heJEMony
Eric Li, Jenny Han, Matteo Wong
APCS2 pd3
UMLs — Burger Hegemony

Empire
Private ArrayList<Store> _stores Private int _employeeSalary Private double _budget Private int _totalEmployeeSatisfaction Private int _totalCustomerSatisfaction Private int _percentMeat Private ConcurrentLinkedQueue<String> _actionsList //there will be specific actions you can add that will be added automatically by the computer, such as buying a store, buying more cows, etc, each with an allotted time and will only be dequeued when that ends Private Stack<String> _milestones //they get popped off Private ALHeap<Farm> _farmHeap; Private ALHeap<Farm> _farmList; Public void buyStore(Store s, int cost) Public Store seeStore(String name) Public int getEmplSat() Public int getCustSat() Public int budget() Public void setBudget(double amount) Public int getMeatPercent() Public void setMeatPercent() Private void generateFarmHeap() Public void accessNewFarm() Public void runOperations()

Store

private ArrayList<Employee> _employees;
private int _customerSatisfaction;//represents number of customers per day
private int _employeeSatisfaction;//if too low strikes (extra feature)
private double _salary;//maybe, maybe not
private double _priceBurger;
private double _dailyRevenue;//for ease of calculations
private double _operationsCost;//cost of operations per cycle
private String _name;

private int _timeCreation; private static final String[] STORENAMES = {"Burger \nPalace", "Burger\nJoint", "Best\nBurger", "Speedy\nBurgers", "Top\nBurger", "Burger\nVillage", "Burger\nQueen", "Mc-\n-Dlanod", "Still\nCondo", "Blue\nCastle"}; private static int storePlace=9; public Store(int time) public void increaseStorePlace() public String getName() public Employee getEmployee(int i) public int numEmployees() public void modCustomerSatisfaction(int i) public void modEmployeeSatisfaction(int i) public void setDailyRevenue(Farm f) double getDailyRevenue() public void hire(Employee e) public void fire(int i) public void setSalary(double s) public double getSalary() public void setPrice (double s) public double getPrice() public boolean areCustomersHappy() public void increaseOperationsCost() public double getOperationsCost() public int getCreationTime() public void lowerEmployeeSatisfaction()

Employee
private String _name; private int _satisfaction; public Employee(String s) public String getName() public int getSatisfaction() public void decreaseSatisfaction()

Farm
private double percentRealMeat;

private String _name; private boolean chosen; public Farm () public Farm (double percent,String name) public double getPercentRealMeat () public double getCostPerPatty() public String getName() public int compareTo(Object o) public void toggleChosen() public boolean isChosen()

Interface Food
//will have several subclasses. Patty, Lettuce, Tomato, Bun, etc. Each will have it's own representation in Processing. No methods other than the patty

Patty implements Food
Boolean isCooked Public void cook()

Order
Private ConcurrentLinkedQueue<Food> _order Public void addToOrder() Public Food constructOrder()

Woo
import java.util.ArrayList; PImage img; PImage emp; PImage store; PImage farm; PImage miniStore; Empire empire; Int state

int totalTime;
int timeAction;
double storeCost;
Store currStore;
int currStoreNum;
int storeClosedScreenStartTime;

void setup()
void draw()
void mouseClicked()
void drawMenu()

boolean overButton(int x, int y, int width, int height)

void keyPressed()

void beginEmpire()
void printBudget()
void storesScreen()
void updateStoresScreen()
public String dollarToStr(double d)
void fireEmployeeButton(Store s)
void runIndividualStore(Store s)

void checkStoreButtons()
void storeClosed()

void printQ(int s)

void setupFarm()
void drawFarm()
Void farmButtons()

Minigame
//*****Minigame variables ALDeque<Order> _orders=new ALDeque<Order>(); ArrayList<Integer> burgerTimes = new ArrayList <Integer>();

```
int miniTime=0;//time variable
int y=185;//for printing orders
Order currOrder = new Order();
int placeForFood=500;//where you add food to
int currOrdNum=1;//which order currently on
float cash=0;
boolean bunClick=false;
int realTime=0;
int counterT=0;
PImage tomato;
PImage lettuce;
PImage patty;
PImage bun;
PImage topbun;
PImage cheese;

void setupMinigame() {
void drawMinigame()
void buttons()
void printOrders(int linelength)
void loadOrders()
void drawButtons()
void checkOrder()
```