Oregon State University CS 162 Fall 2015

Name: Steven Lim

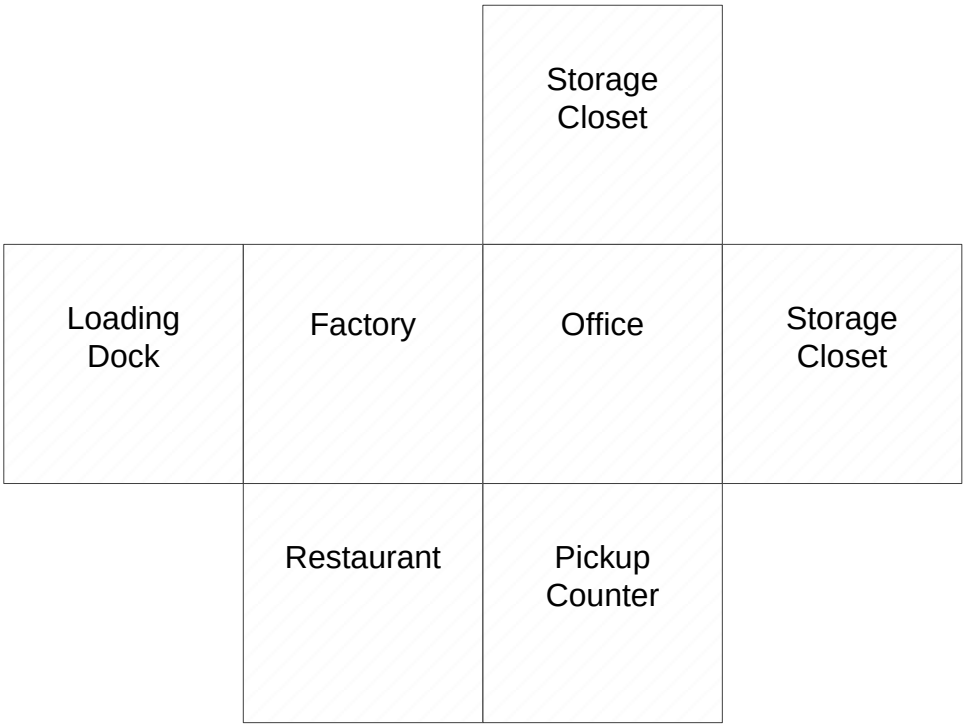
TA Name: Ian McQuoid

Date: 12/8/2015

Final Project Reflections

High Level Design of Pizza Business Game:

Map of the Pizza Business Building:



The player is the owner of a pizza business which operates a pizza restaurant, a pickup counter for pizza takeout orders, and a frozen pizza factory. There will be distinct customer feedback from each of the three groups of customers: the restaurant patrons, the takeout customers, and the frozen pizza retailers. When the game begins, each of the three customer groups are unhappy. The player must walk around the pizza business building to investigate and fix any problems with the business operations which are causing the customer groups to be unhappy. The goal of the game is to satisfy all three customer groups within a limited budget.

The player will be given an initial budget of $2000 at the start of the game. There are three groups of employees: the cooks, the factory workers, and the front-line staff (host, waiters, and pickup counter staff). Every group of employees is paid by the hour. The cook employee group costs $100 per hour. The factory employee group costs $200 per hour. The front-line employee group costs $50 per hour. Every two times the player walks through a door inside the

building, an hour of game time will pass and the player must pay her employees out of her budget. Sometimes when an hour of game time has elapsed a random amount of revenue from customer sales will be added to the budget. In addition, there are a number of business operational issues causing customer dissatisfaction which must be resolved. Some issues may be resolved with an item but other issues will require money from the budget. The player loses the game and the game is over if the budget goes down to zero dollars before each of the three customer groups is satisfied.

Quite often specific items will be required to investigate and fix any operational issues. The player will have a briefcase allowing her to carry a maximum of four items at any time. In addition, all items can only be transferred between two specific container types. For example, a comment card item can only be transferred from a mailbox container to the player's briefcase and from the player's briefcase to a recycle bin. No item can be "dropped on the floor."

Whenever the player walks into or interacts with a room, clues will be provided to help the player find and resolve any issues with business operations. The player should use the clues to interact with the elements of the room and choose the appropriate actions to take. For example, the Storage Closet will always be dark when the player walks into the room. The player will be told that the Storage Closet is dark when she walks in. The player must then decide to turn on the light switch to light up the closet. The player will be barred from doing anything in the Storage Closet until the room is lit up. The player is also encouraged to get customer feedback obtained from various rooms to check on her game progress.

Another element of the game is employee pay raises. Sometimes it is necessary to raise the pay of one or more groups of employees to achieve customer satisfaction. The size of the pay raises shall be 50% of the old pay rate for each employee group. However, one game hour after the player raises the pay of at least one employee group, the employee groups which did not get pay raises will demand to receive a pay raise within six game hours or else they will go out on strike. If any employee group goes out on strike, then the game is over and the player loses the game. Therefore, the player should either give pay raises to all employees or try to satisfy all customer groups before any employee goes out on strike.

Room List: Loading Dock, Factory, Storage Closet (2), Office, Restaurant, Pickup Counter

Item List: Screwdriver, Wrench, Hammer, Magnifying Glass, Knife & Fork, Comment Card (3), Factory Document, Cook Document, and Front-line Document

Item Container List: Toolbox, Mailbox, Recycle Bin, and Player's Briefcase

Action, Clue, and Solution List:

<u>Loading Dock:</u>

Action #1: Inspect Delivery Trucks

Possible Clues: Delivery truck interiors feel warm – should be freezing; delivery trucks are good

Solution: Upgrade delivery trucks (Cost: $100)

Action #2: Inspect Ingredients

Required Item: Magnifying Glass

Possible Clues: Spoiled meat; spoiled vegetables; meat and vegetables are good

Solution: Order new meat ingredients (Cost: $100); order new vegetable ingredients (Cost: $100)

Action #3: Use Computer

The player will be presented a menu on the Computer and prompted to choose from the following options: upgrade the delivery trucks, order new meat ingredients, or order new vegetable ingredients

Factory:

Action #1: Read Online Customer Feedback from Frozen Pizza Retailers on the Computer

Possible Clues: Quantity of frozen pizza output is below normal; quality of frozen pizza output is below normal; praise for your business and pledges to order again

Action #2: Inspect/Fix Flash Freezer Machines

Required Item to Fix Machines: Wrench

Possible Clues: Quantity of frozen pizza output is below normal; quantity is fine

Solution: Apply Wrench item to Machines (Cost: $0)

Action #3: Monitor or Meet With Factory Employees

Possible Clues: Quality of the frozen pizza output is below normal; quality is fine

Solution: Raise pay of factory employees. The player must present a Factory Document item from the Payroll Department to the factory employees to prove that the pay raise was implemented. (Cost: $300 per game hour)

Storage Closet (contains Toolbox):

Action #1: Walk into room

Possible Clues: Room is dark and cannot open Toolbox

Solution: Turn On Light Switch (Cost: $0)

Action #2: Open Toolbox

Possible Clues: Toolbox may contain Screwdriver, Wrench, Hammer, Magnifying Glass, and/or Knife & Fork

Requirement: All tool items can only be placed into the Briefcase or into a Toolbox

Office (contains Recycle Bin):

Action #1: Payroll Department

Possible Clues: The player can choose from the following options: Pay raise for cook employees, pay raise for factory employees, and pay raise for front-line employees

If the player's Briefcase has space for the Document items, then the player will receive a Factory Document after raising the pay of factory employees, a Cook Document after raising the pay of cook employees, and a Front-line Document after raising the pay of front-line employees.

Requirement: A pay raise for an employee group can only be implemented if the player Briefcase has room to receive the Document item which is relevant to the employee group. For example, a pay raise for cook employees can only be implemented if the player's Briefcase can receive the Cook Document item. Only one pay raise for each employee group can be implemented per game. However, the player can always request a duplicate Document item from the Payroll Department for a pay raise that is already implemented.

Action #2: Open Recycle Bin

Possible Clues: Some Comment Cards and Documents can be seen in the Recycle Bin

Requirement: Comment Card and Document items can only be placed into the Briefcase or into the Recycle Bin

<u>Restaurant:</u>

Action #1: Review Restaurant Pizza

Required Item: Knife & Fork

Possible Clues: Chewy pizza crust when the crust should be crispy; crust has good crispy texture

Solution: Raise pay of cook employees (must go to Payroll Department; Cost: $150 per game hour) AND upgrade the pizza oven (Cost: $0)

Action #2: Inspect/Upgrade Pizza Oven

Required Item to Upgrade Oven: Screwdriver

Solution: Upgrading the oven will improve the restaurant pizzas (Cost: $0)

Action #3: Inspect/Upgrade Dining Area

Required Item to Upgrade Dining Area: Hammer

Solution: Upgrading the dining area may improve customer satisfaction (Cost: $0)

Action #4: Chat with Patrons

Possible Clues: Complaints about dining area décor; complaints about the front-line service; complaints about the pizza crust; love for the pizza, praise for the restaurant, and pledges to return again

Solutions: Upgrade the dining area (Cost: $0), raise pay of front-line employees (must go to Payroll Department; Cost: $75 per game hour), raise pay of cook employees (must go to Payroll Department; Cost: $150 per game hour) and/or upgrade pizza oven (Cost: $0)

Action #5: Monitor or Meet With Cook Employees

The player must present a Cook Document from the Payroll Department to the cook employees to prove that the pay raise was implemented.

Action #6: Monitor or Meet With Front-line Employees

The player must present a Front-line Document from the Payroll Department to the front-line employees to prove that the pay raise was implemented.

Pickup Counter (contains Mailbox):

At the start of the game or immediately after the player resolves a takeout customer issue based on the old Comment Card, a new Comment Card item is "generated" by the Mailbox. If all the takeout customer issues are resolved, then the happy customer Comment Card will be the last Comment Card generated by the Mailbox for the duration of the game. A maximum of one Comment Card item can be present in the Mailbox at any time. Therefore, if necessary, a new Comment Card item generated by the Mailbox will "overwrite" an older Comment Card item already in the Mailbox.

Action #1: Open Mailbox

Possible Clues: Mailbox contains a Comment Card item or may be empty

Requirement: The Comment Card item must be inside the Briefcase before it can be read. A Comment Card item can be transferred from the Mailbox to the Briefcase but it cannot be transferred from the Briefcase to the Mailbox. To discard the Comment Card item, the item must be placed into the Recycle Bin in the Office room.

Only three distinct Comment Cards exist in the game:

Comment Card #1: Customer complains about lack of sales promotions. Other competing pizza businesses throw in free side dishes or beverages with a pizza purchase.

Comment Card #2: Customer complains about the lack of a customer loyalty program. Other competing pizza businesses have customer loyalty programs.

Comment Card #3: Customer loves the pizza, praises the player's business, and pledges to order again.

Action #2: Implement a New Takeout Sales Promotion

Possible Clues: Read various Comment Card items

Solution: The player can implement a takeout sales promotion (free soda with pizza takeout purchase; Cost: $100), implement a takeout customer loyalty program (purchase five takeout pizzas and get one free; Cost: $100), or implement both sales promotions

Player's Briefcase (Item container):

The player may view the contents of her Briefcase at any time during the game. She may view the contents of the Comment Card items in her Briefcase.

Action to Check on Restaurant Patrons: Chat with Patrons in the Restaurant room

Action to Check on Takeout Customers: Read Comment Card items which originate from the Mailbox in the Pickup Counter room

Action to Check on Frozen Pizza Retailers: Obtain online customer feedback from the Computer in the Factory room

Design Description:

The program shall consist of three distinct abstract base class hierarchies and a group of game functions to control the overall game. The three abstract base classes are: Room, Item, and Container.

The Room class hierarchy defines the various rooms of the linked structure. Each room has four Room pointers to represent the "doors" to other rooms. Each room has a Container pointer to represent an item container in the room such as a Mailbox or a Toolbox. Each room has a global bool called roomState to represent the complete resolution of business operational issues local to the room as well as local bools specific to its room type. The local bools represent the status of potential business issues in the room. If an unresolved issue exists in the room, then the corresponding local bool is set to false. The display of clues to the player also depends on local bool values of each room. When the player solves a problem that was present in a room, one of the local bool variables of the room will change from false to true. The global roomState bool variable is re-evaluated as the result of ANDing all the local bool variables together. Therefore, when all of the local bool variables of a room are set to true, then the roomState bool is set to true. Then the game functions can determine that the player has resolved all the problems of that room by checking if the roomState bool is true. The game ends and the player wins when the roomState bool of every room is set to true. Each room also has a static array of three bool variables called payRaiseState. The game functions use the array to determine whether pay raises for the different employee groups have been implemented. This is used to determine whether or not the employees will go out on strike and the game ends. Each room will have pure virtual methods to allow the player to perform actions in the room such as printActionList() and special(). Each room will have additional methods to allow the player to move from one room to another called printNavigationList() and getNextRoomLocation(). Finally, each room will have status methods to allow the game functions to determine the state of each room called getRoomState(), getCookRaiseState(), getFactoryRaiseState(), and getFrontRaiseState(). The Room subclass constructors also take in an integer parameter representing the game difficulty level. At the highest difficulty setting, the room state is set to

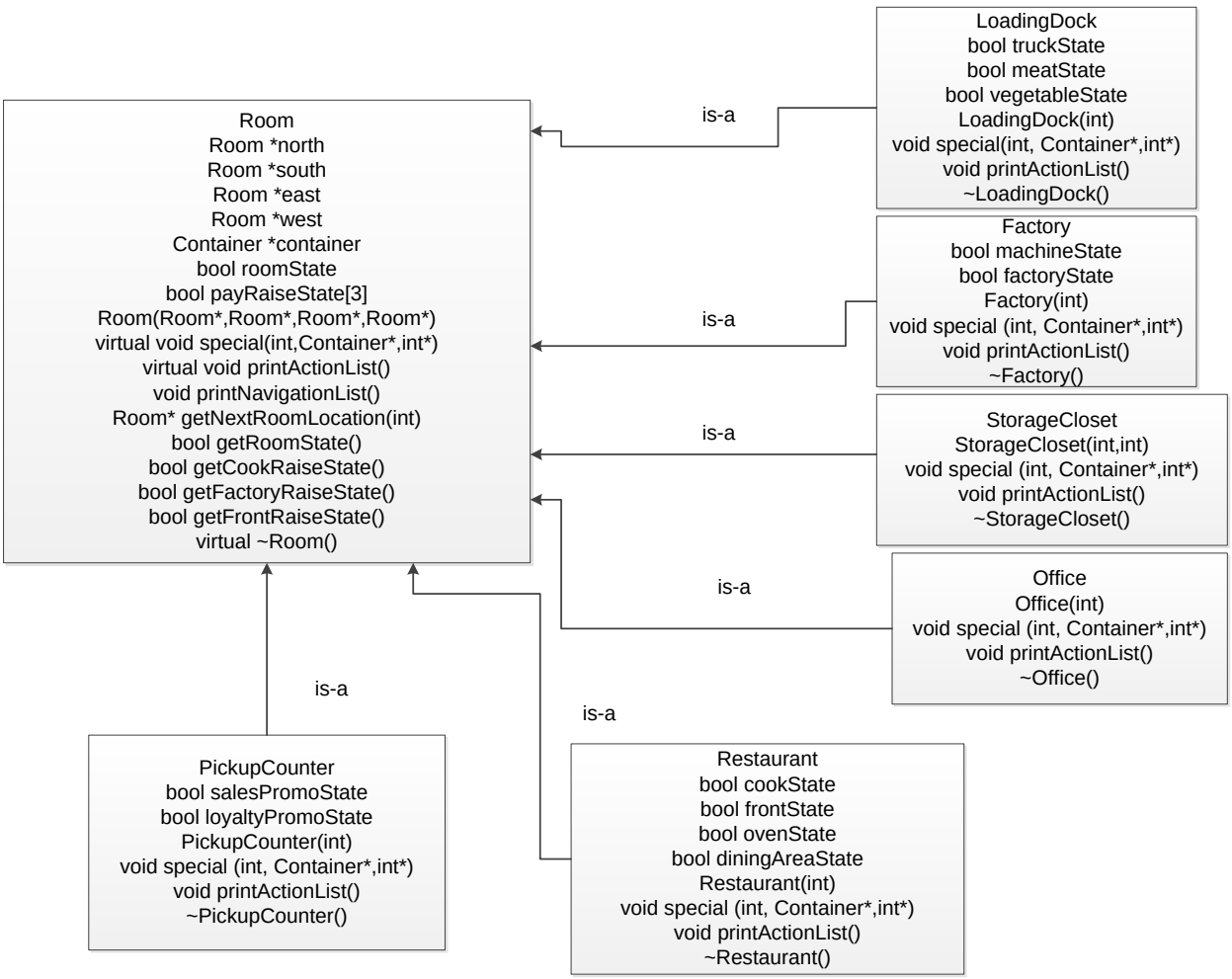
The Item class hierarchy defines the various items in the game. Each item has an integer to represent the type of item it is and an integer to represent the type of container the item is inside of. The container type is used to enforce the rule that Comment Card contents can only be read if the Comment Card item is inside the Briefcase container. There are accessor and mutator methods for container type and an accessor method for item type. There is also a pure virtual method called viewItem() to display the item's attributes and contents to the user.

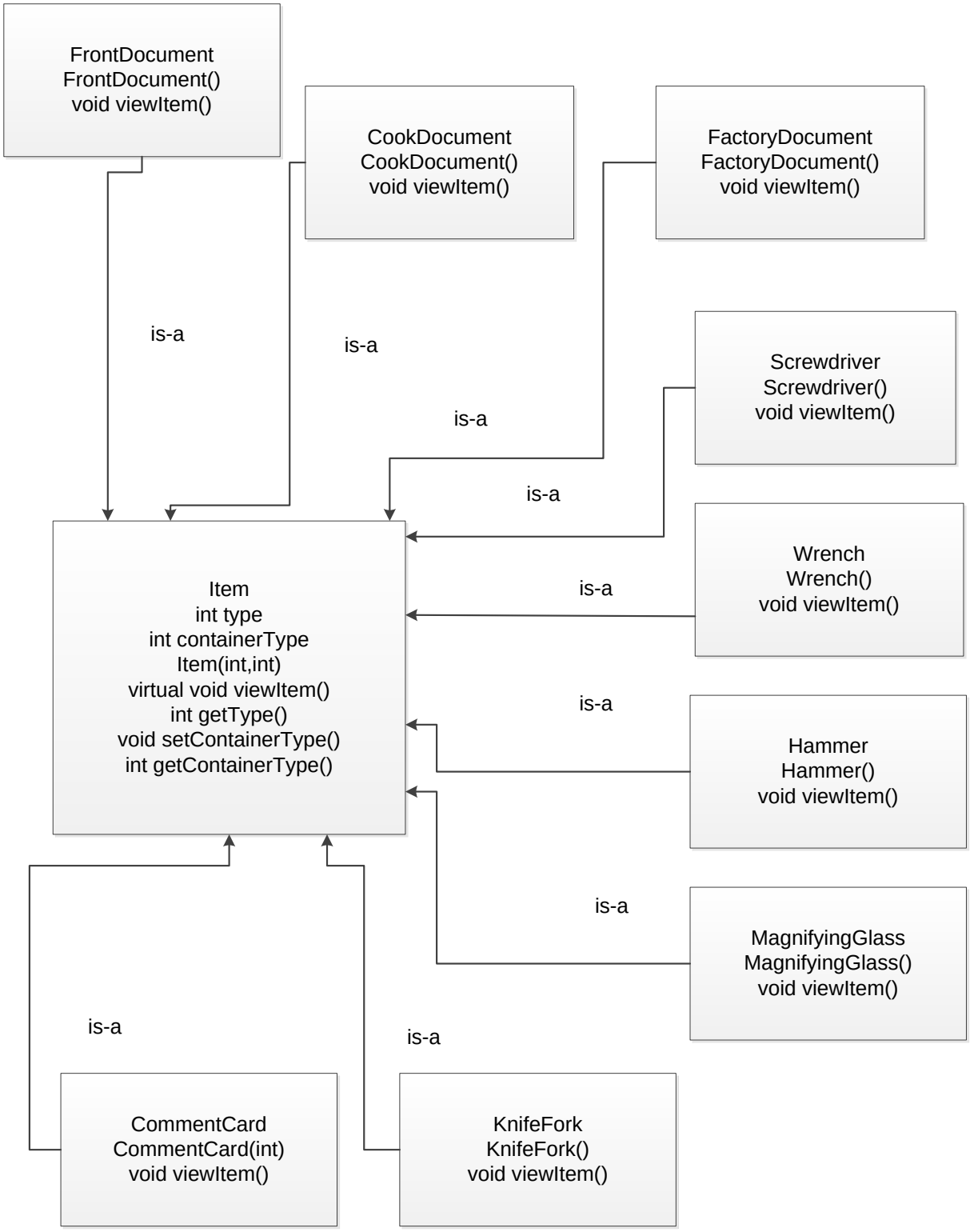
The Container class hierarchy defines the various item containers in the game. Each container has an integer to represent the type of container it is and accessor method to get the container type. Each container is essentially an STL vector of Item pointers. The viewContents() method calls the viewItem() method for each item in the container. The

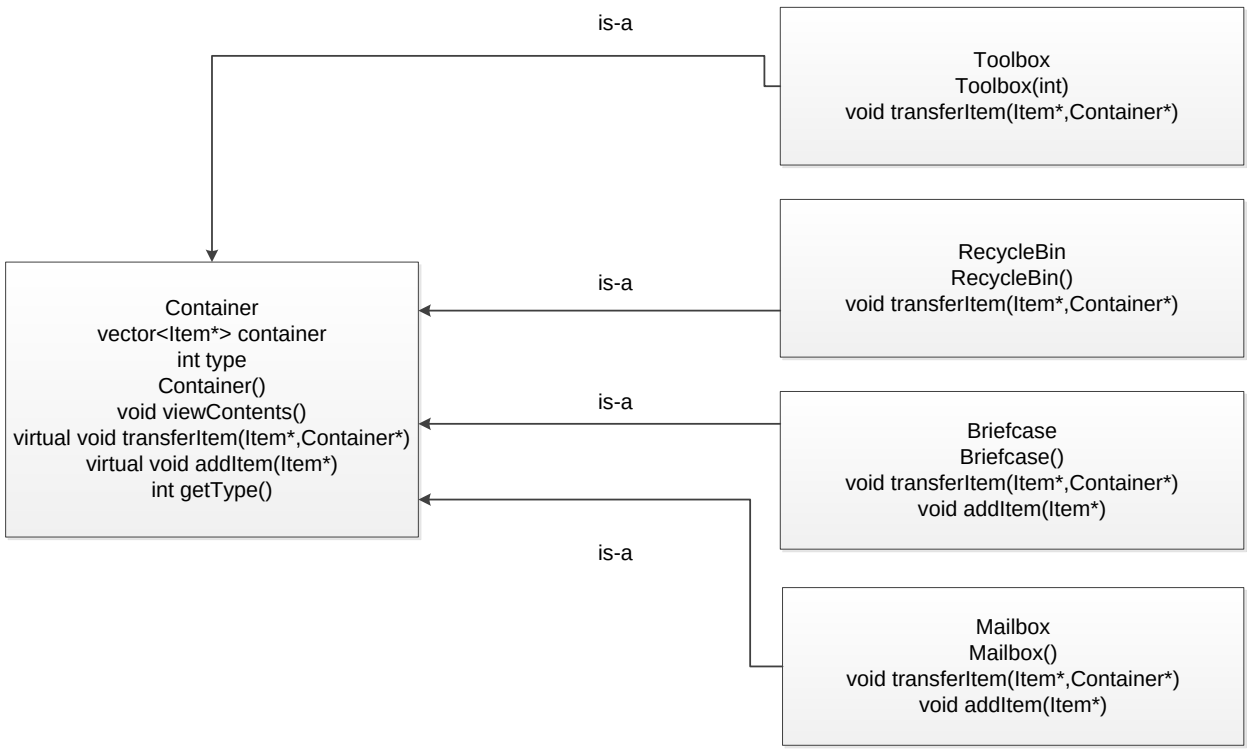
transferItem() method is used to transfer an item from one container to another container. The addItem() method is used to spawn an item inside of a container. The ability to spawn items is necessary to initialize the game parameters as well as to create some items out of thin air during the game.

The group of game functions will initialize the game by calling the Room constructors and will determine when the game ends by analyzing status information from all the rooms and keeping track of the player's budget with an integer. The game functions will also declare whether or not the player wins the game. The game functions keep track of the player's location with a Room pointer and will provide the Briefcase object to the Room objects using a Briefcase pointer. The game will also provide the address of the budget variable to the Room objects so that various room actions can subtract money from the budget. The game will also count the number of player movements between any two rooms in order to advance the game clock and subtract money from the budget accordingly. Upon each advancement of the game clock, the budget may be increased by a random amount to represent customer sales revenue. An integer will be used to represent the game clock in units of hours. All customer groups are considered to be happy if the roomState bool variable for every room is set to true. Therefore a bool variable called customersHappy will be evaluated as the result of ANDing the roomState bool variables of all the rooms. The game will also keep track of the pay raise status of the three employee groups using bool variables. If all three bool variables have the same value, then no employee strike will occur. Otherwise, a strike will occur after six game hours. When any group of employees go out on strike, the game immediately ends and the player loses the game. If at least one customer group is unhappy and the budget is zero or less, then the game ends and the player loses the game. If all three customer groups are happy, the budget is non-zero, and no employee strike occurs, then the game ends and the player wins the game.

Class Hierarchy Diagrams (starting on next page):


Room
Room *north
Room *south
Room *east
Room *west
Container *container
bool roomState
bool payRaiseState[3]
Room(Room*,Room*,Room*,Room*)
virtual void special(int,Container*,int*)
virtual void printActionList()
void printNavigationList()
Room* getNextRoomLocation(int)
bool getRoomState()
bool getCookRaiseState()
bool getFactoryRaiseState()
bool getFrontRaiseState()
virtual ~Room()
is-a
LoadingDock
bool truckState
bool meatState
bool vegetableState
LoadingDock(int)
void special(int, Container*,int*)
void printActionList()
~LoadingDock()
is-a
Factory
bool machineState
bool factoryState
Factory(int)
void special (int, Container*,int*)
void printActionList()
~Factory()
is-a
StorageCloset
StorageCloset(int,int)
void special (int, Container*,int*)
void printActionList()
~StorageCloset()
is-a
Office
Office(int)
void special (int, Container*,int*)
void printActionList()
~Office()
is-a
PickupCounter
bool salesPromoState
bool loyaltyPromoState
PickupCounter(int)
void special (int, Container*,int*)
void printActionList()
~PickupCounter()
is-a
Restaurant
bool cookState
bool frontState
bool ovenState
bool diningAreaState
Restaurant(int)
void special (int, Container*,int*)
void printActionList()
~Restaurant()


FrontDocument
FrontDocument()
void viewItem()
CookDocument
CookDocument()
void viewItem()
FactoryDocument
FactoryDocument()
void viewItem()
is-a
is-a
is-a
Screwdriver
Screwdriver()
void viewItem()
is-a
Item
int type
int containerType
Item(int,int)
virtual void viewItem()
int getType()
void setContainerType()
int getContainerType()
Wrench
Wrench()
void viewItem()
is-a
is-a
Hammer
Hammer()
void viewItem()
is-a
MagnifyingGlass
MagnifyingGlass()
void viewItem()
is-a
is-a
CommentCard
CommentCard(int)
void viewItem()
KnifeFork
KnifeFork()
void viewItem()



Test Plan:

Test Case	**Expected Result**
Play the game on hard difficulty	The player is more likely to lose the game.
Play the game on medium difficulty	The player is equally likely to win the game or lose the game.
Play the game on easy difficulty	The player is more likely to win the game.
Unit Test: The player moves between rooms two times.	Game clock advances by one hour. The budget decreases by the proper amount to simulate the pay rate for each employee group. The budget increases by a random amount to simulate incoming customer sales revenue. Under special circumstances, the player should be warned of an impending employee strike. If the budget goes to zero or less before the customers are happy or an employee strike occurs, then the game should end.
Unit Test: The player moves back and forth between two rooms.	The menu of room actions and navigation options should be as expected for each room that the player moves to. In other words, the moving back and forth between two rooms shouldn't cause the player to move to a third distinct room. Moving to a new room should not cause the bool state of the old room to change.
Unit Test: Test player navigation throughout the business building	The player navigation throughout the entire building should be consistent for the duration of the game. In other words, if the same navigational path from the

	same starting point in the building is taken, then the end point result should always be the same.
Unit Test: Special actions are taken by the player for each room.	The state of each room should be modified properly in response to the player's actions. For example, if the player upgrades the delivery trucks, then the delivery truck status of the loading dock should change from false to true for the duration of the game. The player's budget should be decreased by the proper amount. Any pay raise actions should be reflected in the budget and possibly trigger an employee strike warning.
Unit Test: Add and remove items from the Briefcase	Items are properly transferred to and from the Briefcase based on established type-based rules for Items and Containers.
Unit Test: Attempt to add an item when the Briefcase already has four items	Briefcase should reject the item.
Unit Test: Read Comment Card contents	Comment Card contents should be displayed only when the Card item is in the Briefcase container.
Unit Test: Resolve a takeout customer issue	Mailbox properly spawns a new Comment Card when a takeout customer issue is resolved
Unit Test: Attempt to put non-tool and tool items in the Toolbox	The Toolbox should take tool items and reject non-tool items
Unit Test: Attempt to put any item in the Mailbox	The Mailbox should reject any item.
Unit Test: Attempt to put paper and non-paper items in the Recycle Bin	The Recycle Bin should take paper items and reject non-paper items
Unit Test: Implement a pay raise at Payroll Department in Office room	Briefcase should receive the relevant Document item if the Briefcase has space for it. Otherwise, the pay raise implementation should be rejected.
Unit Test: Implement a factory employee pay raise at Payroll Department in Office room	Briefcase should receive the Factory Document item and not another type of Document item.
Unit Test: Resolve a business operational issue	The clues related to the resolved issue should change to a positive note. For example, if the clue stated that the frozen pizza delivery truck interior used to be warm, then after resolution of the issue, the clue should state that the truck interior is freezing cold.
Unit Test: Attempt to upgrade the pizza oven when the oven is in a bad state	If the Screwdriver item was used, then the pizza oven state should be good. If any other item was used, then the pizza oven state should remain bad.

Test Results:

Test Case	**Actual Result**
Play the game on hard difficulty	The player is more likely to lose the game unless the player knows the exact sequence of steps to resolve all of the business operational issues. All possible

	business operational issues are enabled at the start of the game and they must be resolved with a limited budget.
Play the game on medium difficulty	The player is equally likely to win the game or lose the game unless the player knows the exact sequence of steps to resolve all of the business operational issues. Most of the possible business operational issues are enabled at the start of the game and they must be resolved with a limited budget.
Play the game on easy difficulty	The player is more likely to win the game. This game mode has the fewest business operational issues enabled at the start of the game and the budget provides more than enough cushion to win the game.
Unit Test: The player moves between rooms two times.	Game clock properly advanced by one hour. The budget decreased by the proper amount to simulate the pay rate for each employee group. The budget does not increase by a random amount because of a design change. The employee strike feature was removed due to a design change. If the budget goes to zero or less before the customers are happy, then the game does end and the player is properly notified that she lost the game.
Unit Test: The player moves back and forth between two rooms.	A segmentation fault related to improper initialization of the Room pointers did occur during unit testing. The issue caused navigation between rooms to fail. After the fix, navigation between rooms worked as expected.
Unit Test: Test player navigation throughout the business building	The player was able to navigate throughout the entire building in a consistent manner for the duration of the game. The building layout does not somehow change based on the current room location of the player.
Unit Test: Special actions are taken by the player for each room.	The state of each room is modified properly in response to the player's actions. The budget and room clues were updated properly. Any pay raise actions were properly reflected in the budget but due to a design change, the employee strike warning feature was removed.
Unit Test: Add to and remove items from the Briefcase	A segmentation fault occurred during unit testing. After various fixes, Items were properly transferred to and from the Briefcase based on established type-based rules for Items and Containers. All of the items fall under two possible types: paper and tool.
Unit Test: Attempt to add an item when the Briefcase already has four items	Briefcase properly rejected the item.
Unit Test: Read Comment Card contents	Comment Card contents are displayed only when the Card item is in the Briefcase container.
Unit Test: Resolve a takeout customer issue	Some bugs with the Mailbox did occur during unit testing. After some fixes, the Mailbox properly spawned a new Comment Card when a takeout customer issue is resolved. The Mailbox properly overwrites the old Comment Card with a new

	Comment Card if the Mailbox is full when a customer takeout issue is resolved.
Unit Test: Attempt to put non-tool and tool items in the Toolbox	The Toolbox only takes tool items and rejects non-tool items.
Unit Test: Attempt to put any item in the Mailbox	The game is designed such that the player cannot even attempt to add an Item to the Mailbox.
Unit Test: Attempt to put paper and non-paper items in the Recycle Bin	The Recycle Bin properly takes paper items and rejects non-paper items.
Unit Test: Implement a pay raise at Payroll Department in Office room	Briefcase received the relevant Document item when the Briefcase had space for it. Otherwise, the Briefcase does not take the Document item. The game is designed to implement the pay raise when the employees are presented with the proper Document item.
Unit Test: Implement a factory employee pay raise at Payroll Department in Office room	Briefcase received the Factory Document item and not another type of Document item.
Unit Test: Resolve a business operational issue	The clues related to the resolved issue did change to a positive comment from the customer or an expected room state change.
Unit Test: Attempt to upgrade the pizza oven when the oven is in a bad state	When the Screwdriver item was used, then the pizza oven state was changed to a good state. If any other item was used, then the pizza oven upgrade was rejected.

Reflections:

Design Changes:

- Add string variable called name and getRoomName method to Room class to display the name of the room to the player
- Add useItem method to Container class to get Item pointer from a container without removing it from the container (allow player to “use” an item on a room element)
- Add lightState bool variable to the StorageCloset class and designed the Storage Closet room to always keep the light on for the duration of the game after the light switch is turned on
- Add a parameter list to the Container constructor to set the container type
- Add removeItem method to Container class to remove an Item pointer from a container (used to transfer items between two containers)
- Add a setNeighbors method to Room class which would set the Room pointer values of a Room object (needed to fix bug)
- Remove employee strike feature from the game design
- Remove revenue feature (random increases in budget during the game) from the game design

Bugs:

- Segmentation fault when attempting to print the room navigation list (fix was to add a setNeighbors method to the Room class, since originally the Room constructor initialized the Room pointer values which was leading to garbage values)
- Segmentation fault when dealing with the Recycle Bin container in the Office room (forgot to instantiate the Recycle Bin object in the Office constructor)
- Segmentation fault for out of bounds access error when trying to remove an Item from an empty Container (logic error related to a check of the container size in the useItem method)
- Game did not end in a player win after all the business operational issues were resolved (forgot to set the roomState bool to true in the Office room constructor since the Office room has no issues to be resolved and by default the Room constructor sets the roomState bool to false)
- The two instantiations of the StorageCloset class had the same room name
- The CommentCard item could not be properly retrieved from the Mailbox container (typo in the transferItem method call between Briefcase and Mailbox)