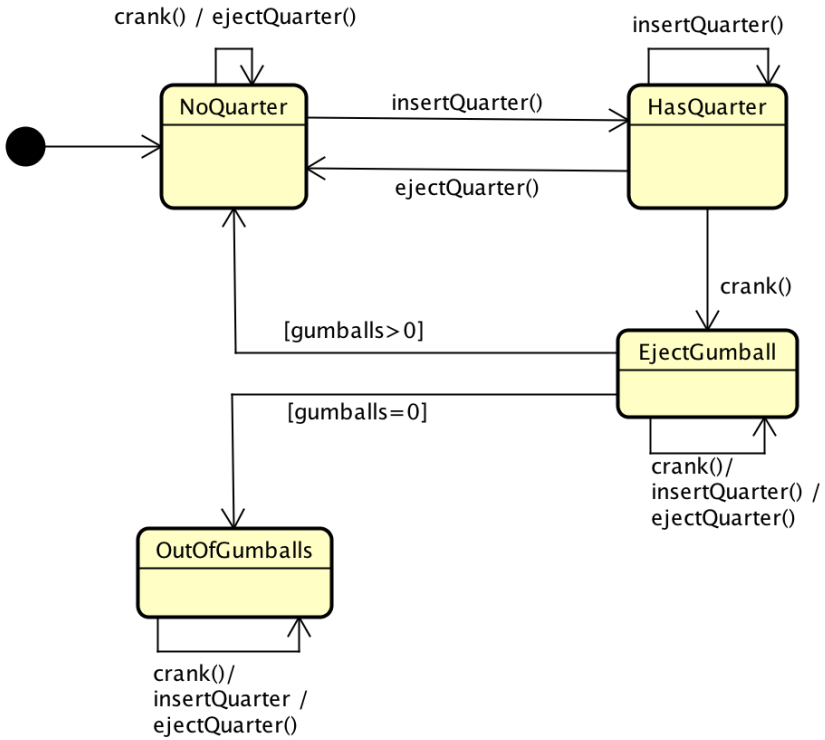
WHAT ABOUT FINITE STATE AUTOMATA IN OOP TERMS?

If you have an object with methods that you call on certain events, and some (other) methods that have different behavior depending on the previous calls.... surprise! you have a state machine!

ANY EXAMPLES THAT COME TO YOUR MIND?

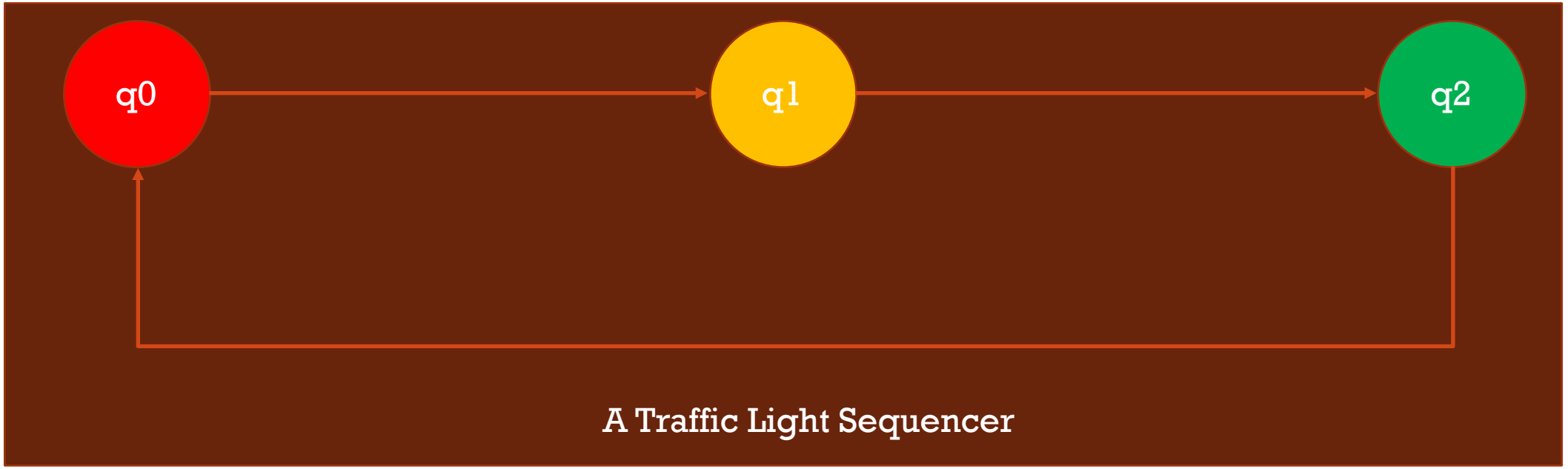


ANY EXAMPLES THAT COME TO YOUR MIND?



Familiar?

APPLICATIONS OF FINITE STATE MACHINES....

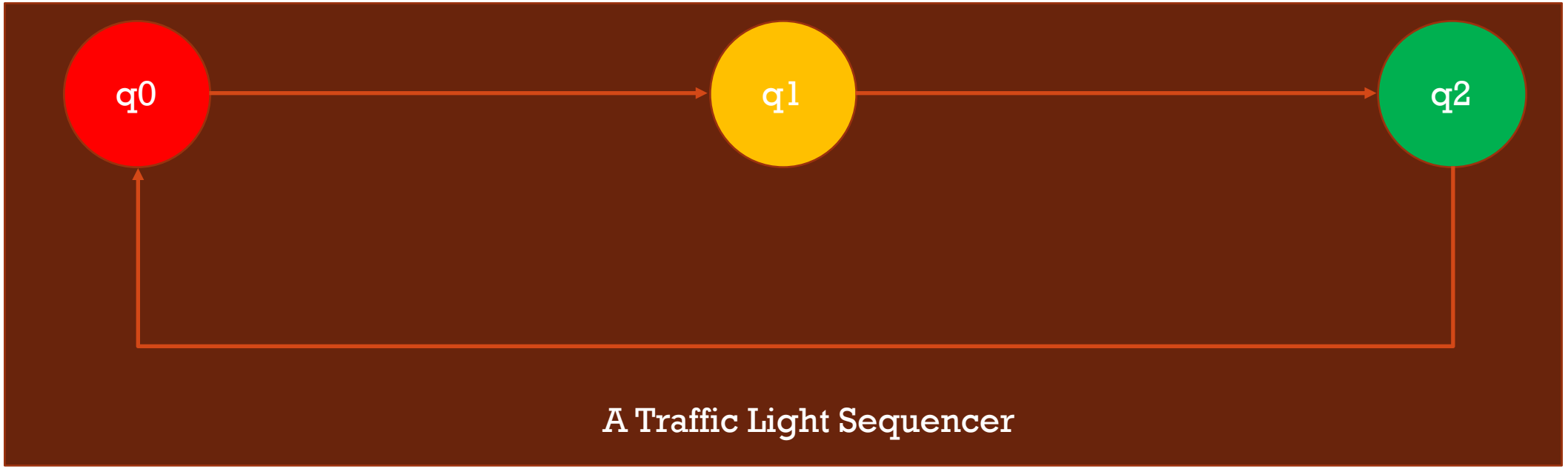


A Traffic Light Sequencer

A TRAFFIC LIGHT SEQUENCER

Traffic Light (time triggered | sensor [event] triggered)

- *States*: RED, YELLOW, GREEN (simplest example)
- *Transitions*: After a timer change RED to GREEN, GREEN to YELLOW, and YELLOW to RED. Could also be triggered on sensing cars in various (more complicated) states.



APPLICATIONS OF FINITE STATE MACHINES....

1	2 abc	3 def
4 ghi	5 jkl	6 mno
7 prs	8 tuv	9 wxy
*	0	#

An alphanumeric keypad

APPLICATIONS OF FINITE STATE MACHINES....

1	2 a b c	3 d e f
4 g h i	5 j k l	6 m n o
7 p r s	8 t u v	9 w x y
*	0	#

An alphanumeric keypad

Pressing the button *2/a/b/c* in dial mode prints the number 2.

Two consecutive presses on the button *2/a/b/c* in texting mode prints the letter b.

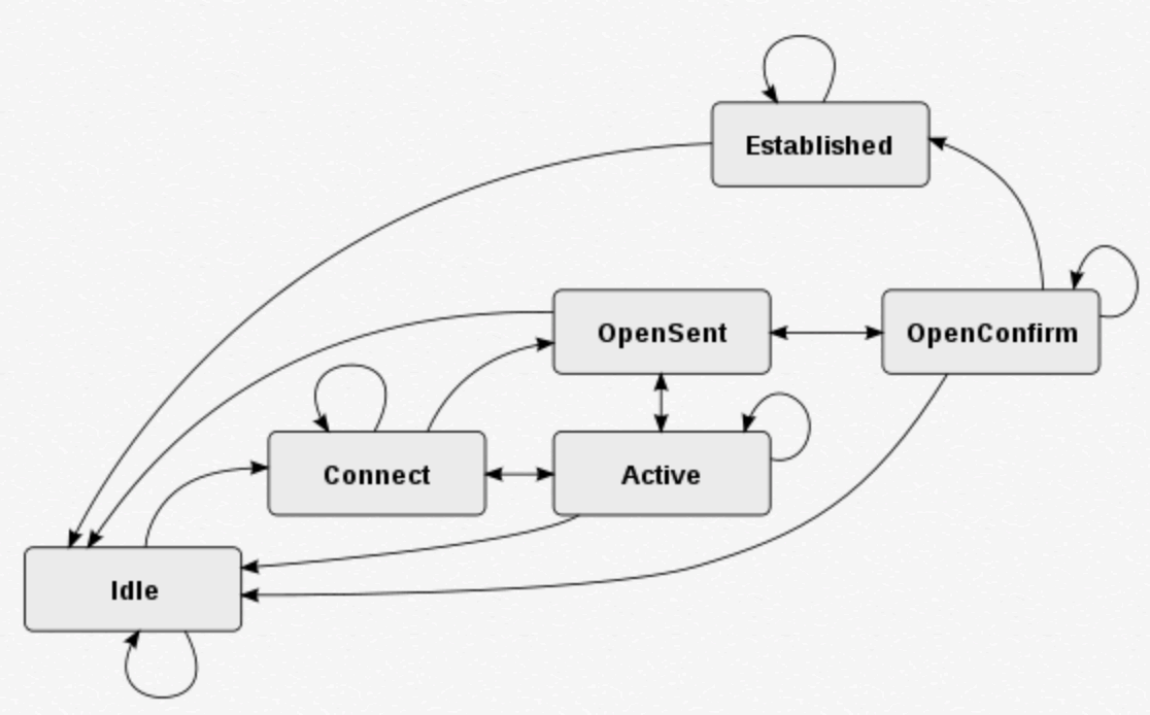
Four consecutive presses on the button *2/a/b/c* in texting mode prints the number 2.

Long pressing the same button in text mode prints the number 2.

MORE APPLICATIONS OF FINITE STATE MACHINES….

- **A Safe** (event triggered)
  - *States*: Multiple "locked" states, one "unlocked" state
  - *Transitions*: Correct combinations/keys move you from initial locked states to locked states closer to unlocked, until you finally get to unlocked. Incorrect combinations/keys land you back in the initial locked state (sometimes known as *idle*.
- **Vending Machine** (event triggered, a variation of the **safe**)
  - *States*: IDLE, 5_CENTS, 10_CENTS, 15_CENTS, 20_CENTS, 25_CENTS, etc, VEND, CHANGE
  - *Transitions*: State changes upon insertion of coins, bills, transition to VEND upon correct amount of purchase (or more), then transition to CHANGE or IDLE (depending how ethical your Vending Machine is)

A SLIGHTLY COMPLICATED ONE: BORDER GATEWAY PROTOCOL



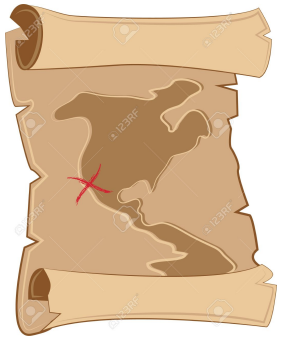
BGP is a protocol which backs the the core routing decisions on the internet. It maintains a table to determine the reachability of hosts from a given node, and made the internet truly decentralized.

The BGP protocol determines the messages that are sent to peers in order to change their state.

TREASURE HUNT

Goal: Find the best route to the Treasure Island.

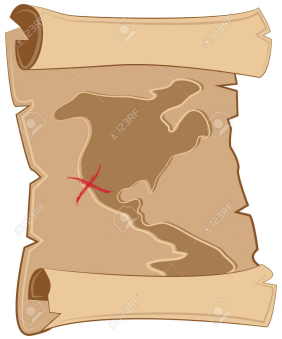
Context: Friendly pirate ships sail along a fixed set of routes between the islands in this part of the world, offering rides to travelers.



TREASURE HUNT

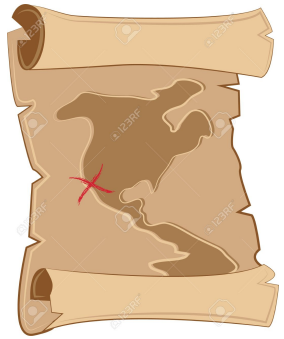
Rules:

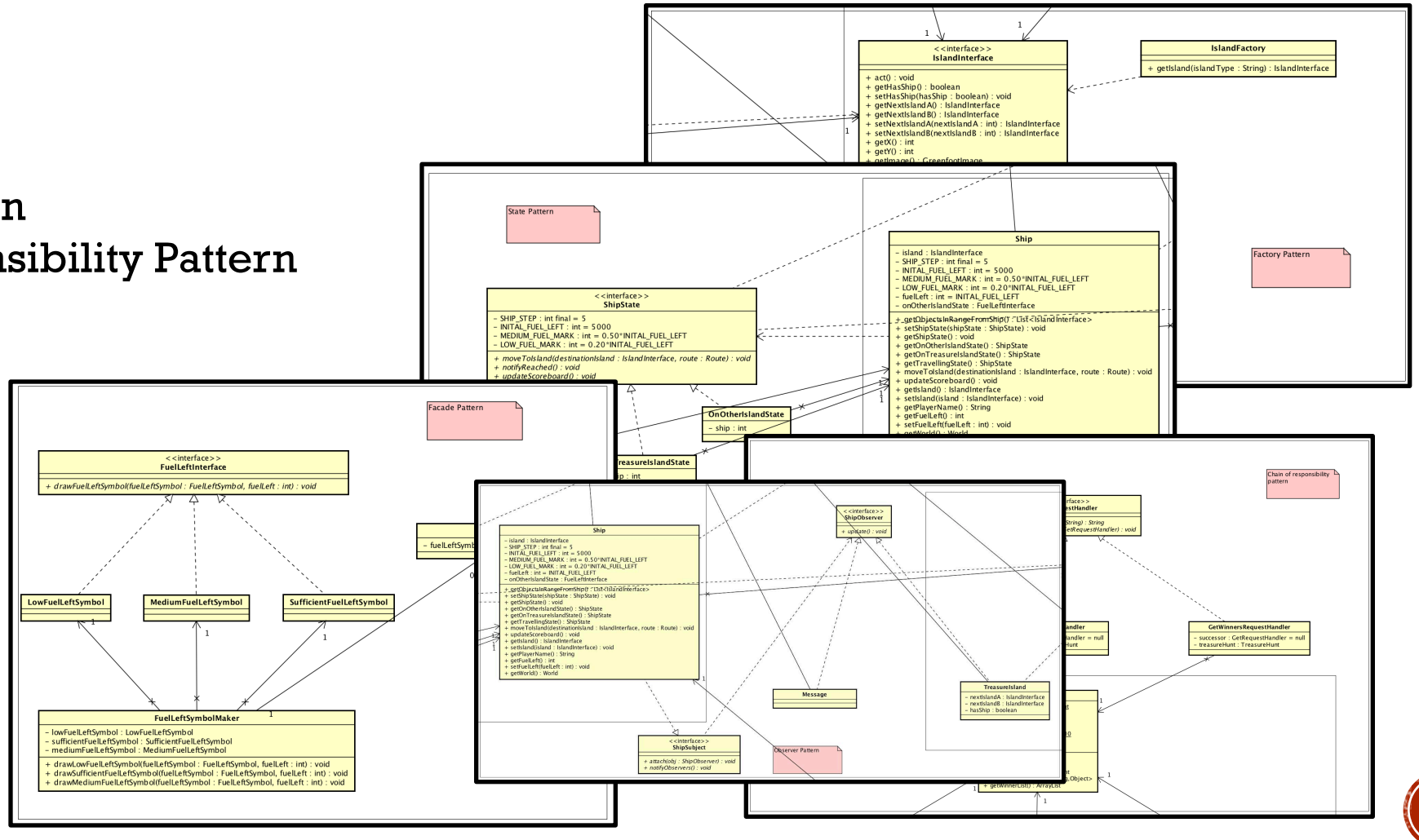
- Each island has two departing ships, A and B, which you can choose to travel on.
- At each island you arrive at you may ask for either ship A or B (not both).
- The person at the island will tell you where your ship will take you to next, but the pirates don't have a map of all the islands available.



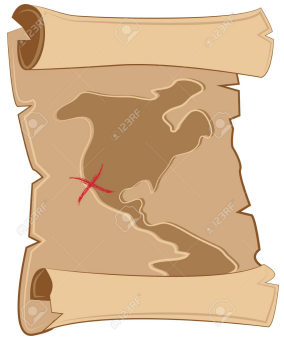
DESIGN PATTERNS IMPLEMENTED

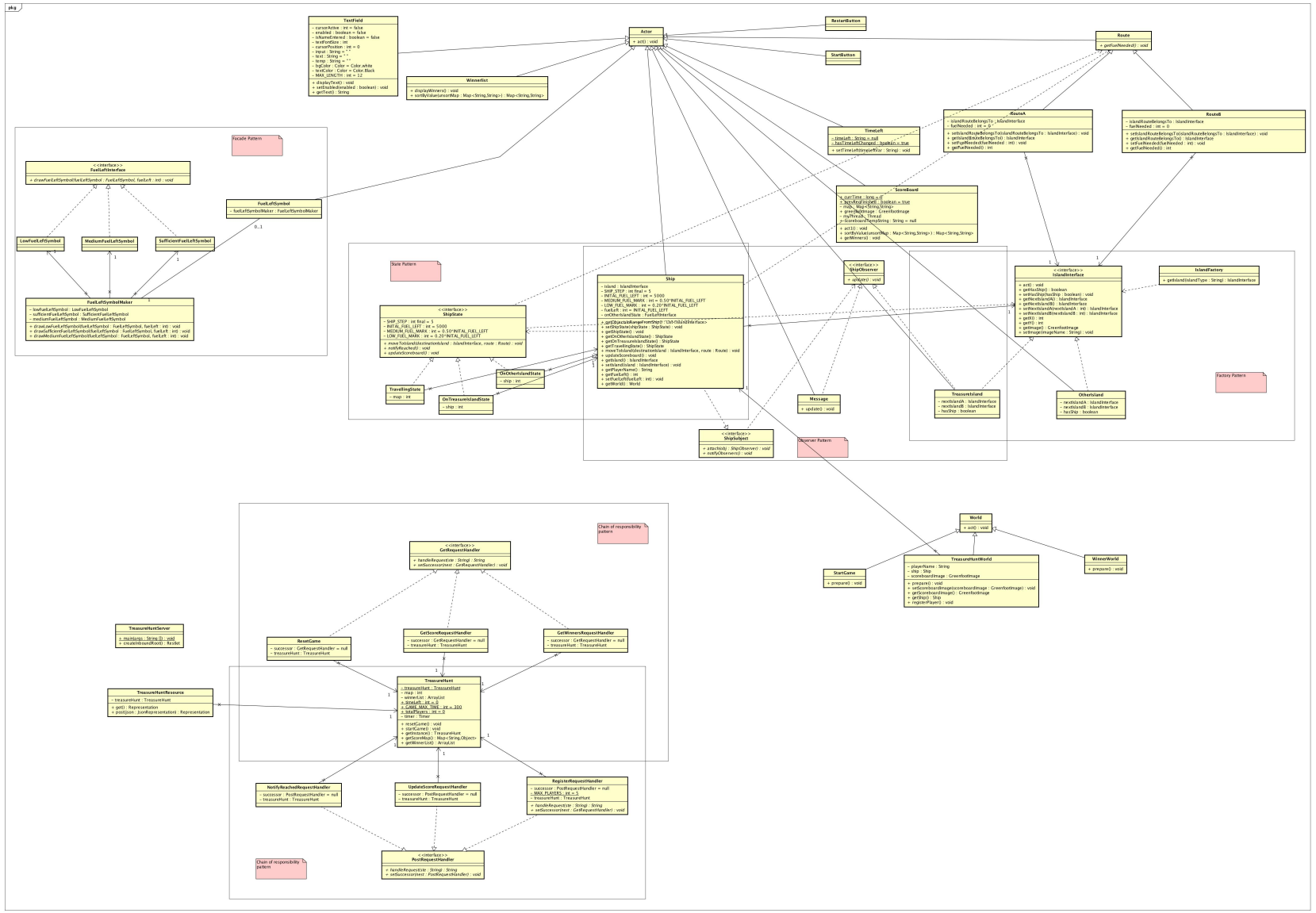
- State Pattern
- Factory Pattern
- Observer Pattern
- Chain of Responsibility Pattern
- Facade Pattern



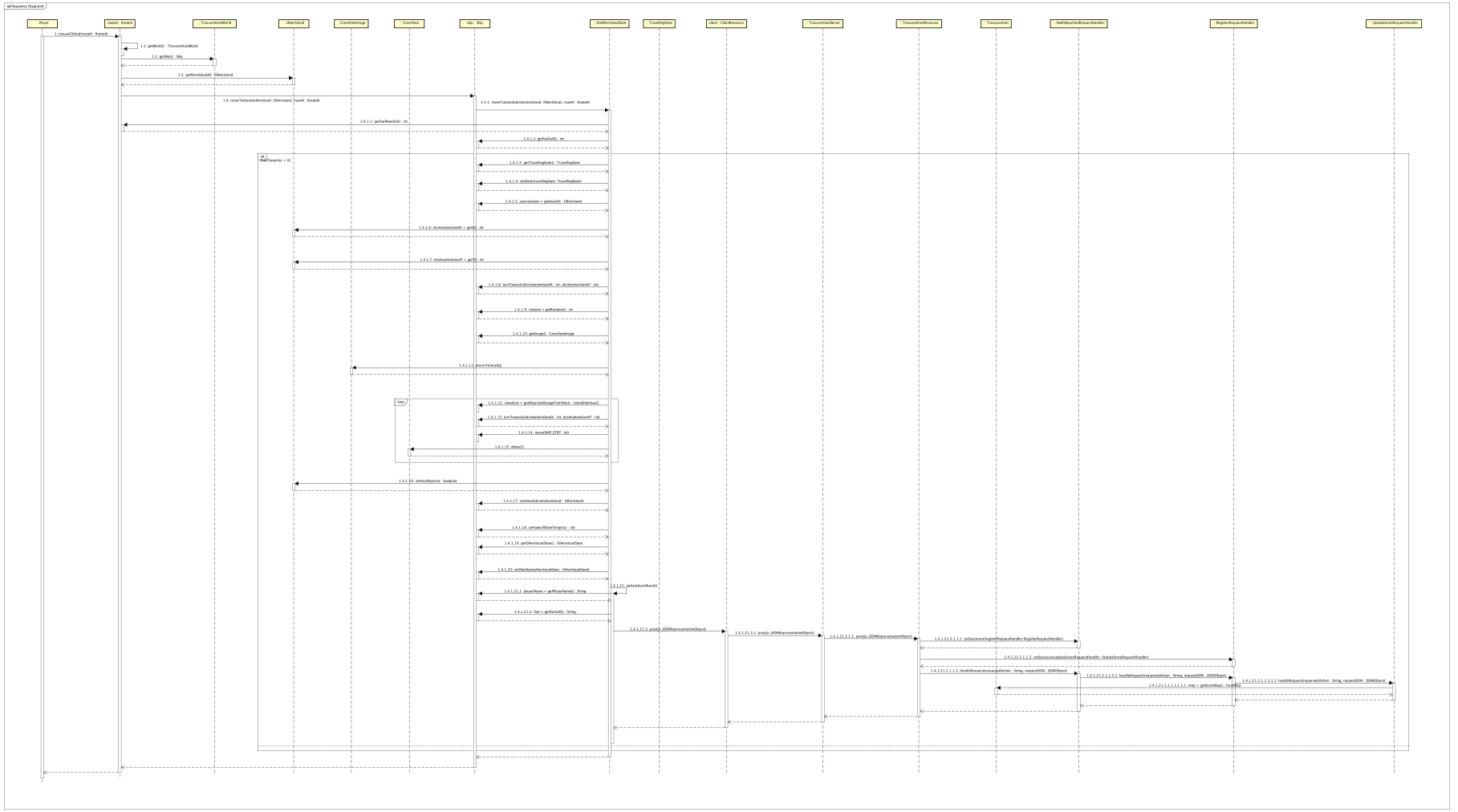


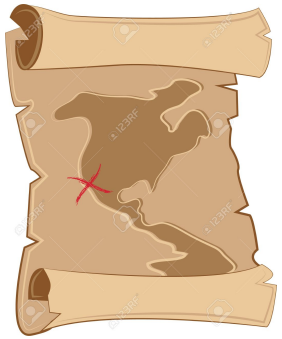
CLASS DIAGRAM



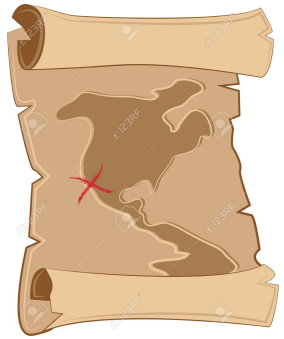


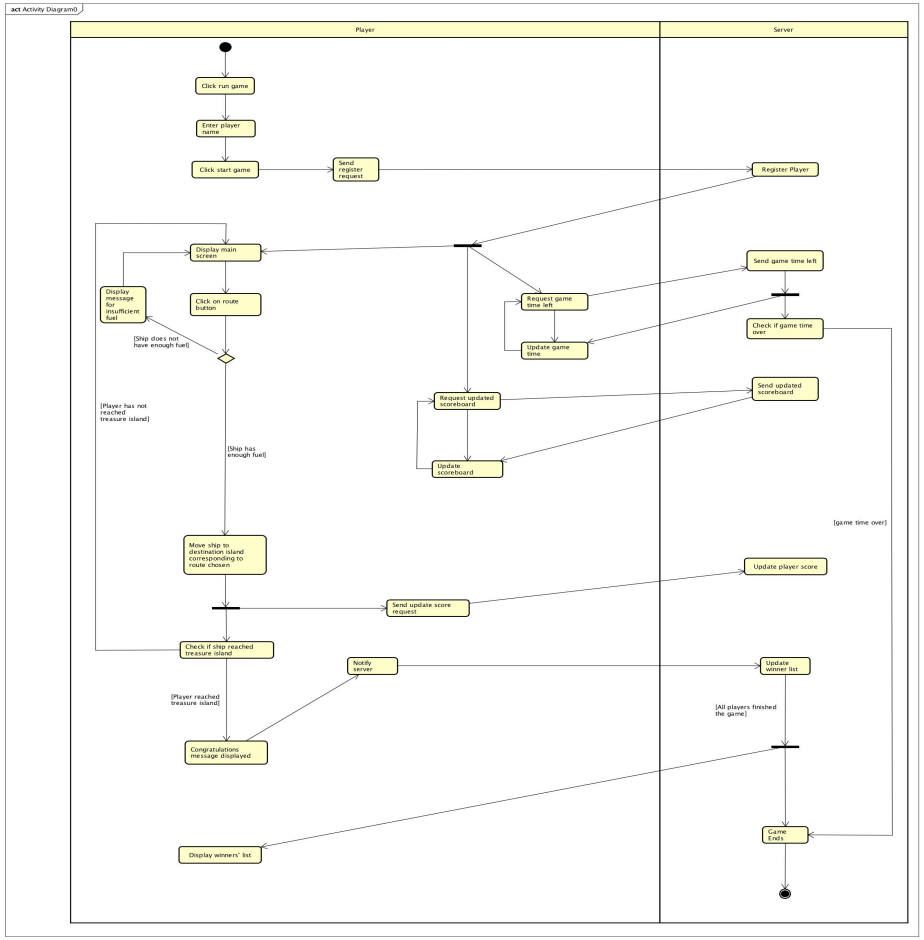
SEQUENCE DIAGRAM





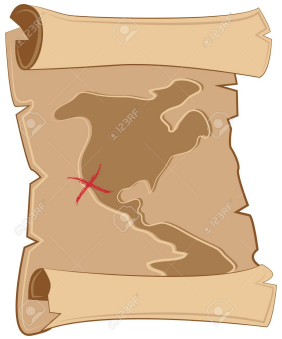
ACTIVITY DIAGRAM





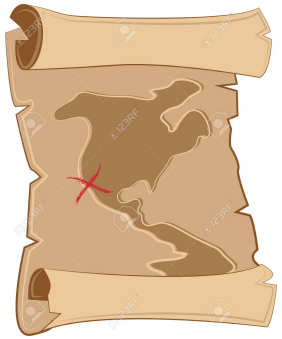
USE CASE SPECIFICATIONS

Use Case Name	Move ship from one island to another.
Related Requirements	Ship should be able to move from one island to another.
Goal in Context	The player clicks on island routes to move ship from one island to another.
Preconditions	The ship should be present on island whose route is clicked. The ship needs to have enough fuel left to traverse to the next island.
Successful End Condition	The player reaches the destination island.
Failed End Condition	The player does not reach the destination island or game time is over.
Primary Actors	Player
Secondary Actors	None.
Trigger	Player chooses either Route A or Route B from the current island.



USE CASE SPECIFICATIONS....CONTD

Main Flow	Step	Action
	1	Player clicks on either Route A or Route B of an island.
	2	Verify whether ship is present at that particular island.
	3	Fetch the destination island.
	4	Check if ship has enough fuel to reach destination island.
	5	Ship takes chosen route.
	6	Ship reaches destination island.
Extensions	Step	Action
	2.1	No action taken on ship if it is not present at the that island.
	4.1	Ship cannot move to destination island if there is not enough fuel left.
	6.1	Game ends if destination island is Treasure Island.

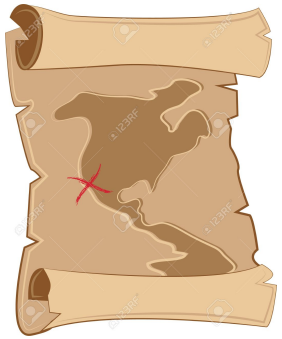


UI WIRE FRAMES

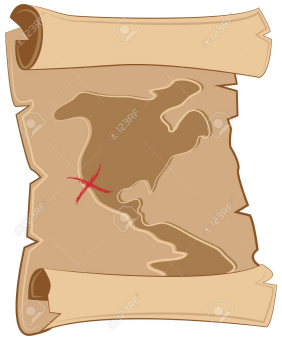
Treasure
HUNT

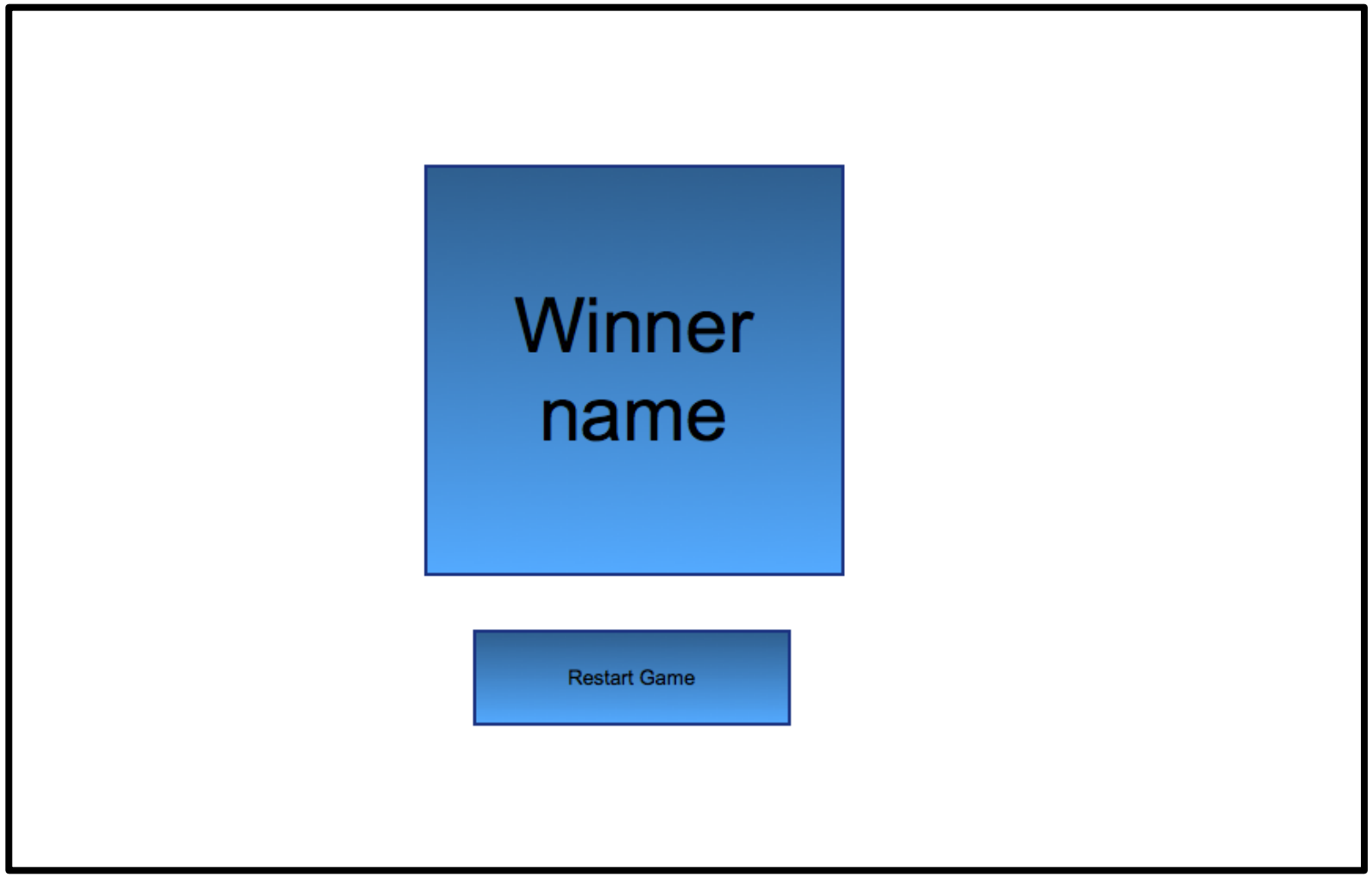
Enter your name

Start Game

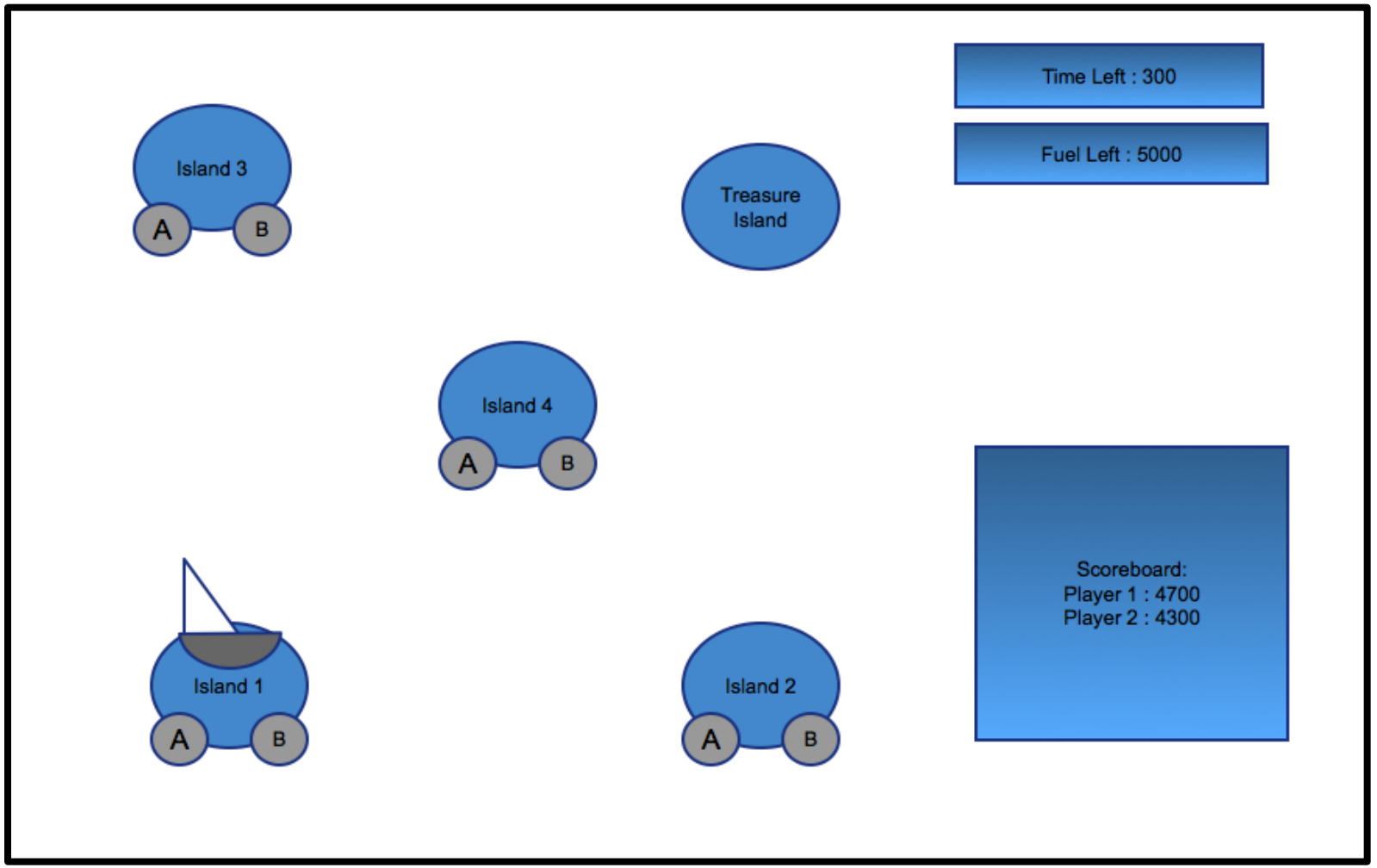


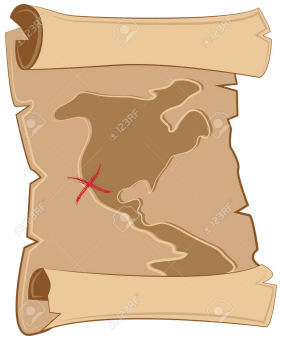
UI WIRE FRAMES....CONTD





UI WIRE FRAMES....CONTD





SUMMARY

- What are Finite State Automata?
- Why should you care about Finite State Automata?
- Finite State Automata in OOP terms
- Examples and applications of Finite State Automata
  - Gumball, Vending Machine, Traffic Light Sequencer
  - Alphanumeric Keypad, Border Gateway Protocol
- Treasure Hunt:
  - A java based game developed on Greenfoot
  - Player finds shortest path to Treasure Island
  - UML Diagrams
  - UI Wire Frames

THANK YOU MATEY!