10/10/2008

M5: Oregon Trail Design Documentation

Oregon Trail Pioneers of the Smalltalk Persuasion

Mary Lane / Justin Lauria / Chris Wiggins / Sam Rickles

M5: Oregon Trail Design Documentation

Oregon Trail Pioneers of the Smalltalk Persuasion

Changes to Utility Objects

- The ‘Turn’ class was eliminated and its responsibilities were allocated to the Gameplay, Climate, Trail, and RandomEvent.
- The ‘Date’ class was added to assist the Climate class in keeping up with the specific time of year.

Changes to Application Objects

1. InitializeGameAppModel, GamePlayAppModel, TradeAppModel, LandmarkAppModel, RiverCrossingAppModel, StoreAppModel, SettlementAppModel, RandomEventAppModel, and HuntAppModel were added.

Updated Scenarios

Scenario 1

A player named Chris gets home from a long day of classes and homework, which he left in his dorm room by accident and forgot to turn in by the deadline. In his frustration, he decides that he needs to play a fun game. He settles on Oregon Trail. He opens up the program and creates a new game. Chris chooses his profession, buys initial items, and starts moving along the trail. After one turn, the wagon wheel breaks and Chris decides to fix it, costing him one wagon wheel from his inventory. The player then proceeds with the game and takes another turn. At this point, Chris reaches a river, and he is provided with three options: to caulk the wagon and float across, ford the river, or pay the toll for a ferry. Chris decides to float across and flips over, as this option has a chance of failure. Chris loses his clothes and some bullets. He is disheartened by this, but he continues along the unforgiving path.

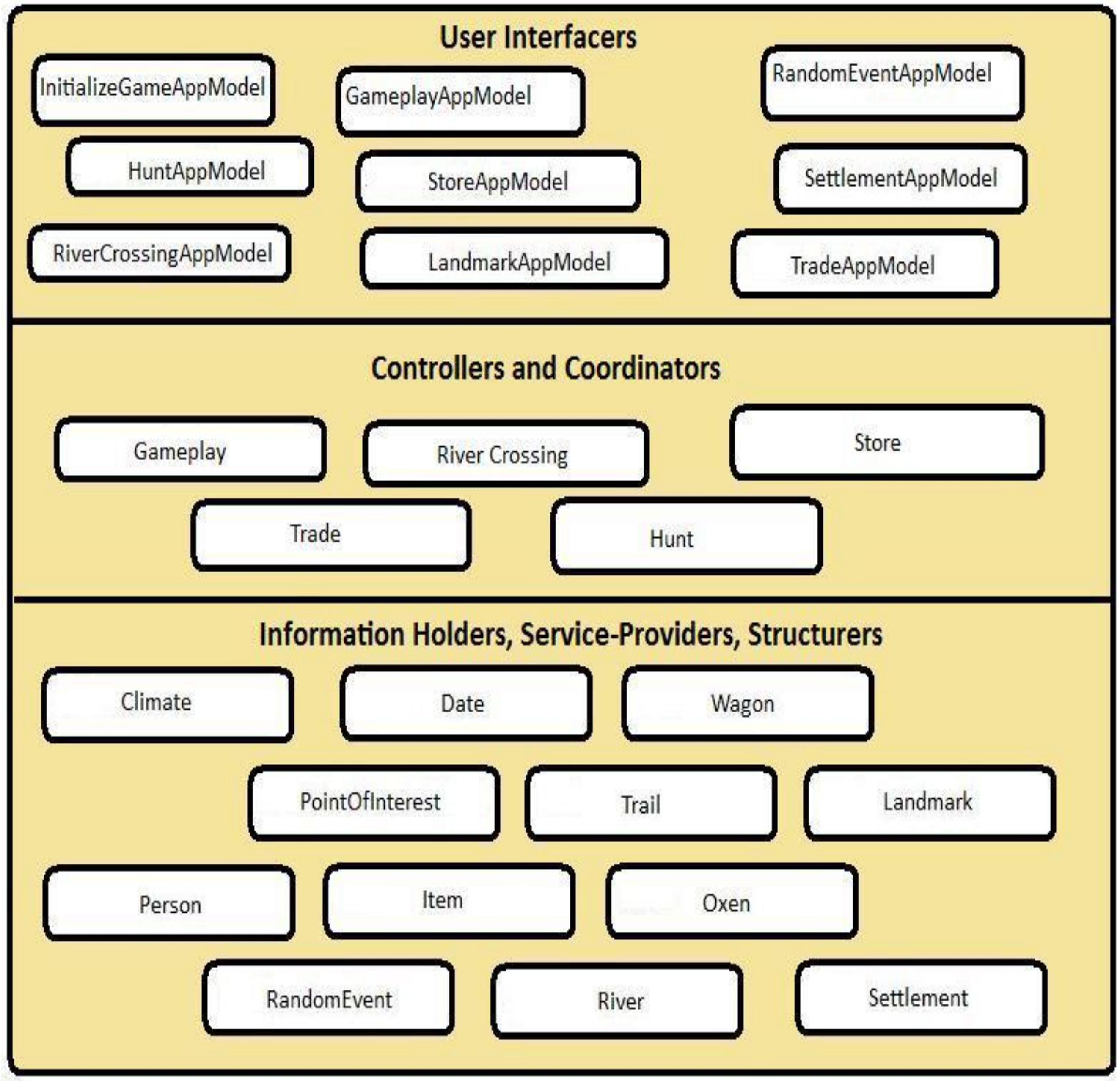
Scenario2

A dedicated Oregon Trailer has been playing for several hours, and in the game, the date is December, and snow storms have slowed wagon progress. Food levels are low and rations have been brought down to bare bones. Party member Bill catches a cold, and without rest, he will surely die. The player is not concerned with Bill's health and continues pushing towards Oregon at a grueling pace. Bill's health fades as the days pass. Only a few turns outside of Fort LauRiWigLa, Bill bites the dust. Although the rest of the party members are failing in health as well, stocking up in food and resting at the fort strengthens their health.

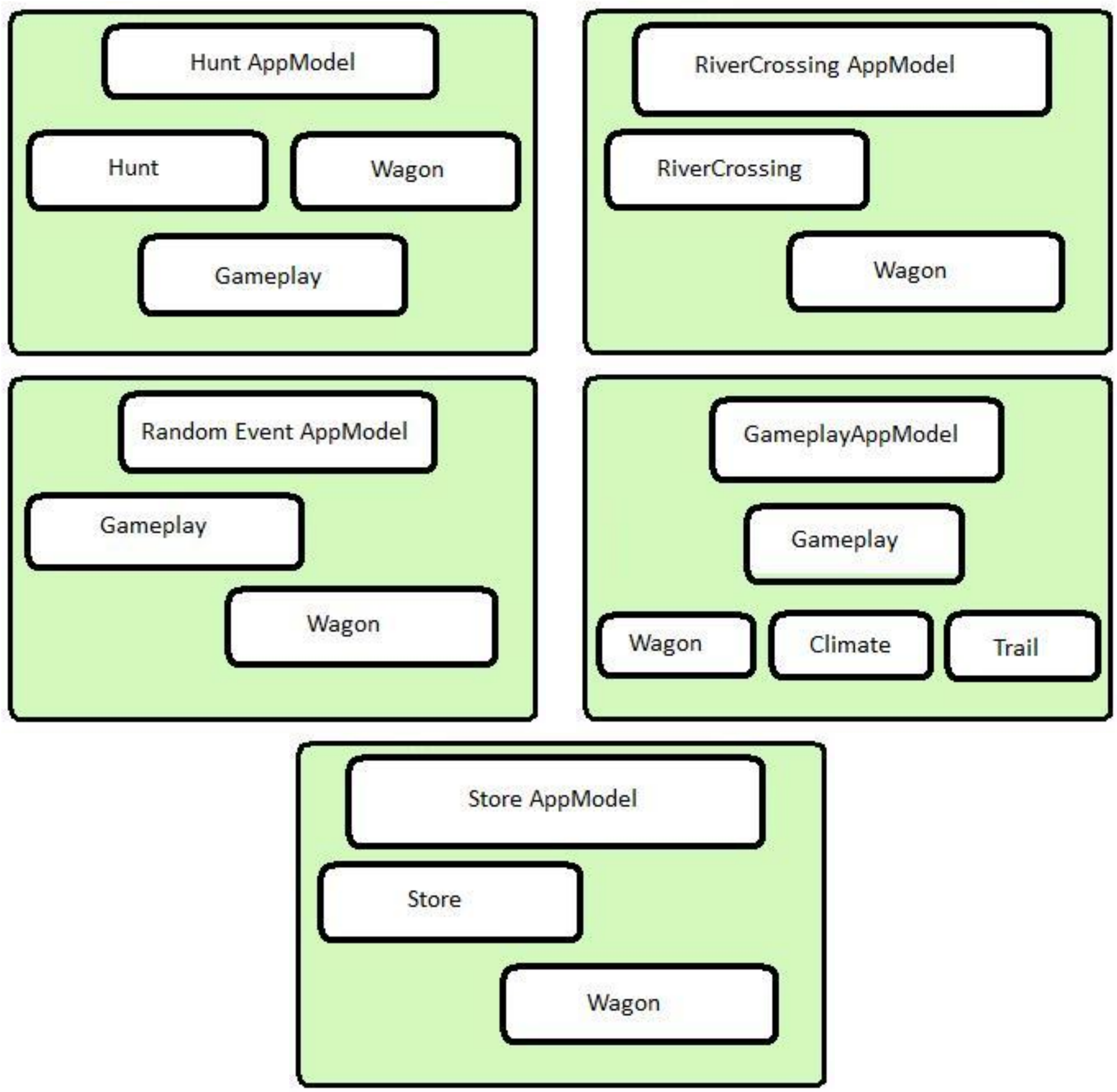
Scenario 3

A player, Kat, has been inconsiderate of the fellow crew members and let them die off. In hopes of getting to Oregon and keeping all of the gold for herself, she's been moving at a steady pace with bare bones rations. She has been hunting enough to keep herself alive and resting enough to stay healthy for the duration of the trip. Finally, after a long and strenuous journey, the player makes it to Oregon and wins the game. Woo hoo!

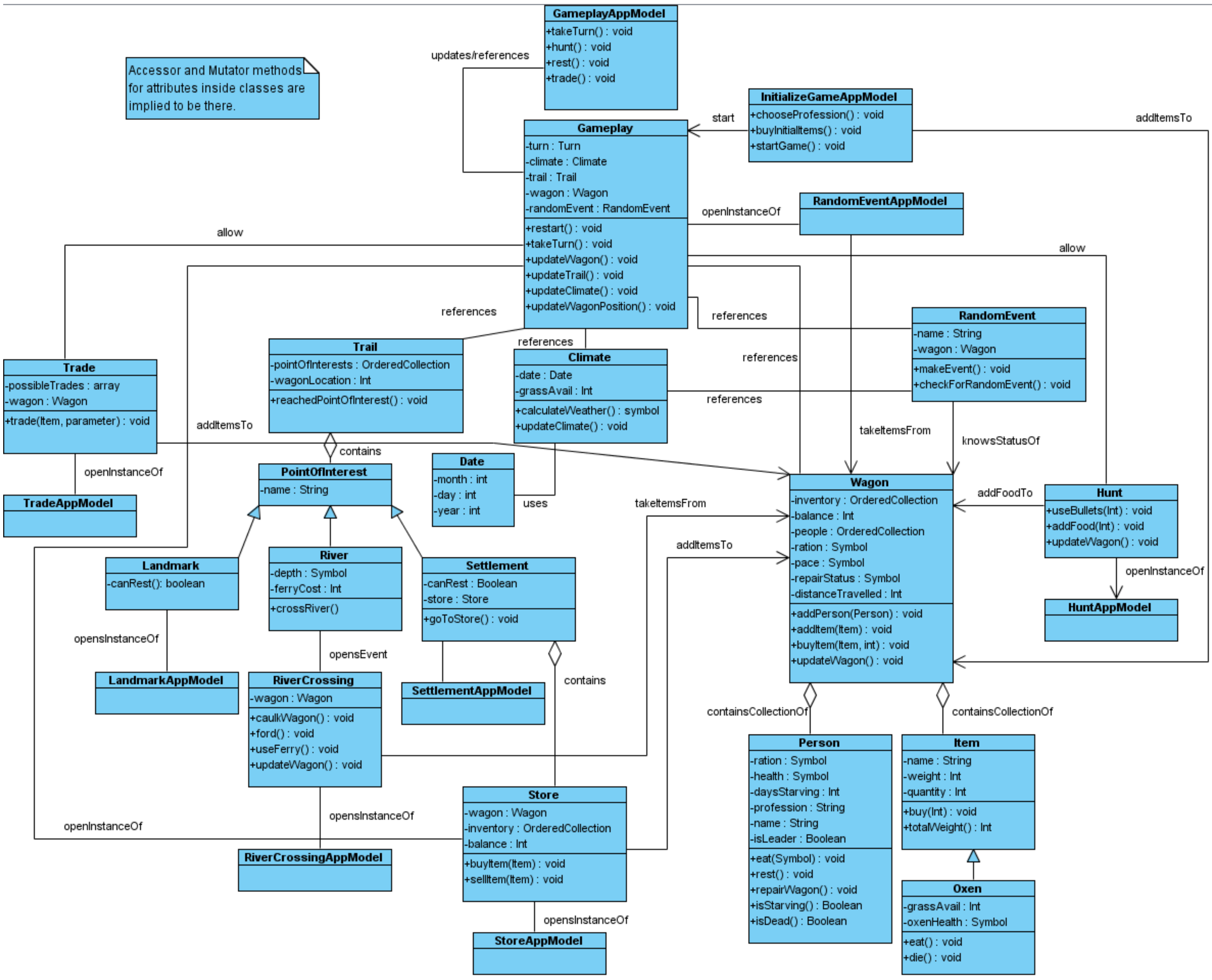
Software Architecture



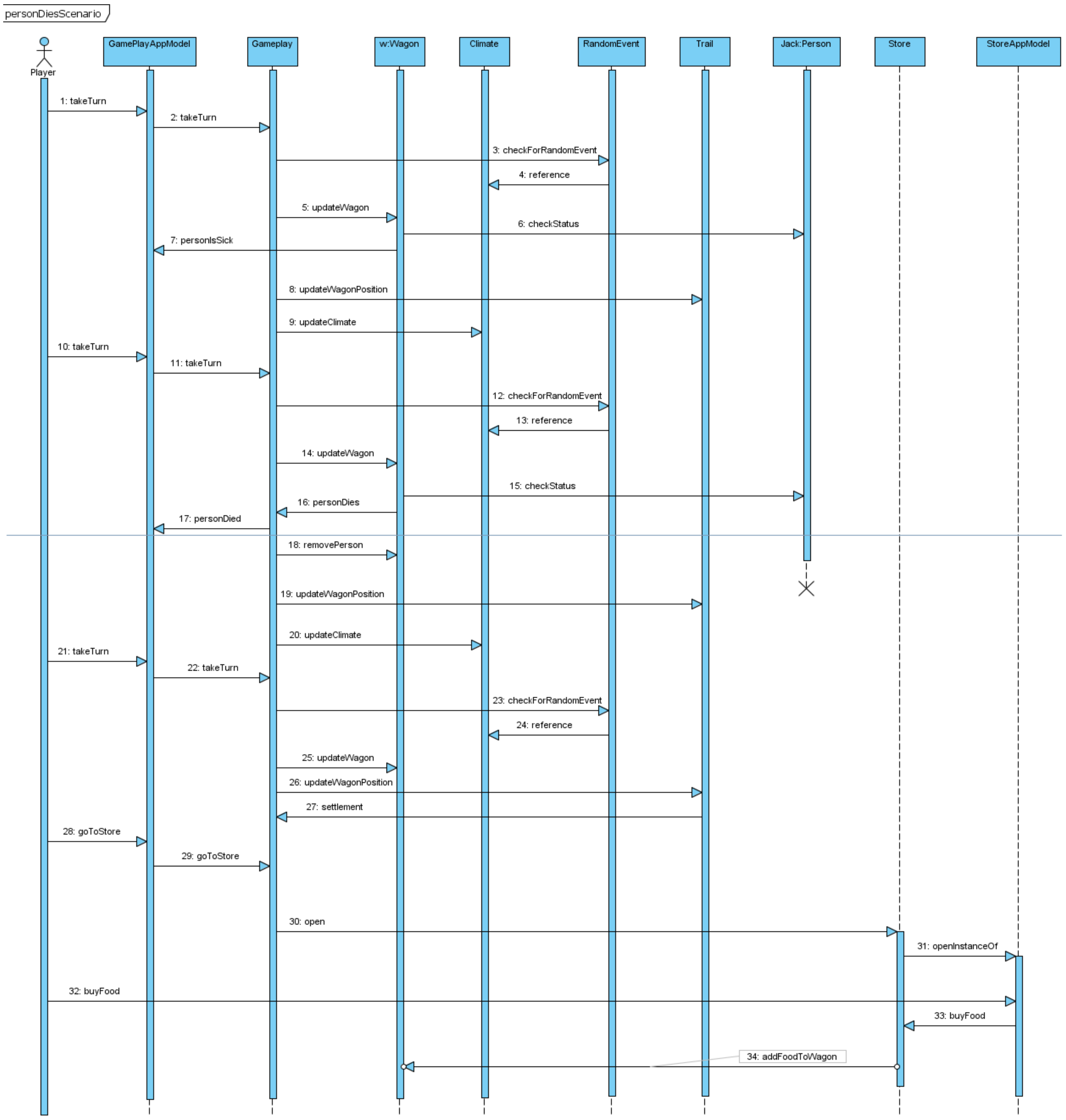
Trust Boundaries

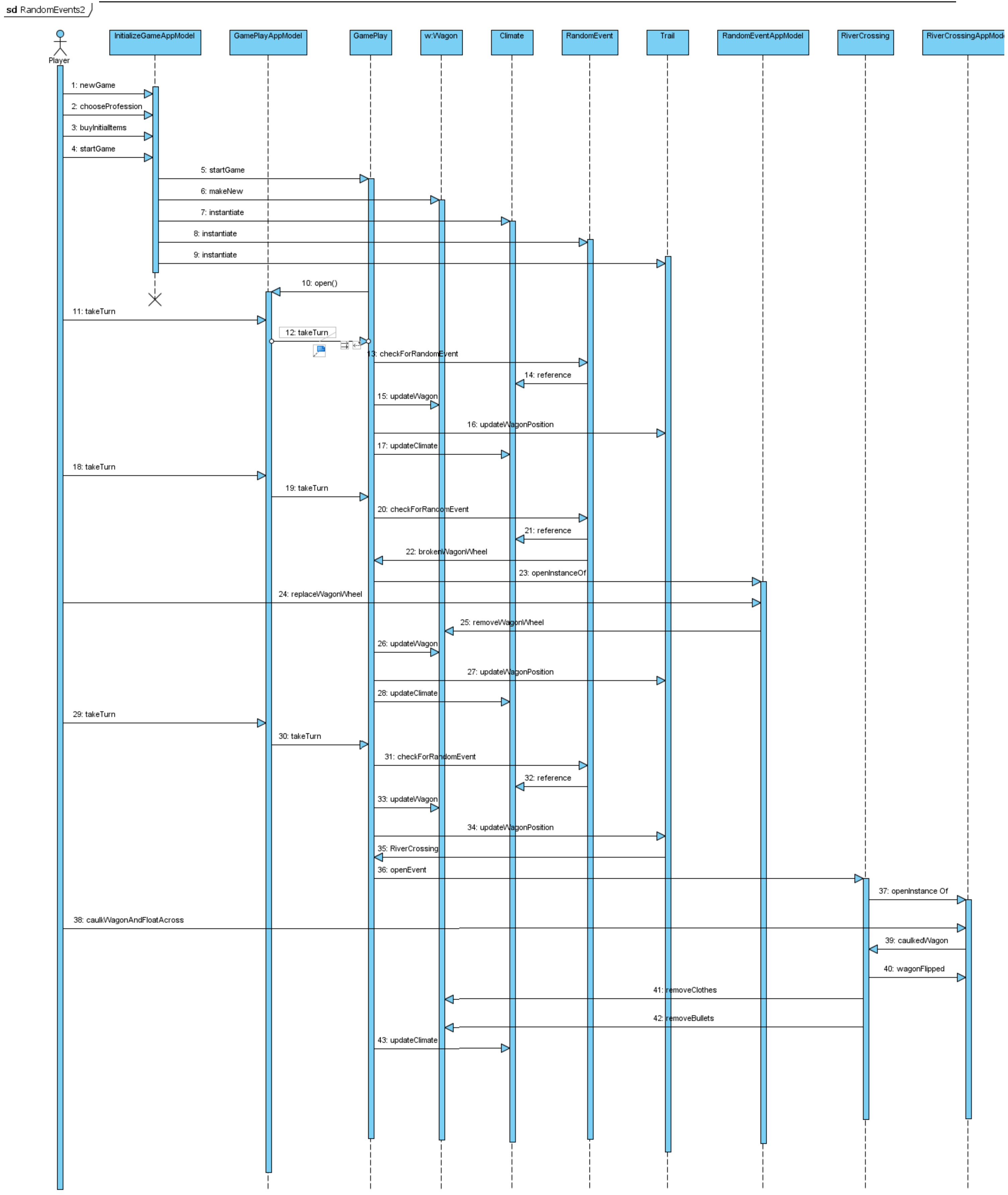


UML Class Diagram



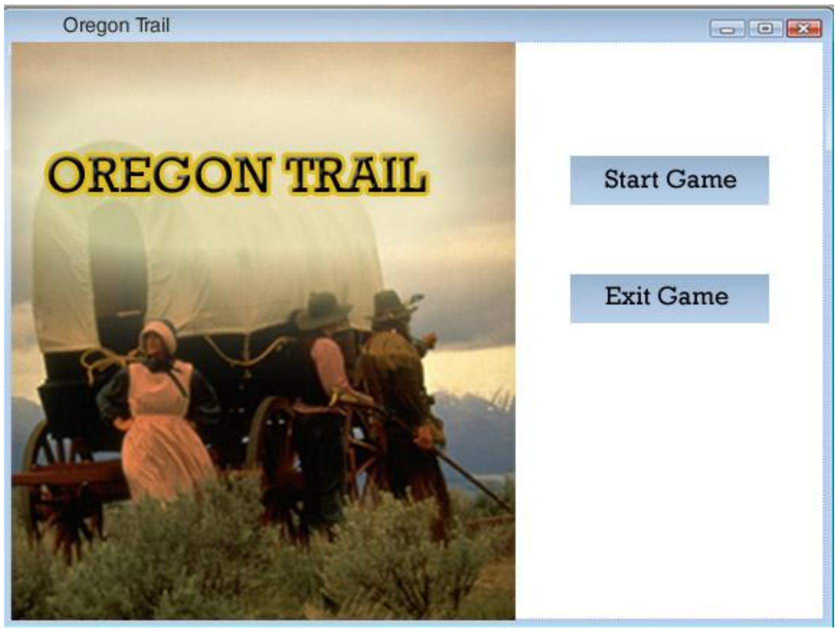
UML Sequence Diagrams



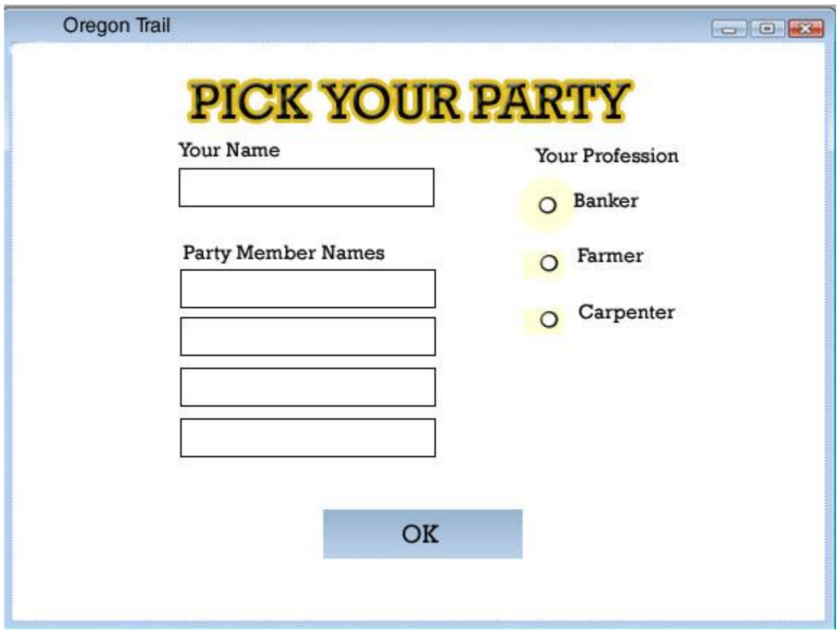

sd RandomEvents2
Player
InitializeGameAppModel
GamePlayAppModel
GamePlay
w:Wagon
Climate
RandomEvent
Trail
RandomEventAppModel
RiverCrossing
RiverCrossingAppMode
1: newGame
2: chooseProfession
3: buyInitialItems
4: startGame
5: startGame
6: makeNew
7: instantiate
8: instantiate
9: instantiate
10: open()
11: takeTurn
12: takeTurn
13: checkForRandomEvent
14: reference
15: updateWagon
16: updateWagonPosition
17: updateClimate
18: takeTurn
19: takeTurn
20: checkForRandomEvent
21: reference
22: brokenWagonWheel
23: openInstanceOf
24: replaceWagonWheel
25: removeWagonWheel
26: updateWagon
27: updateWagonPosition
28: updateClimate
29: takeTurn
30: takeTurn
31: checkForRandomEvent
32: reference
33: updateWagon
34: updateWagonPosition
35: RiverCrossing
36: openEvent
37: openInstance Of
38: caulkWagonAndFloatAcross
39: caulkedWagon
40: wagonFlipped
41: removeClothes
42: removeBullets
43: updateClimate

User Interface Prototypes

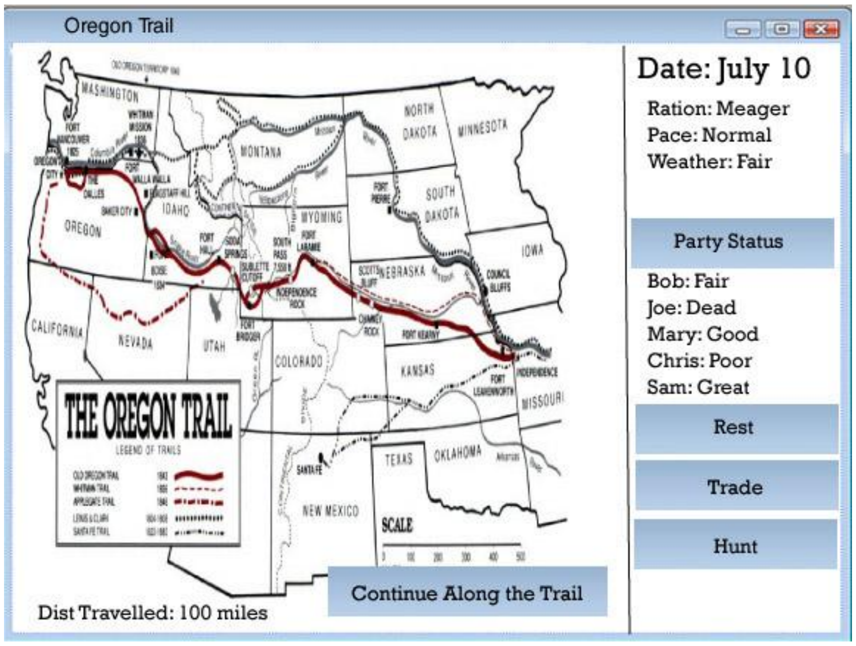
Start Screen



Select Party Screen



Map Screen



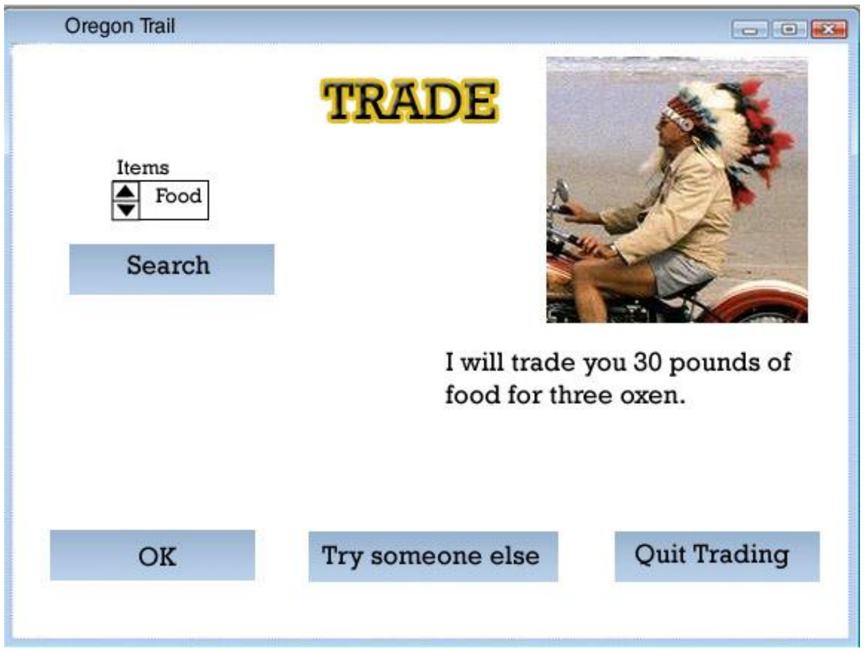
Date: July 10

Ration: Meager
Pace: Normal
Weather: Fair

Bob: Fair
Joe: Dead
Mary: Good
Chris: Poor
Sam: Great

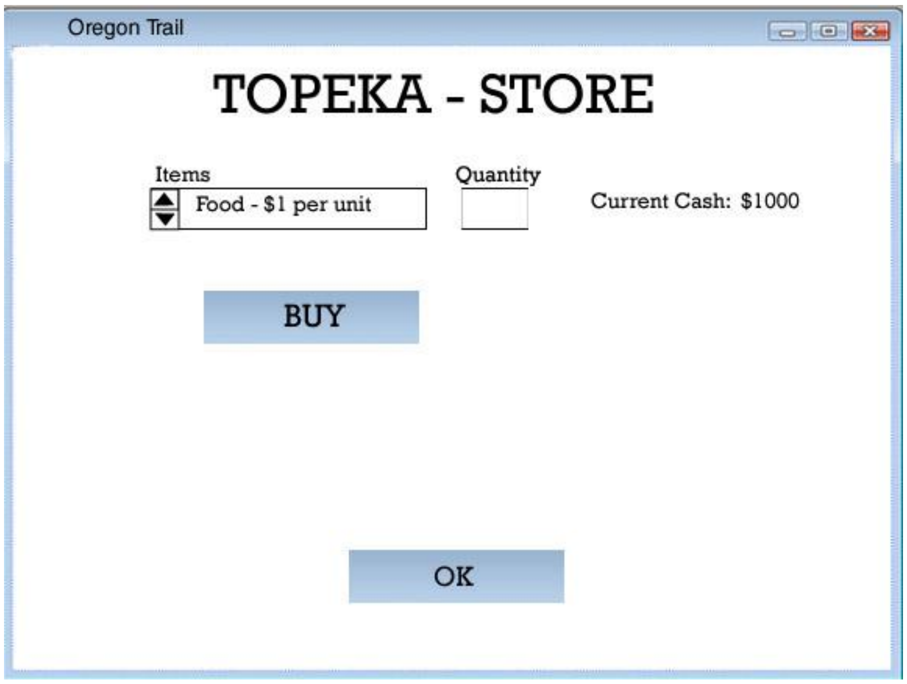
Dist Travelled: 100 miles

Trade Screen

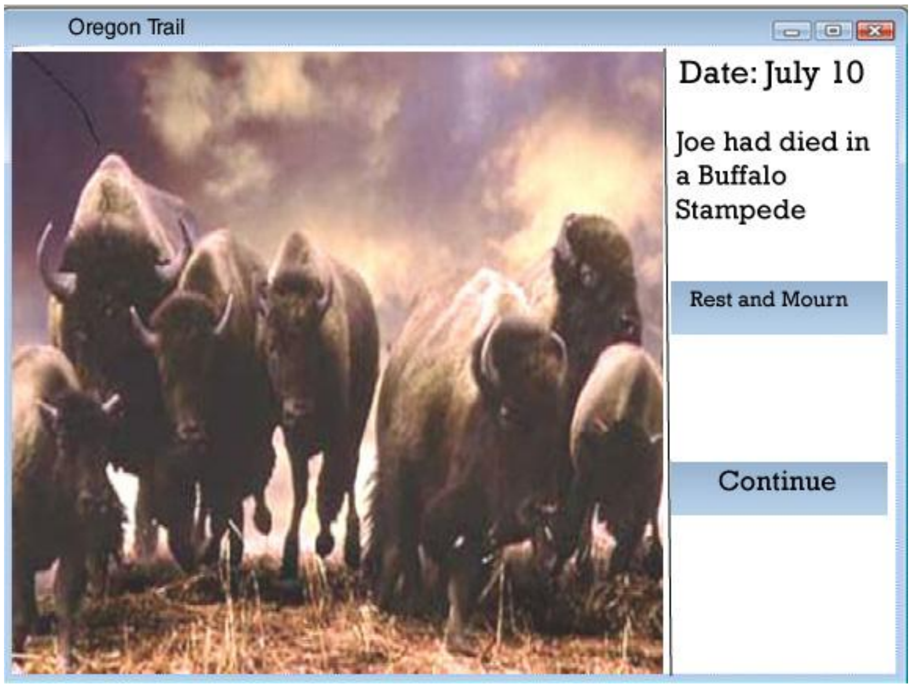


I will trade you 30 pounds of food for three oxen.

Store Screen



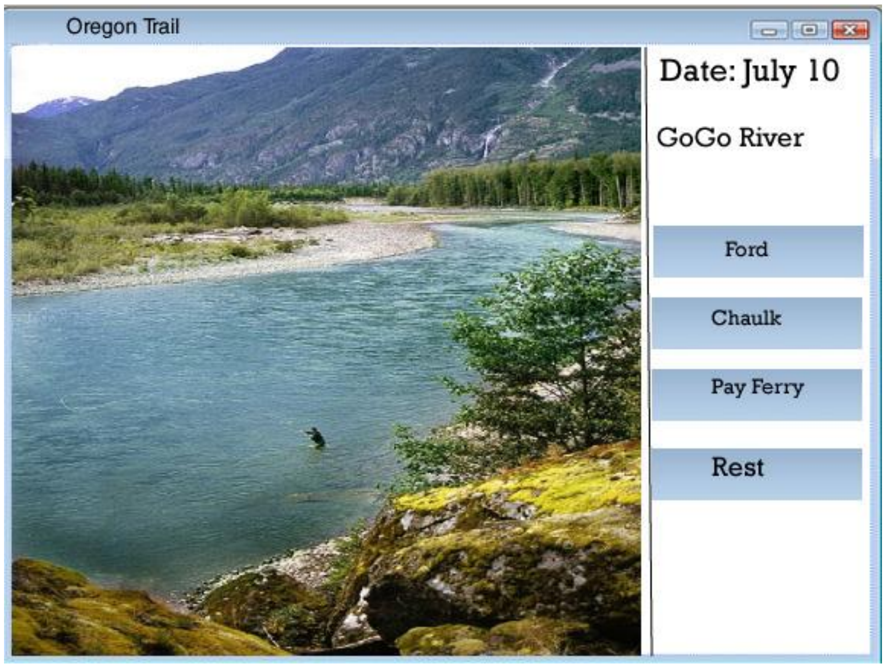
Random Event Screen



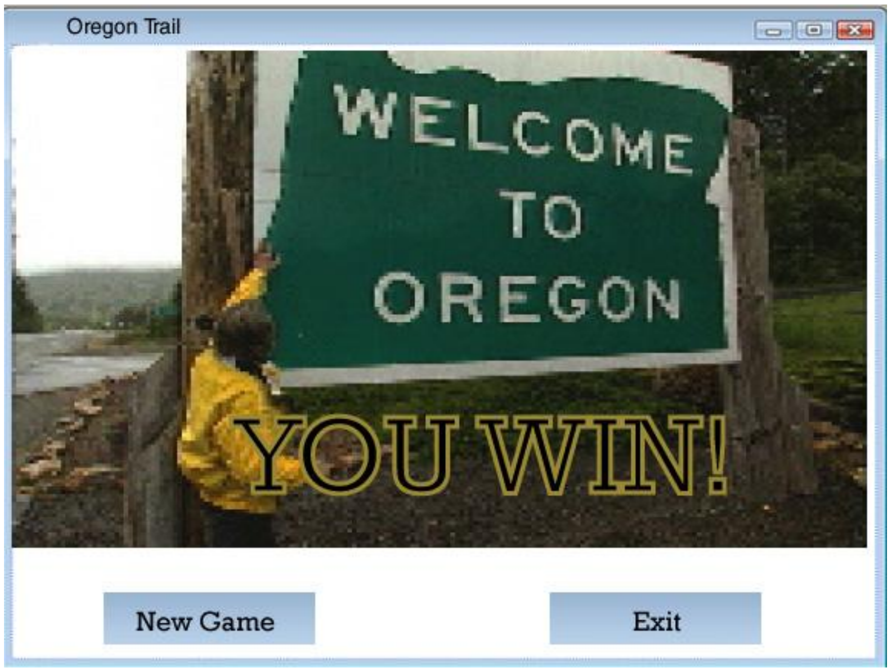
Date: July 10

Joe had died in a Buffalo Stampede

River Crossing Screen



Win Screen



Contracts

addItem	**Obligation**	**Benefits**
Client: Person	(Satisfy precondition) an instance of Wagon exists item is an OrderedCollection anItem is an instance of Item	Makes sure that the method is taking in valid operators in order to properly complete its function of adding items to the array.
Supplier (Server)	(Satisfy postcondition) item includes anItem Updates item array increases the quantity of the given item	Easier operation because we know that the array of items contains Item objects. The list of items is also kept up to date.

Landmark	**Obligation**	**Benefits**
Client: Person	(Satisfy precondition) Has a name that is a string.	The landmark only has to know which landmark it is and the rest of the checking is done by the pointOfInterest class.
Information Holder: pointOfInterest	(Satisfy postcondition) Updates the rest that the wagon got.	Makes sure that the status of the people in the wagon is updated.

Hunt	**Obligation**	**Benefits**
Client: Person	(Satisfy precondition) A wagon object exists Inventory of items exists	Allows the user to hunt and add food to their load and need to be able to access valid wagon as well as a valid item array.
Coordinator: Gameplay	(Satisfy postcondition) Updates the inventory collection, adding a valid Item object to it. Returns the chance the player	Makes sure that inventory is updated and the items in it are Item objects. Calculating the probability that the player will hit the game makes the game

Store	**Obligation**	**Benefits**
Client: Person	(Satisfy precondition) Contains items that are item objects in the store. Has a valid balance	Keeps the store's information accessible and valid so that the player is able to make purchases from the store.
Service Provider: Settlement	(Satisfy postcondition) Returns a valid item object if one is bought from the store. Updates the wagon's inventory and the wagon's balance.	Allows the player to make purchases from the store and then transfer them to their wagon. It also knows to adjust the balance in the wagon according to what

Exception Handling Strategies

If an error occurs in the application while we are building and testing it, we plan to use the 'balk' strategy by displaying an error dialog screen, followed by the termination of the application. For example, if the Wagon class tries to add a String to the OrderedCollection of Persons, we will output to the screen that an exception has occurred and where it occurred, then terminate the program. If exceptions occur after we have debugged it using the Balk and Terminate strategy, we will handle them by notifying the user (using a Dialog box) that the last thing they did caused an exception, roll back (undo) what they have done, and allow them to retry.