```
private static void produceNewCompany() {
          //Decide which item to produce
44000
          int newItemType;
          if(monopolyMode == false) {
             newItemType = randomInteger(1, listOfAltItems.size() + 1);
          else {
Testing Economic Ideas by Using a
         Simulation of an Economy
             ListkItems new Ingredients = Name: Mark Klinchin
             Decide the ingredients
             for(Iterator<Item> i = ListOfALLItems.Iterator(); i.hasNext();)
                Item this could School: Lower Moreland High School
                //This makes it so that on Year in PJAS: 4th ingredients to make a new item, but there coul
30000
                double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                   newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
44000
                                     Raitionale
           int newItemType;
           if(monopolyMode == false) {
              newItemType = randomInteger(1, listOfAltItems.size() + 1);
           } else {
              newItemType = listOfALLItems.size()+1;
"Nobody ever had really actually estimated a demand curve ... I literally could
not find a good example where we could put it in a box ... to say, 'This is what
a demand curve really looks like in the real world,"
                   -Steve Levitt, Professor of Economics at University of Chicago
820000
                 //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                 double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                 if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                     newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
44000
                                          Problem
           int newItemType;
           if(monopolyMode == false) {
               newItemType = randomInteger(1, listOfAltItems.size() + 1);
            else {
               newItemType = ListOfALLItems.size()+1;
           Item newItem;
           if(isThisAlreadyAnItem(newItemType) == false) {
Will the revenue of companies selling one product be greater when a monopoly
on the product exists, compared to when competition occurs?
               for(Iterator<Item> i = listOf/AllItems.iterator(); i.hasNext();)
                   Item thisCouldBeAnIngredient = i.next();
80000
                   //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                   double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                   if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                      newIngredients.add(thisCouldBeAnIngredient);
```

//Decide which item to produce
int new Research I: Economics Basics
newItemType = randomInteger(1, listOfAltItems.size() + 1);

• Demand: How much people want an item

else {

- Demand curve: Plot price over products and see how much is demanded for each price
- Supply: The amount of an item companies make
 - Supply Curve: Plot price over products and see how much is supplied for each price
- Intersection of the two curves: Optimal price and amount of a product



```
//Decide which item to produce
int new Research II: On the Margins
new Item Type = random Integer (1, list Of All Items. size() + 1);
} else {
```

- Marginal Cost (MC) The cost of adding one more of something
 - For firms: The cost of producing one more good
- Marginal Benefit (MB) The benefit of one additional thing
 - For firms: The price of the next good that is sold
- The Law of Diminishing Returns As the amount of a product increases, MB decreases (not necessarily total benefit)
- Intersection of MC and MB: Ideal amount of anything to produce

```
Item thisCouldBeAnIngredient = i.next();

//This makes it so that on average, it takes 2 ingredients to make a new item, but there coul double chanceOfItemBeingAnIngredient = 2 / ListOfALLItems.size();

if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
    newIngredients.add(thisCouldBeAnIngredient);
}
```

//Decide which item to produce int newItem Research III: Firm Theory

```
newItemType = randomInteger(1, listOfAllItems.size() + 1);
} else {
```

Price (P) – How much a good costs

private static void produceNewCompany() {

- Quantity (Q) Amount of products produced
- Marginal Revenue (MR) Revenue that changes based on the amount of products sold, revenue of selling one more product (similar to MB)
- Marginal Cost (MC) Change in total cost of making one more item
- Variable Cost Cost of producing that changes based on amount of goods
 - Examples: Ingredients, Labor (); LhasNext();
- Fixed cost (FC) Cost that does not change based on amount of goods
 - Description of the characteristic of the control of the control of the characteristic of
- Total Cost (TC) to Cost (TC) Total Cost (TC)
- Average Cost TC:/Qents.add(thisCouldBeAnIngredient);

int newItem Research IV: Competition [Monopoly] Research IV: Competition

- newItemType = randomInteger(1, listOfAltItems.size() + 1);
 } else {
- Perfect Competition Infinite number of companies producing same exact good
 - Competitive firms Companies under competition

private static void produceNewCompany() {

- **P**=**MC**=**MB**
- Does not actually exist
- Monopoly Industry in which one company produces a good
 - Output effect When quantity increases, revenue increases
 - Price effect When quantity increases, price decreases
 - P>MC=MR
 - Monopolies can make P>MC on average, it takes 2 ingredients to make a new item, but there coul
- Since P can be greater than MC in monopoles, they can get more revenue than competitive firms



```
private static void produceNewCompany() {

//Decide which item to produce
int newItemType;
if(monopolyMode == false) {

newItemType = randomInteger(1, listOfAltItems.size() + 1);
} else {
```

- Based on the research, in a simulation of an economy in which one company sells a product, or a monopoly, company will have a higher average revenue per market run than if companies were under competition.
 - Null hypothesis: in a simulation of an economy in which one company sells a product, or a monopoly, companies will have the same average revenue per market run than if companies were under competition.

```
//Decide the ingredients
for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();) {
    Item thisCouldBeAnIngredient = i.next();

    //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
    if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
        newIngredients.add(thisCouldBeAnIngredient);
    }
}
```

```
private static void produceNewCompany() {
           //Decide which item to produce
44000
           int newItemType; Experiment I: Classes
               newItemType = randomInteger(1, listOfAltItems.size() + 1);
            else {
               newItemTvne = listOfALLItems.size()+1;
                                                                                      Company
40000
                Person
                       cuayanitem(newlt
                           going to make an
                                                                 he item and ingredients
614
                      ems newIngredients = n
                                                  Controller
                 Decide the ingredients
                                                                 hasNext();)
               for(Iterato (Item> i = ListOf/
                   Item thistorldBeAnIngredient = 1.nexx
80000
                   //This makes it so that on avera
                                                                              make a new item, but there coul
                                                       takes 2 ingredients
                                                                          size();
                   double chanceOfItexBeingAn
                                                                 fALLItem.
                   if(smallChanceOccurr)(chan
                                                    Item
                       newIngredients.add(
```

/Decide which item to produce Experiment II: Class Descriptions

newItemType = randomInteger(1, listOfAltItems.size() + 1);

tem thisCouldBeAnIngredient = i.next();

**ControllerewItemType = listOfALLItems.size()+1;

private static void produceNewCompany() {

- Only class that is run
- Contains list of all people, items, and companies
- Manages most relationships between objects Listeltems newlngredients = new ArrayLi

Decide the ingredients

- Person
 - Iterator<Item> i = listOfAllItems.iterato Has money
 - Gets money by receiving it from companies
 - Spends money by trying to buy all Items as cheap as possible add(thisCouldBeAnIngredi
 - Has willingness to pay for every item

- Item
 - Has market price
 - Can have ingredients (other items)
- Company
 - Gets money by selling items
 - Produces one type of item
 - Has a list of employees
 - If amount of employees decreases, wages increase (otherwise, they decrease)
 - Manages price it sells by increasing oul price if it previously gave revenue, decreasing otherwise
 - Has optimal amount of products to produce (to simulate MC curve)

```
private static void produceNewCompany() {
            //Decide which item to produce
           Experiment III: Starting Condition
44000
                newItemType = randomInteger(1, listOfAltItems.size() + 1);
            else {
• 3 lists created emType = listOfALLItems.size()+1;
    • One with 100 people (each with $10)

    One with 2 items

      One with 2 companies (one per item)

if (isThisAlreadyAnItem(newItemType) == false)
             //if we are going to make a new item, this creates the item and ingredients
                ListkItem> newIngredients = new ArrayListkItem>();
                //Decide the ingredients
                for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();)
                   Item thisCouldBeAnIngredient = i.next();
80000
                   //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                   double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                   if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                       newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
           Experiment IV: Pre-Market Runs
              newItemType = randomInteger(1, listOfAltItems.size() + 1);

    All companies pay fixed cost ($100)

    Companies produce optimal amount of products

    Products require buying ingredients and labor

    Ingredients are bought directly from other companies
   • For each product produced, less labor is required (MC going down)

    For each hour of labor, an employee is payed the company salary

              //Decide the ingredients
              for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();)
                 Item thisCouldBeAnIngredient = i.next();
30000
                 //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                 double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                 if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                     newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
           Experiment V: Market Runs
               newItemType = randomInteger(1, listOfAltItems.size() + 1);
            else {

    All people cycle through each item

    For each item, see which company sells it

   • Find the company that sells the item for the cheapest price

    Try to buy from that company

       • If the person is willing to pay for the item and can afford to buy it, the transaction is made
               List (Item> newIngredients = new ArrayList (Item>();
               //Decide the ingredients
               for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();)
                  Item thisCouldBeAnIngredient = i.next();
&@000
                  //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                  double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                  if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                      newIngredients.add(thisCouldBeAnIngredient);
```

Experiment VI: Post-Market Runs

```
newItemType = randomInteger(1, listOfAltItems.size() + 1);
```

• Refresh People statistics Litems size()+1;

private static void produceNewCompany() {

- Every 10 rounds, people choose new company to work for (highest wage)
- Every 10 rounds, people create new willingness to pay for all items
- Different time for every person, same length of time before refresh
- Produce new companies
 - Every round, there is a 5% chance that a new company is formed
 - If monopoly mode is off, it produces a random item (one option per existing item, one option for making a new item)

temm this creates the item and ingredients

- If monopoly mode is on, I produces a new item
- Monopoly mode is a Boolean value that can be turned on or off before execution

```
newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
  Experiment VII: Additional Setup Information
              newItemType = randomInteger(1, listOfAltItems.size() + 1);
           else {
• One round definition stofAllItems.size()+1;
   • Pre-market run

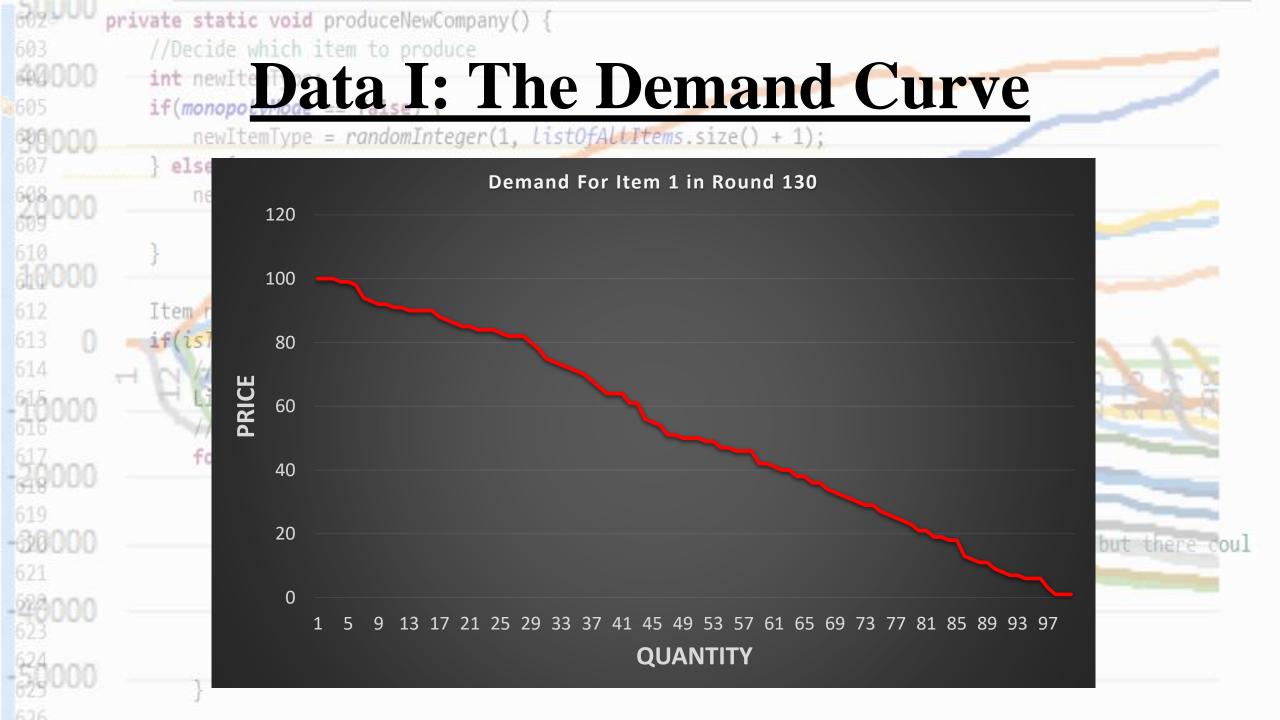
    Market run

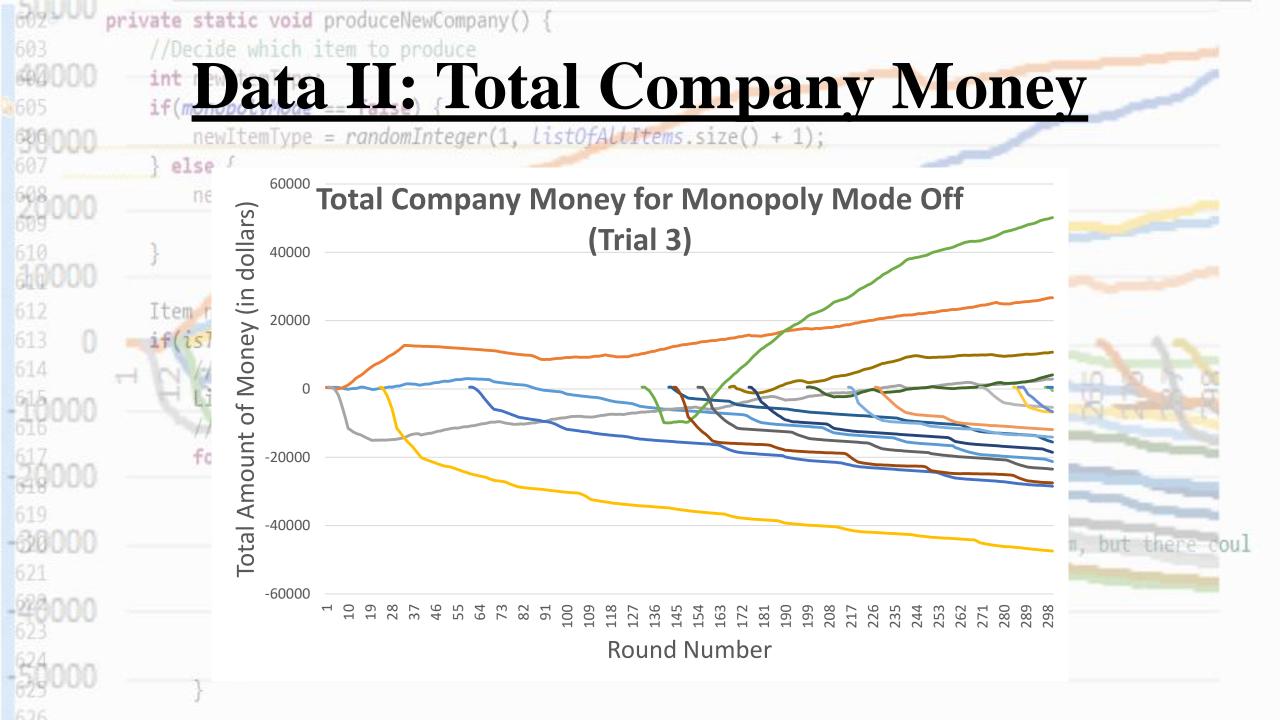
    Post-market run

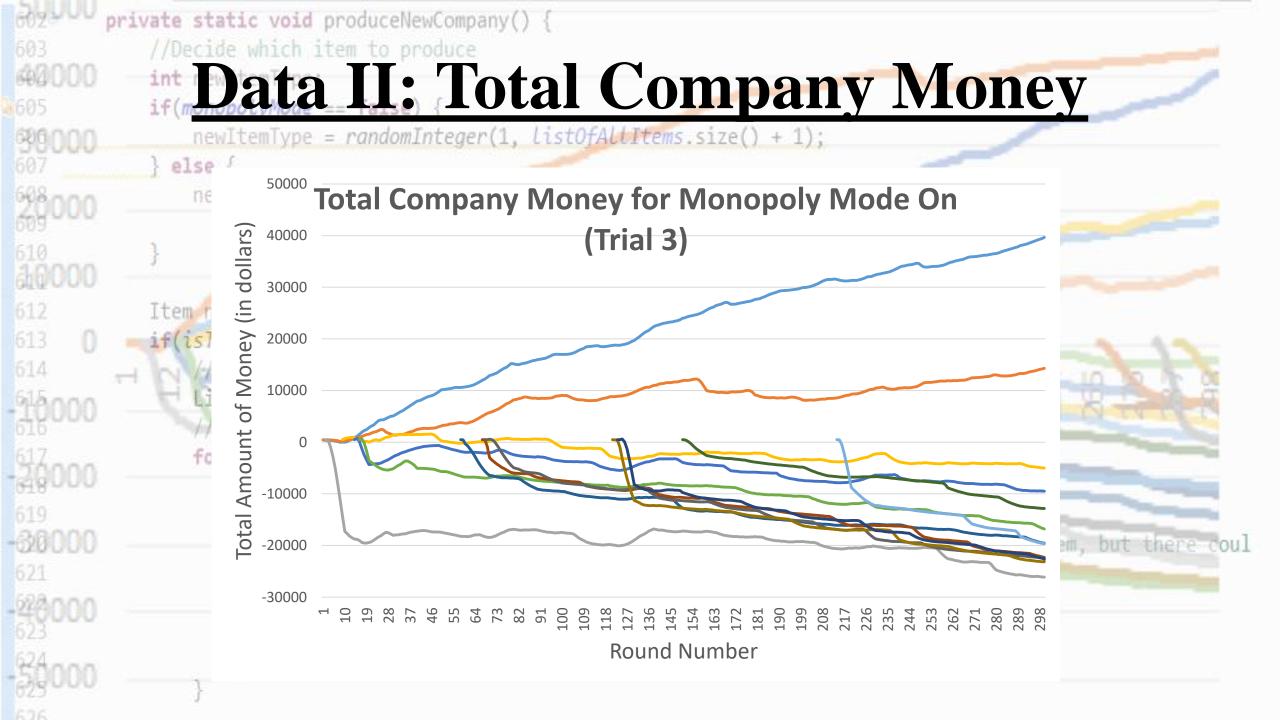
• 300 rounds per trial
                                   a new item, this creates the item and ingredients
• 10 trials per group of trials - new ArrayList*Item>();
• 2 groups of trials i = tistOfAllItems.iterator(); i.hasNext();)

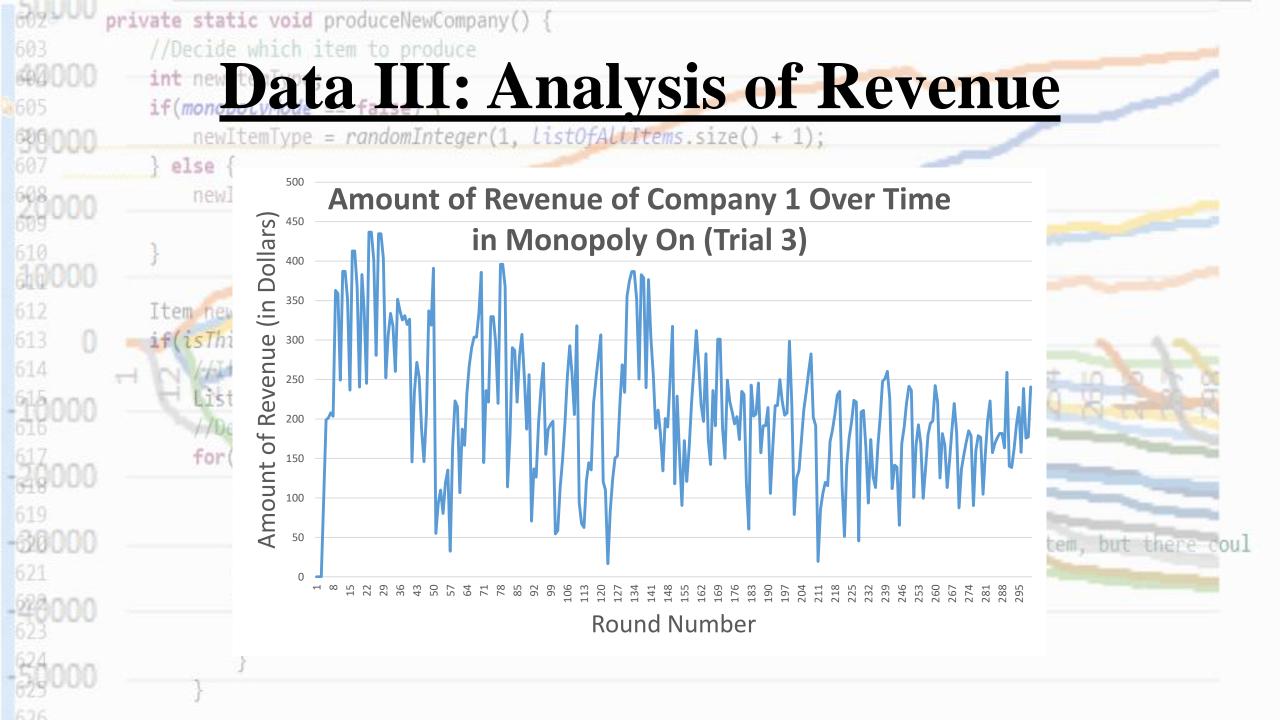
    Monopoly mode off wide-AnIngredient = 1.next();

   • Monopoly mode on //Inis makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                 double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                 if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                    newIngredients.add(thisCouldBeAnIngredient);
```









```
private static void produceNewCompany() {
           //Decide which item to produce
          Data III: Analysis of Revenue
              newItemType = randomInteger(1, listOfAltItems.size() + 1);
           else {
• In each trialItemType = listOfALLItems.size()+1;
   • Find the average revenue per round for each company (300 data points)
   · Average all of these numbers to find average revenue per round for entire trial

    Find the average revenue per round for each experimental group

   • Find the average revenue per round for each trial

    Average all of those numbers to find the average for each experimental group

               //Decide the ingredients
              for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();)
                  Item thisCouldBeAnIngredient = i.next();
                  //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                  double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                  if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                     newIngredients.add(thisCouldBeAnIngredient);
```

private static void produceNewCompany() { //Decide which item to produce

44000

Data III: Analysis of Revenue

ହେନ	000		ne	wItemT	ype	= 1	andom.	Integer	(1, L	istOfA	LLItem	s.size	() + 1);						
Isa7	Ser Ser Ser	1	oles																	
279	191.6994	-	279	191.6994		279	191.6994	193.9243	120.4764	78.6838	60.24458	50.01144	59.16977	33.46215	24.1373	13.95636	2.667912	1.492362	1.518233	1.758313
280	112.9344		280	112.9344	ype	280	112.9344	236.6534	171.0008	205.4521	77.96357	65.14808	49.30814	49.24619	36.20595	16.28242	15.60729	1.462213	4.099229	3.726108
281	93.04071		281	93.04071		281	93.04071	59.16336	0	0	62.01648	2.171603	25.95909	34.72488	0	0	0	0	0	3.888113
282	98.1828		282	98.1828		282	98.1828	51.48276	92.22129	44.02453	0	23.22496	0	0	15.16568	0	0	19.12489	4.067915	3.888113
283	98.1828	}	283	98.1828		283	98.1828	136.0858	53.20459	92.94068	2.484007	20.19561	13.43774	18.69068	10.1797	0	0	0.597653	1.355972	2.770281
284	87.48088		284	87.48088		284	87.48088	169.2775	118.9236	51.36196	2.484007	20.52039	24.80814	23.60928	0	0	0	0	0	0
285	187.295		285	187.295		285	187.295	146.0433	46.53534	123.5234	6.706818	2.280044	43.41424	19.6744	22.93311	0	0	0	0	1.082793
286	209.7704	It	286	209.7704	19	286	209.7704	187.2325	59.77664	38.60105	114.0159	22.39541	11.3704	14.4596	16.23355	3.442517	3.216958	15.96364	0	5.413967
287	199.7814	- 1-9	287	199.7814	eady	287	199.7814	239.4835	91.4231	103.7526	46.94773	31.12582	44.07953	19.88888	8.521517	1.721259	3.216958	15.07677	9.143563	10.82793
288	101.5992		288	101.5992	Curay	288	101.5992	239.4835	119.5533	101.6352	71.66274	25.05249	11.01988	33.14813	12.20276	10.83958	4.342894	1.539785	3.516755	7.146436
289	196.5866	-10	289	196.5866	are	289	196.5866	258.6421	126.5858	105.3467	50.16392	51.56694	39.88046	21.41879	12.84501	4.335832	15.92394	3.714956	18.74309	23.47406
290	202.8606	\rightarrow	290	202.8606	mes	290	202.8606	282.1551	123.0696	124.5006	66.11365	43.31623	18.35767	31.22392	10.27601	41.5976	26.05736	9.456251	30.67051	25.60807
- 291	115.0241	-	291	115.0241	HI P	291	115.0241	90.13287	5.383506	25.53859	29.0255	44.23503	0	0	0	0	0	18.23706	0	0
292	68.52566		292	68.52566	the	292	68.52566	96.15022	22.02252	69.31933	0	23.88692	22.90623	7.924708	4.859471	0	15.84616	0	2.629765	0
293	122.4555		293	122.4555	ator	293	122.4555	76.92018	36.7042	69.31933	17.57055	4.421818	29.49569	44.50029	4.693808	8.983781	3.961539	0	0	0
294	147.4464		294	147.4464	a LOI	294	147.4464	103.5278	80.74925	110.292	35.1411	27.88453	16.94433	0.609593	0	1.122973	0	0	0	0.434395
295	172.4373		295	172.4373	thi	295	172.4373	153.0411	117.4534	86.89673	21.08466	32.31065	15.14106	3.949328	6.540378	2.156716	17.60564	0	0	13.03186
296	194.9291		296	194.9291		296	194.9291	189.0508	69.73798	125.2242	36.73392	42.93333	30.69731	17.77198	5.755533	4.313431	13.76073	1.308114	15.5932	11.29428
297	217.4209		297	217.4209		297	217.4209	229.5617	157.8253	58.03071	3.06116	42.49071	37.34839	3.949328	46.08369	0	1.380264	0	4.158187	0
**298	247.41		298	247.41	15 1	298	247.41	279.075	169.0986	89.96917	70.79336	35.05851	28.65082	20.64776	14.9772	9.861066	11.63651	8.737475	10.60485	11.13532
299	212.4227		299	212.4227	le r	299	212.4227	279.075	169.0986	59.97944	41.74993	41.19656	23.76508	22.67205	8.664776	6.104469	7.854648	8.065361	10.60485	10.27876
300	81.58187		300	81.58187		300	81.58187	295.7295	197.8453	128.5693	84.68429	42.91309	47.07259	38.86637	45.65846	50.79162	30.34111	31.07041	19.08873	10.94401
301				165.4905	mall	301	165.4905	243.3473	136.4176	154.6587	85.94696	62.72372	39.73142	36.5302	19.89145	28.04063	10.2551	10.16834	7.056051	3.765789
I ANN	WWW.					Town in		1.1	11110	3 Jn.	a +	1.5		177						

newIngredients.add(thisCouldBeAnIngredient);

private static void produceNewCompany() { //Decide which item to produce Data III: Analysis of Revenue newItemType = randomInteger(1, listOfAltItems.size() + 1); olso 120.4764 78.6838 60.24458 50.01144 59.16977 33.46215 24.1373 13.95636 2.667912 1.492362 1.518233 1.758313 0 1.082793 0 0.434395 0 23.22496 301 165.4905 243.3473 136.4176 154.6587 85.94696 62.72372 39.73142 36.5302 19.89145 28.04063 10.2551 10.16834 7.056051 3.765789 98.1828 51.48276 92.22129 44.02453 0 15.16568 0 19.12489 4.067915 3.888113 53.20459 92.94068 2.484007 20.19561 13.43774 18.69068 0 1.082793 double chanceOfItemBeingAnIngredien 0 15.84616 0 2.629765 if(smallChanceOccurrs(chanceOfItemB newIngredients.add(thisCouldBeA 0 1.380264 301 165,4905 243,3473 136,4176 154,6587 85,94696 62,72372 39,73142 36,5302 19,89145 28,04063

```
private static void produceNewCompany() {
            //Decide which item to produce
           Data III: Analysis of Revenue
44000
                newItemType = randomInteger(1, listOfAltItems.size() + 1);
            else {
                newItemType = listOfALLItems.size()+1;
             Monopoly Mode off
              71.20993 69.61965 67.58548 71.15215 59.25903 66.94451 83.63996 72.05477 87.45442
                                                                                           67.619
             Monopoly Mode on
              71.71598 70.07242 71.02088 75.73776 76.40646 68.29898 71.92695
                                                                         77.5265
                                                                                 73.7966 68.26062
            if(is/hisAlreadyAnitem(newlitemlype) == fals
                                                                                                       Standard
                                                        s creates the item and ingredients
               White we are going to make a new item
                                                                                               Average Deviation
             Monopoly Mode off
             71.20993 69.61965 67.58548 71.15215 59.25903 66.94451 83.63996 72.05477 87.45442
                                                                                          67.619 71.65389 8.210516
             Monopoly Mode on
             71.71598 70.07242 71.02088 75.73776 76.40646 68.29898 71.92695
                                                                        77.5265
                                                                                73.7966 68.26062 72.47632 3.288528
30000
                    //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                   double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                   if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                       newIngredients.add(thisCouldBeAnIngredient);
```

614

```
private static void produceNewCompany() {
         //Decide which item to produce
         Data IV: Five-Number Summary
            newItemType = randomInteger(1, listOfAltItems.size() + 1);
         l oleo /
    Monopoly Mode On
                         68.269.63 71.82
                                              75.9077.53
    Monopoly Mode Off
68.26
                      67.43
                               70.39
                                           74.95
                                                                              87.45
   60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
                                                                                         coul
                        Average Revenue Per Company Per Round
               if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                  newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
          Data V: Testing the Hypothesis
              newItemType = randomInteger(1, listOfAllItems.size() + 1);
           else {

    Average revenue per round with no monopolies: $71.65

    Standard deviation: 8.21

    Average revenue per round with monopolies: $72.48

    Standard deviation: 3.29

• Percentage difference: 1.14%
                                         tem, this croates the item and ingredients
   • Not statistically significant
              for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();)
                 Item thisCouldBeAnIngredient = i.next();
80000
                 //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                 double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                 if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                    newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
          //Decide which item to produce
          int newItemType; Conclusions I
        int newItemType;
              newItemType = randomInteger(1, listOfAltItems.size() + 1);
           else
• Hypothesis: Rejected stofALLItems.size()+1;
```

- Null hypothesis accepted
- Reasoning: No statistically significant difference between revenues of companies under competition (\$71.65) and monopolies (\$72.48) • Possible errors

to make a new item, this creates the item, and ingredients

- - Unrealistic simulation edients = new ArrayList&Item>();
 - Not everyone needs to buy everything
 - Optimal amount of products comes naturally, is not a random number
 - Simulation not long enough
 - Often companies would begin gaining money as trial ends

```
double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
    newIngredients.add(thisCouldBeAnIngredient);
```

```
private static void produceNewCompany() {
           //Decide which item to produce
          int newItemType; Conclusions II
       int newItemType;
              newItemType = randomInteger(1, listOfAltItems.size() + 1);

    If the experiment were to be repeated

    Add dynamic population

    Population fluctuates in the short run

    Population exponentially grows in the long run

    • Add demand-side complexity (pe) == false)
       • Indifference curves a to make herewill amount a coates the liter and ingredients
       • Inferior goods newIngredients = new ArrayList<Ttem>();
Further study terator (Item > i = ListOfAllItems.iterator(); i.hasNext();)
                     e the ingredients

    How true is the Law of Supply?

    How true is the Law of Demand?

    • How true is the Law of Diminishing Returns? [15t0fAllItems.size()]

    Real world application currs (chanceOfItemBeingAnIngredient)) {
```

- Economic policies can be tested using accurate simulations instead of theory
- Predictions for the future of the economy can be predicted using accurate simulations

```
private static void produceNewCompany() {
                 //Decide which item to produce
                                                         Works Cited
                 int newItemType;
                 if(monopolyMode == false)
                       newItemType = randomInteger(1, listOfAltItems.size() + 1);
                  else {
Finkelstein, Gabriel. "Fundamentals of Microeconomics." Center for Talented Youth 2016. Dickinson University, Carlisle. July-Aug. 2016. Lecture.
Jerison, Amalia. "Lecture 30 Notes." State University of New York Albany. State University of New York Albany, n.d. Web. <a href="http://www.albany.edu/~aj4575/LectureNotes/Lecture30.pdf">http://www.albany.edu/~aj4575/LectureNotes/Lecture30.pdf</a>.
Taylor, John. "Key Concepts in Microeconomics." The University of Pittsburgh. The University of Pittsburgh, n.d. Web. <a href="http://www.pitt.edu/~upjecon/MCG/MICRO/MicroConcepts.html">http://www.pitt.edu/~upjecon/MCG/MICRO/MicroConcepts.html</a>>.
"The 51 Key Concepts." Library of Economics and Liberty. Library of Economics and Liberty, n.d. Web. <a href="http://www.econlib.org/library/Topics/HighSchool/KeyConcepts.html">http://www.econlib.org/library/Topics/HighSchool/KeyConcepts.html</a>>.
                 if(isThisAlreadyAnItem(newItemType) == false) {
                       Wir we are going to make a new item, this creates the item, and ingredients
                       List (Item> newIngredients = new ArrayList (Item>();
                        Decide the ingredients
                       for(Iterator<Item> i = listOfAllItems.iterator(); i.hasNext();)
                             Item thisCouldBeAnIngredient = i.next();
                             //This makes it so that on average, it takes 2 ingredients to make a new item, but there coul
                            double chanceOfItemBeingAnIngredient = 2 / ListOfAllItems.size();
                             if(smallChanceOccurrs(chanceOfItemBeingAnIngredient)) {
                                  newIngredients.add(thisCouldBeAnIngredient);
```

```
502-111
        private static void produceNewCompany() {
            //Decide which item to produce
44000
            int newItemTyp
            if(monopolyMode == fa
                newItemType = rai
                                      nteger(1, Li
                                                       ALLItems
39000
             else {
608
609
000
                                      LLItems.size
                newItemType = Lis
40000
            Item newItem;
            if(isThisAlre
             //If we are going to make a new item, this creates the item, and ingredients
614
                List (Item> newIngredients = new ArrayList (Item>();
16000
                 //Decide the ingredients
                for(Iterator(Item> i = listOfAllItems.iterator(): i hasNext():) {
                    Item thi
                                  dBeAnlı
-30000
                                                                                         new item, but there coul
                    //This m
                                       that on av
                                                                   2 ingr
                                                                             ats to
                                                                 ist0fA
                    double c
                                       mBeingAnIng edien
                                                                             ms.size(
                                                                 redient
                    if(small
                                      rrs(chance( ItemBei
                                  e0cc
                                  ients.add/thisCouldBeAnIng
                                                                 nt);
                         newI
```