|  |  |
| --- | --- |
| CIT 260 Team Project Proposal |  |
| Project Name | Billy Bob’s Farm |
| Team Members | Dan Crosby, Leanne Kendrick |
| Description | This project will include a game in which a player is given a farm plot, and at each round they can purchase plants which have a certain cost, a certain number of rounds for plants to mature, a certain amount of profit that might be earned, and certain disasters that might occur in the course of playing the game. Players of this game will learn to strategically select which plants to grow. |
| Plants that can grow on the farm and default properties:  (Note – the values in the table are intended as seed values only and will be adjusted as needed to make game play more challenging and interesting). | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | Carrots | Tomatoes | Potatoes | Corn | Watermelons | | Rounds to maturity | 1 | 2 | 2 | 3 | 3 | | Square footage required | .2 | 1 | .5 | .5 | 5 | | Harvest Rounds | 1 | 5 | 2 | 3 | 2 | | Seed cost | .02 | .15 | .20 | .10 | .05 | | Average sale price | .20 | 1.50 | 1.00 | .75 | 7.50 | |
| User Interface | A simple text-based interface.  At the beginning of the game, the user will be presented with options to start a game, or view the high scores, which would be retrieved from disk, or to exit.  After starting a game and at the start of each round, a player will be presented with their current cash level, the round number, an inventory of their farm/garden plot, and a description of the available UI controls to purchase plants and/or to complete the round.  At the completion of each round, the game will determine if any catastrophic events occurred at the vegetable or plant level (such as a tornado or tractor failure, which may affect the results of the round, including killing off plants, or adjusting the going market rate at which profits are realized.  At the completion of the game, a text file showing the players ending garden inventory and cash levels will be logged to the file system. The high scores list will also be updated. |
| Classes | We are planning 8 classes as indicated in the accompanying UML diagram.   * The main class is a static method which will direct the traffic for the game flow and contain any global functions that are not specific to any other class type. * The farm class is the main class that will contain the inventory of plants / vegetables that have been purchased. Note that each player has their own farm, so if it is a 2 player game, there will be 2 farms with different player names. * The plant class is an abstract class that will be implemented by the different vegetable types. This class will contain some base fields and methods that are common to all types of vegetables, including routines to determine if a disaster has occurred that might affect all crops. * Plant/vegetable specific classes for the 5 types of vegetables selected for this game. Each of these classes will inherit from the plant class and implement or override the methods as appropriate. Custom logic for disasters that can occur at the plant level will be implemented here. Different plants will also have custom progress messages if they require more than 1 round to mature. |
| Technical Notes | This project will implement Inheritance where in the individual vegetables will inherit features from the “plant” class.  This project will implement polymorphism when the Farm class loops through the plant objects and performs functions that are specific to the individual vegetable types rather than the plant class type, even though the plantList arrayList contains objects of type Plant. |
| Game completion | The game by default will complete after 10 rounds. Players who have earned more than $10 will be congratulated, players earning less than this will be advised that they have failed and should play again to ensure that they do not starve. |

Main Class:

* Note: Constants defined with Final are in all caps as per UML standards.
* Note: Fields and methods defined as static are underlined.



Farm, Plant, and Vegetable specific classes:

