**Lemonade – Deliverables / Time Table**

* ~~Define Rules based on features~~
* ~~Determine game flow based on rules~~
  + ~~Welcome~~
  + ~~Name~~
  + ~~Start money~~
  + ~~Etc…~~
* ~~Break down game into sections~~
  + ~~Welcome window~~
  + ~~Determine funds~~
  + ~~Make decisions~~
  + ~~Etc…~~
* ~~Determine classes~~
* ~~Determine functions~~
  + ~~Place functions~~
* ~~member variables needed~~
* algorithms
* fill in functions
* Create Game Loop
* Test

**Game Rules**

* Game will last for a total of 7 days
* Each player starts out with a random number based on Randomized number based on their difficultly level.
  + Easy
  + Medium
  + Hard
* Each player decides regarding how much inventory they want to purchase.
  + Lemons
  + Sugar
  + Ice
  + Cups
* Each player decides on their recipe used at the beginning of the day.
  + Lemons
  + Sugar
  + Ice
  + Cups
* Each player will decide at the beginning of the day regarding how much lemonade they will make.
  + Each pitcher contains 10 cups of lemonade
* Each player will sell lemonade based on the factors such as.
  + Weather
  + Customer Probability
  + Price Point
  + Thirst
* Each player will have a limit on the total items that can be purchased daily.
* Each player must maintain a bank account with a balance over $.01.
* Each player will have a daily inventory count.
  + Inventory will spoil????
* Revenue will be maintained daily and customers will be able to see Net Income sales numbers.
* Revenue will be maintained throughout the game and customers will be able to see Net Income and sales numbers.
* Weather will change daily.
* Each player will can go bankrupt to end the game and start over.

**Game Flow**

* Welcome Screen is Displayed.
* Rules will be displayed.
* Players will be given the option to input their name
* Player selects their level mode (Easy – Medium – Hard)
  + Random Dollar amount is generated and placed in their bank account.
* Player selects the total number of days they want to play
* Weather generates into a list – ***NEEDED FOR ALGORITHM TO DETERMINE PROBABLY OF PURCHASE***
* Customers generated on each day based on weather

Game Loop

* Players will be given the option to go to the store.
* Players will be given the option to buy items.
  + Lemons
    - List Created for each instance of the Objects
  + Cups
    - List Created for each instance of the Objects
  + Ice
    - List Created for each instance of the Objects
  + Sugar
    - List Created for each instance of the Objects
* Price for each item will be determined and players confirm purchase.
* Inventory will be displayed.
* Players determine number of pitchers to be made.
  + Players determine recipe using ice, sugar, lemonade. ***- NEEDED FOR ALGORITHM TO DETERMINE PROBABLY OF PURCHASE***
* Player Inventory is adjusted based on decisions
* Players determine price point for each cup.
* Players try to sell lemonade
* A Total number of Customers for the day randomly generates and the probability of purchase varies on multiple factors
  + Weather
  + Customer Probability
  + Price Point
  + Thirst
* Customers purchase lemonade
* Player Inventory (Cups) is adjusted based on purchases
* Sales numbers are calculated.
* Transaction numbers are generated and displayed for the day
  + Net Profit/Less
  + Total Sales
* Transaction numbers are generated and displayed for the entire life of the lemonade stand
  + Net Profit/Less
  + Total Sales
* Inventory numbers are calculated and displayed
* Players given option to continue playing
* Loop back to the store

**Game Sections**

**Options Menu– Fourth Completed Dec 8th**

* Welcome Screen
* Rules
* Player Selection/Name
* Level Mode
* Go to the Store
* Store Option Menu
* Continue to Play / Bankrupt option
* Game Loop

**Inventory Management– First Completed Dec 5th**

* ~~Item Purchasing~~
* ~~Calculation of inventory after store~~
* ~~Display Inventory~~
* Calculation of daily inventory after making lemonade
* Calculation of inventory after selling lemonade/using cups

**Lemonade Stand Operation – Second Completed Dec 6th**

* Determine number of pitchers
* Price Point determination
* **Customer Generation**
  + Algorithm for probabilities
    - For Generating customers
    - For generating likely hood of purchase
* Lemonade purchase

**Sales Tracking – Third Completed Dec 7th**

* Sales number calculation
* Transaction numbers are generated and displayed (Daily)
* Transaction numbers are generated and displayed (Lifetime)

**Debug/Testing – Last Completed Dec 9th**

**Classes**

**Cups Of Lemonade – [[1]](#endnote-1)**

* Class Responsibilities
  + **Function:** Create **Sale Price** will be associated
  + **Function:** **Sale Price** added to **player bank**

**Customers –[[2]](#endnote-2)**

* Class Responsibilities
  + **Function:** Generated Customer based on weather list
  + **Function:** Generated ***playerDisposition*** for purchase

**Day –[[3]](#endnote-3)**

* Class Responsibilities
  + **Function:** Provides player with a choice for game days played.

**Game –**

* Class Responsibilities
  + **Function:** Run the Game Loop

**Inventory –**

* Class Responsibilities
  + **Function:** Calculation of Inventory (Daily)
    - Uses the **List of Lemons**
    - Uses the **List** of Cups
    - Uses the **List of Ice**
    - Uses the **List of S­ugar**
  + **Function:** Display Inventory (Daily)

**CupsofLemonade –**

* Class Responsibilities
  + Create **List of Cups**
  + **Function:** Choose number of cups to make
  + **Function:** Select Item for Recipe
    - Lemons
    - Cups
    - Ice
    - Sugar
  + **Function:** Calculation of Inventory used for Recipe

**Player – (Parent Abstract Class)**

* Class Responsibilities
  + **Function:** **Name** the Players

**Human Player – Child Class**

* Class Responsibilities
  + **From Parent Class - Function:** **Name** the Players
  + **Function:** Display **player bank**

**Program –**

* Class Responsibilities

**Lemonade Sales– Disp**

* Class Responsibilities
  + **Function:** Selection of “Item” for Purchase – using same function as store
  + **Function:** Calculate Sales (Daily)
  + **Function:** Calculate “Sales List” (Daily)
  + **Function:** Display “Sales List” (Daily)
  + **Function:** Display “Sales List” (Lifetime)

**Store –**

* Class Responsibilities
  + **~~Function:~~** ~~Selection of “Item” for Purchase~~
    - ~~Above “Item” would be for the following. Should be able to use one function to select each item because it shouldn’t matter what the function is purchasing.~~
    - **~~Function:~~** ~~Selection of Lemons for Purchase~~
    - **~~Function:~~** ~~Selection of Cups for Purchase~~
    - **~~Function:~~** ~~Selection of Ice for Purchase~~
    - **~~Function:~~** ~~Selection of Sugar~~
  + **Function:** Purchase

**User Interface –**

* Class Responsibilities
  + **Function:** Given Option to Go to Store
  + **Function:** Display Welcome Screen
  + **Function:** Display the Rules
  + **Function:** Select Game Level
  + **Function:** Restart Game Choice

**Weather –**

* Class Responsibilities
  + **Function:** Generate list of weather for the days played
  + **Function:**  Generate list of temperatures

**Lemon –**

* Class Responsibilities
  + **Function:** Add to **List** of Lemons purchased

**Cups –**

* Class Responsibilities
  + **Function:** Add to **List** of Lemons purchased

**Ice –**

* Class Responsibilities
  + **Function:** Add to **List** of Ice purchased

**Sugar –**

* Class Responsibilities
  + **Function:** Add to **List** of Sugar purchased

1. Class Added by Game Designer [↑](#endnote-ref-1)
2. Class Determined Accurately Per Recommendations [↑](#endnote-ref-2)
3. Class Not Accurately Determined Per Recommendations

   Keywords:

   **Out.** This C# keyword signifies a reference parameter. Sometimes methods must return more than one value and not store class state.  
   **Out,** a keyword, fills these requirements. With it we pass parameters whose changes are realized in their calling methods. In a method, an out argument must be assigned.

   ref tells the compiler that the object is initialized before entering the function, while out tells the compiler that the object will be initialized inside the function.

   So while ref is two-ways, out is out-only. [↑](#endnote-ref-3)