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

Empire
private ArrayList<Store> _stores private ALQueue<Integer> _storesToClose private int _employeeSalary private double _budget private double _totalEmployeeSatisfaction private int _totalCustomerSatisfaction private ALQueue<Integer> _actionsList private int _patties private int[] _pattiesArray private ALHeap<Farm> _farmHeap private ArrayList<Farm> _availableFarms private Farm selectedFarm; private boolean hasAds Public Store getStore(int i) Public boolean getHasAds() Public void modHasAds(boolean b) Public void closeStore(int i) Public int nextStoreToClose() Public void queueStoreToClose(int i) Public double getTotalEmployeeSatisfaction() Public int getTotalCustomerSatisfaction() Public void modTotalEmployeeSatisfaction() Public double calculateTotalEmployeeSatisfaction() Public void modTotalcustomerSatisfaction(int i) Public void buyStore(Store s) Public void queueBuyStore(double d) Public Integer peekActions() Public int popActions() Public double getBudget() Public boolean isEmpty() Public void modifyBudget(double d) Public void runOperations(int tNow) Public void ecoli(int dec) Public int size() Public void addAction(int i) Public void setBudget(double s) Public void usePatties(Store s)

Public boolean buyPatties(int numPatty, Farm farm)
Public int getPatties()
Public ALQueue<IntegeR> retQ()
Private void buildFarmHeap()
Public void accessNewFarm()
Public int numUnlockedFarms()
Public Farm getFarm(int i)
Public void toggleAllOtherFarmsChosen(int i)
Public void setSelectedFarm(Farm farm)
Public Farm getSelectedFarm()
Public int getFarmNum()

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[] EMPLOYEENAMES={"Eric", "Jenny", "Matteo", "Topher", "Brian", "Kevin", "Donald", "John", "Matthew", "Chris", "Carl", "Jonas"}
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
Private int adType;

public void increaseStorePlace()
public String getName()
public Employee getEmployee(int i)
public int numEmployees()
public int getEmployeeSatisfaction()
public void onStrike()
public void endStrike()
public void raise(double d)
public void modCustomerSatisfaction(int i)
public int getCustomerSatisfaction () public void modEmployeeSatisfaction(int i)

public void setDailyRevenue(Farm f)
public 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 boolean striking()
public void lowerEmployeeSatisfaction()
public void increaseEmployeeSatisfaction(int i)
public void setAd(int i)
public int getAdType()
public double adSuccess(Farm f)
public String personName()

Employee

private String _name
private int _satisfaction

public Employee(String s)
public String getName()
public int getSatisfaction()
public void decreaseSatisfaction()
Public void modSatisfaction(int i)
Public double getSalary()
Public void modSalary(double d)

Farm implements Comparable

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, Cheese. Each will have it's own representation in Processing. No methods other than constructor for each item

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

Woo.pde (in processing, so all permissions are public)
int state boolean hasBeenSetUp PImage img PImage emp PImage store PImage farm PImage miniStore PImage hire PImage fire PImage ecoli PImage strike PImage win PImage out PImage clean PImage adScreen PImage farmChange PImage addAd PImage noAd boolean ecoliState

Empire empire
int totalTime
int timeAction
int farmToToggle
double storeCost=50000
double storeSell=10000
Store currStore
int currStoreNum
int storeClosedScreenStartTime=0
boolean playedMinigame
boolean strikeBoo
boolean changingFarm

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

boolean overButton(int x, int y, int width, int height)
String dollarToStr(double d)
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()
Void ecoliRun()
Void ecoliButton()
Void loadInfo()
Void loadAdScreen()
Void adButtons()

Minigame.pde (processing so all permissions are public)

ALDeque<Order> _orders

ArrayList<Integer> burgerTimes
int miniTime
int y
Order currOrder
int placeForFood
int currOrdNum
float cash
int bunClick
boolean played;
int realTime;
int counterT;
PImage tomato;
PImage lettuce;
PImage patty;
PImage XYZ;//bottom bun
PImage ABC;//top bun
PImage cheese;

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

ALDeque<T>

Private ArrayList<T> _deque

Public void addLast(T val)
Public T peekLast()
Public T peekFirst()
Public T pollFirst()
Public T pollLast()
Public boolean isEmpty()
Public int size()
Public String toString()
Public boolean contains(T val)

ALHeap<T>

Private ArrayList<T> _heap

Public String toString()
Public void add(T addVal)
Public boolean isEmpty()
Public T peekMin()
Public T removeMin()
Private int minChildPos(int pos)
Private T minOf(T a, T b)
Private void swap(int pos1, ing pos2)

ALQueue<T>

Private ArrayList<T> _queue;

Public int size()
Public void enqueue(T x)
Public T dequeue()
Public T peekFront()
Public boolean isEmpty()
Public T getN(int s)
Public String toString()

Interface Deque<T>

void addLast(T val);
T peekLast();
T pollFirst();
T pollLast();
T peekFirst();
boolean isEmpty();
int size();
boolean contains(T val);

Interface Queue<Quasar>

public Quasar dequeue();
public void enqueue(Quasar x);
public boolean isEmpty();
public Quasar peekFront();