```java
/*
 * Programmer:  Chad Philip Johnson
 * Date Created:  Thursday, November 15th, 2012
 * Date of Last Modification:  Thursday, November 15th, 2012
 *
 * Description:
 * Farm.class contains the main program subroutines used by MyFarm.class.  It
 * depends on Barn.class to create instances of Barn which contain instances of
 * Animal, Animal.class to define various farm animals, CinReader.class for user
 * input, and FarmStorage.class for saving/loading games.
 */

import java.util.Random;
import java.text.DecimalFormat;
import java.io.Serializable;

/**
 * Farm.class contains the main program subroutines used by MyFarm.class.  It
 * depends on Barn.class to create instances of Barn which contain instances of
 * Animal, Animal.class to define various farm animals, CinReader.class for user
 * input, and FarmStorage.class for saving/loading games.
 *
 * @author Chad Philip Johnson
 * @version 1.0
 */

public class Farm implements Serializable   {

    // Constant values representing important game balancing properties
    static final int MAX_NUMBER_OF_BARNS    = 6;
    static final int SIZE_SMALL_BARN        = 2;
    static final int SIZE_MEDIUM_BARN       = 3;
    static final int SIZE_LARGE_BARN        = 4;
    static final double COST_SMALL_BARN     = 40000.00;
    static final double COST_MEDIUM_BARN    = 58000.00;
    static final double COST_LARGE_BARN     = 75000.00;
    static final double COST_EXPAND_BARN    = 20000.00;

    static final double STARTING_FUNDS      = 80000.00;
    static final int STARTING_TURNS         = 12;

    static final double COST_MODIFIER       = 0.10;

    static final double STARTING_FEED_PRICE     = 2.00;
    static final double STARTING_COW_PRICE      = 1000.00;
    static final double STARTING_CHICKEN_PRICE  = 100.00;
    static final double STARTING_TURKEY_PRICE   = 150.00;
    static final double STARTING_HORSE_PRICE    = 800.00;

    static final double STARTING_COW_MILK_PRICE = 100.00;
```

```
51      static final double STARTING_COW_MEAT_PRICE = 6000.00;
52      static final double STARTING_CHICKEN_EGG_PRICE  = 40.00;
53      static final double STARTING_CHICKEN_MEAT_PRICE = 300.00;
54      static final double STARTING_TURKEY_MEAT_PRICE  = 750.00;
55      static final double STARTING_HORSE_TRANSPORTATION_PRICE = 3500.00;
56
57      static final int FEED_NEEDED_TO_PRODUCE_COW         = 2;
58      static final int FEED_NEEDED_TO_PRODUCE_CHICKEN     = 6;
59      static final int FEED_NEEDED_TO_PRODUCE_TURKEY      = -1;
60      static final int FEED_NEEDED_TO_PRODUCE_HORSE       = -1;
61      static final int FEED_NEEDED_FULL_GROWN_COW         = 1000;
62      static final int FEED_NEEDED_FULL_GROWN_CHICKEN     = 50;
63      static final int FEED_NEEDED_FULL_GROWN_TURKEY      = 100;
64      static final int FEED_NEEDED_FULL_GROWN_HORSE       = 600;
65      static final int MAX_FEED_PER_DAY_COW       = 50;
66      static final int MAX_FEED_PER_DAY_CHICKEN   = 10;
67      static final int MAX_FEED_PER_DAY_TURKEY    = 15;
68      static final int MAX_FEED_PER_DAY_HORSE     = 35;
69
70      static final int TURN_COST_FEED_ANIMAL      = 2;
71      static final int TURN_COST_PREPARE_PRODUCE  = 1;
72      static final int TURN_COST_PREPARE_ANIMAL   = 2;
73      static final int TURN_COST_BUY_FEED         = 2;
74      static final int TURN_COST_BUY_ANIMAL       = 4;
75      static final int TURN_COST_MOVE_ANIMAL      = 2;
76      static final int TURN_COST_SELECT_ANIMAL    = 3;
77      static final int TURN_COST_SELECT_BARN      = 1;
78      static final int TURN_COST_INCOMPLETE_TASK  = 1;
79      static final int TURN_COST_PACK_BARN        = 2;
80      //static final int TURN_COST_SELL_AT_MARKET = intMaxTurnsPerDay;
81      //static final int TURN_COST_BUY_BARN = intMaxTurnsPerDay;
82      //static final int TURN_COST_EXPAND_BARN = intMaxTurnsPerDay;
83      //static final int TURN_COST_TEAR_DOWN_BARN = intMaxTurnsPerDay;
84      //static final int TURN_COST_FINISH_DAY = intRemainingTurns;
85
86      CinReader driverKeyboard;
87      DecimalFormat currencyFormatter;
88
89      Barn[] barnsOnFarm;
90      double dblMoney;
91      int intAvailableFeed;
92      double dblFeedPrice;
93
94      int intCurrentNumberOfBarns;
95      double dblCowPrice, dblChickenPrice, dblTurkeyPrice, dblHorsePrice;
96      double dblCowMilkPrice, dblCowMeatPrice, dblChickenEggPrice, dblChickenMeatPrice, dblTurkeyMeatPrice,
        dblHorseTransportationPrice;
97
98      Random randomNumber;
99
```

```
100       String strFarmName, strCurrentDay;
101       int intRemainingTurns, intMaxTurnsPerDay, intNumberOfDays;
102       int intSelectedBarn, intSelectedAnimal;
103       boolean blnAnimalSelected, blnProduceToSell;
104
105       double dblMoneyToEarnOnNextSell, dblAmountToEarn;
106       int intNumberOfUnitsSold;
107
108       char chrUserInput;
109
110       /**
111         * Default constructor:
112         * Instantiate CinReader.class, DecimalFormat.class and Random.class and associate them with the current instance of
          Farm.class,
113         * and initialize variables.
114         */
115
116       public Farm()   {
117
118           this.driverKeyboard                = new CinReader();
119           this.currencyFormatter             = new DecimalFormat( "#,###.00" );
120           this.randomNumber                  = new Random();
121
122           this.dblMoney                      = STARTING_FUNDS;
123           this.intAvailableFeed              = 0;
124
125           this.dblFeedPrice                  = STARTING_FEED_PRICE;
126           this.dblCowPrice                   = STARTING_COW_PRICE;
127           this.dblChickenPrice               = STARTING_CHICKEN_PRICE;
128           this.dblTurkeyPrice                = STARTING_TURKEY_PRICE;
129           this.dblHorsePrice                 = STARTING_HORSE_PRICE;
130
131           this.dblCowMilkPrice               = STARTING_COW_MILK_PRICE;
132           this.dblCowMeatPrice               = STARTING_COW_MEAT_PRICE;
133           this.dblChickenEggPrice            = STARTING_CHICKEN_EGG_PRICE;
134           this.dblChickenMeatPrice           = STARTING_CHICKEN_MEAT_PRICE;
135           this.dblTurkeyMeatPrice            = STARTING_TURKEY_MEAT_PRICE;
136           this.dblHorseTransportationPrice   = STARTING_HORSE_TRANSPORTATION_PRICE;
137
138           this.intRemainingTurns             = STARTING_TURNS;
139           this.intNumberOfDays               = 0;
140           this.strCurrentDay                 = "Sunday";
141
142           this.barnsOnFarm                   = new Barn[MAX_NUMBER_OF_BARNS];
143           this.intCurrentNumberOfBarns       = 0;
144           this.intSelectedBarn               = 0;
145           this.intSelectedAnimal             = -1;
146           this.blnProduceToSell              = false;
147           this.blnAnimalSelected             = false;
148           this.dblMoneyToEarnOnNextSell      = 0.00;
```

```
149
150         this.dblAmountToEarn                = 0.00;
151         this.intNumberOfUnitsSold           = 0;
152
153         this.intMaxTurnsPerDay              = STARTING_TURNS;
154
155     }
156
157     /**
158      * Overloaded constructor:
159      * Instantiate CinReader.class, DecimalFormat.class and Random.class and associate them with the current instance of
         Farm.class,
160      * and initialize variables.  Can change the initial number of turns available to the player on an instance of Farm.class.
161      * @param intMaxTurnsPerDay the max number of turns available to the player at the beginning of the game.
162      */
163
164     public Farm( int intMaxTurnsPerDay )    {
165
166         this();
167
168         this.intMaxTurnsPerDay  = intMaxTurnsPerDay;
169
170     }
171
172     /**
173      * Overloaded constructor:
174      * Instantiate CinReader.class, DecimalFormat.class and Random.class and associate them with the current instance of
         Farm.class,
175      * and initialize variables.  Assign the new instance of Farm.class a name.
176      * @param strFarmName The name of the new farm.
177      */
178
179     public Farm( String strFarmName )   {
180
181         this();
182
183         this.strFarmName    = strFarmName;
184
185     }
186
187     /**
188      * Overloaded constructor:
189      * Instantiate CinReader.class, DecimalFormat.class and Random.class and associate them with the current instance of
         Farm.class,
190      * and initialize variables.  Assign the new instance of Farm.class a name and establish the max number of animals for the
         barn.
191      * @param strFarmName The name of the new farm.
192      * @param intStartingBarnSize The number of animals the starting barn can contain.
193      */
194
```

```
195     public Farm( String strFarmName, int intStartingBarnSize )  {
196
197         this();
198
199         this.strFarmName    = strFarmName;
200         createNewBarn( intStartingBarnSize );
201
202     }
203
204     /**
205      * Main menu
206      */
207
208     public void go()    {
209
210         while( true )   {
211
212             // Barn statistics reported to the player
213             System.out.print    ( "\nStatistics for '" + strFarmName + "':\n" );
214             System.out.printf   ( "Day:  %s | Total Days at Work:  %d\n", strCurrentDay, intNumberOfDays );
215             System.out.printf   ( "Current Funds:  $%s | Available Feed:  %d lbs.\n", currencyFormatter.format( dblMoney ),
                intAvailableFeed );
216
217
218             if( barnsOnFarm[intSelectedBarn] != null )  {
219
220                 System.out.print    ( "\nYou have the following animals in this barn:\n" );
221                 System.out.print    ( barnsOnFarm[intSelectedBarn] );
222
223             } else {
224
225                 System.out.printf   ( "\nYou are standing on plot #%d of your farm, which is currently empty.\n", (
                    intSelectedBarn + 1) );
226                 System.out.print    ( "The grass is very green and rich in nutrients.\n" );
227                 System.out.print    ( "One day you hope to build a barn here.\n" );
228
229             }
230
231             // Main menu
232             System.out.print    ( "\nPerform a task:\n" );
233             System.out.print    ( "\tFeed an (A)nimal\n" );
234             System.out.print    ( "\tVisit the (m)arket to sell produce and animals\n" );
235             System.out.print    ( "\tBuy (F)eed\n" );
236             System.out.print    ( "\tBuy a (N)ew Animal\n" );
237             System.out.print    ( "\tPrepare (p)roduce for sale\n" );
238             System.out.print    ( "\tPrepare anima(l) for sale\n" );
239             System.out.print    ( "\t(W)ork with the animals in this barn\n" );
240             System.out.print    ( "\tS(e)lect another Barn\n" );
241             System.out.print    ( "\tBuy a new (B)arn\n" );
242             System.out.print    ( "\tPac(k) animals in this barn\n" );
```

```
243             System.out.print    ( "\tFinish work for the day and (g)o to sleep\n" );
244             System.out.print    ( "\t\n" );
245             System.out.print    ( "\t(S)ave game\n" );
246             System.out.print    ( "\t(Q)uit game\n" );
247             System.out.printf   ( "What would you like to do (you have %d turns left for today):  ", intRemainingTurns );
248             chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
249
250             switch( chrUserInput )  {
251
252                 // Feed an animal
253                 case 'a':
254                     feedAnimal();
255                     break;
256
257                 // Visit the market to sell produce and animals
258                 case 'm':
259                     sellAtMarket();
260                     break;
261
262                 // Buy feed
263                 case 'f':
264                     buyFeed();
265                     break;
266
267                 // Buy a new animal
268                 case 'n':
269                     buyAnimal();
270                     break;
271
272                 // Prepare produce for sale
273                 case 'p':
274                     prepareProduceForSale();
275                     break;
276
277                 // Prepare animal for sale
278                 case 'l':
279                     prepareAnimalForSale();
280                     break;
281
282                 // Work with the animals in this barn
283                 case 'w':
284                     selectAnimal();
285                     break;
286
287                 // Select another barn
288                 case 'e':
289                     selectBarn();
290                     break;
291
292                 // Buy a new barn
```

```
293                 case 'b':
294                     buyBarn();
295                     break;
296 
297                 // Pack animals in this barn
298                 case 'k':
299                     packSelectedBarn();
300                     break;
301 
302                 // Finish work for the day and go to sleep
303                 case 'g':
304                     finishDay();
305                     break;
306 
307                 // Save the current game (currently not functioning)
308                 case 's':
309                     System.out.print    ( "\nYou have chosen to save your game.\n" );
310                     System.out.print    ( "What would you like to name the file?  ");
311                     FarmStorage.storeFarm( this, driverKeyboard.readString() );
312                     break;
313 
314                 // Quit the game
315                 case 'q':
316                     System.out.print    ( "\nThanks for playing FarmSim!\n\n" );
317                     break;
318 
319                 // Incorrect entries cause menu loop to start again
320                 default:
321                     System.out.print    ( "You entered '" + chrUserInput + "' which is not a valid option.\n\n" );
322                     pressEnterToContinue();
323                     continue;
324 
325             }
326 
327             // Exit loop to quit game
328             if( chrUserInput == 'q' )   {   break;  };
329 
330             if( intRemainingTurns == 0 )    {
331 
332                 intNumberOfDays++;
333                 intRemainingTurns = intMaxTurnsPerDay;
334 
335                 strCurrentDay   = changeDay( intNumberOfDays );
336 
337                 if( strCurrentDay.equals("Sunday") )    {
338 
339                     // Set new prices for animals, produce and feed every Sunday
340                     dblFeedPrice    = changePrice( dblFeedPrice, STARTING_FEED_PRICE );
341 
342                     dblCowPrice     = changePrice( dblCowPrice, STARTING_COW_PRICE );
```

```
                    dblChickenPrice = changePrice( dblChickenPrice, STARTING_CHICKEN_PRICE );
                    dblTurkeyPrice  = changePrice( dblTurkeyPrice, STARTING_TURKEY_PRICE );
                    dblHorsePrice   = changePrice( dblHorsePrice, STARTING_HORSE_PRICE );

                    dblCowMilkPrice         = changePrice( dblCowMilkPrice, STARTING_COW_MILK_PRICE );
                    dblCowMeatPrice         = changePrice( dblCowMeatPrice, STARTING_COW_MEAT_PRICE );
                    dblChickenEggPrice      = changePrice( dblChickenEggPrice, STARTING_CHICKEN_EGG_PRICE );
                    dblChickenMeatPrice     = changePrice( dblChickenMeatPrice, STARTING_CHICKEN_MEAT_PRICE );
                    dblTurkeyMeatPrice      = changePrice( dblTurkeyMeatPrice, STARTING_TURKEY_MEAT_PRICE );
                    dblHorseTransportationPrice = changePrice( dblHorseTransportationPrice, STARTING_HORSE_TRANSPORTATION_PRICE );

                }

                System.out.print    ( "\nWhew!  Farming is hard work!  You're getting really tired.\n" );
                System.out.print    ( "Now is a good time to turn in for the day.\n" );

                pressEnterToContinue();

            }

        }


    }

    /**
     * Submenu to feed animals in the currently selected barn.
     *
     */

    private void feedAnimal()   {

        // The variable blnAnimalSelected is "true" when an animal has already been selected in another part of the program
        // NOTE:  "Not enough turns" message is printed from the method checkTurnCost when it returns false
        if( ( checkTurnCost( TURN_COST_FEED_ANIMAL ) ) || ( blnAnimalSelected ) )   {

            if( ( barnsOnFarm[intSelectedBarn] != null ) || ( blnAnimalSelected ) ) {

                if( barnsOnFarm[intSelectedBarn].getCurrentNumberOfAnimals() > 0 )  {

                    boolean blnCompleteLoop = false;
                    int intAnimalIndex      = 0;
                    while( true )   {

                        // Player has run out of feed
                        if( intAvailableFeed == 0 ) {

                            System.out.print    ( "\nYou don't have any more feed for your animals!\n" );
                            System.out.print    ( "Go buy some more and then come back.\n" );
                            pressEnterToContinue();
```

```
393
394                         break;
395
396                     }
397
398                     // Prompt user to feed another animal in this barn (no turn penalty)
399                     // NOTE:  Prompt is suppressed if an animal has already been selected in another part of the program
400                     if( ( blnCompleteLoop ) && ( blnAnimalSelected == false ) ) {
401
402                         System.out.print    ( "\nWould you like to feed another animal in this barn? (y/n)  " );
403                         chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
404
405                         if( chrUserInput == 'n' )   {   break;  }
406
407                     }
408
409                     if( blnAnimalSelected == false )    {
410
411                         // Select an animal
412                         System.out.print    ( "\nTo stay healthy and happy, your animals should be fed only once per day.\n"
                            );
413                         showAnimals();
414
415                         System.out.print    ( "\nWhich animal would you like to feed?  " );
416
417                         intAnimalIndex  = (driverKeyboard.readInt( 1, barnsOnFarm[intSelectedBarn].getMaxNumberOfAnimals() )
                            - 1);
418
419                     } else {
420
421                         // Use currently selected animal, established in another part of the program
422                         intAnimalIndex  = intSelectedAnimal;
423
424                     }
425
426                     if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ) == null )  {
427
428                         System.out.print    ( "\nYou don't have an animal there.\n" );
429                         pressEnterToContinue();
430
431                     } else if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).getLastFeedDay() == intNumberOfDays
                        )   {
432
433                         System.out.print    ( "\nYou have already fed this animal today.\n" );
434                         System.out.printf   ( "Try again tomorrow.\n" );
435                         pressEnterToContinue();
436
437                     } else {
438
439                         // Prompt user to enter the amount of feed to give the animal
```

```
440                         System.out.print    ( "\nHow many pounds of feed would you like to give this animal?\n" );
441                         System.out.printf   ( "(max per day is %d)  ", barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex
                             ).getMaxFeedPerDay() );
442
443                         int intAmountOfFeedGiven    = driverKeyboard.readInt( 1, barnsOnFarm[intSelectedBarn].getAnimal(
                            intAnimalIndex ).getMaxFeedPerDay() );
444
445                         if( intAmountOfFeedGiven > intAvailableFeed )   {
446
447                             // Not enough feed available
448                             System.out.print    ( "\nYou don't have that much feed available.\n" );
449                             pressEnterToContinue();
450
451                         } else {
452
453                             barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).feedAnimal( intAmountOfFeedGiven,
                                intNumberOfDays );
454                             intAvailableFeed -= intAmountOfFeedGiven;
455
456                             System.out.printf   ( "\nAll done.  Your animals have already started eating.\n" );
457                             pressEnterToContinue();
458
459                         }
460
461                     }
462
463                     if( blnAnimalSelected == true ) {   break;  }
464
465                     // Loop has completed at least once
466                     blnCompleteLoop = true;
467
468                 }
469
470                 if( blnAnimalSelected == false )    {
471
472                     decrementRemainingTurns( TURN_COST_FEED_ANIMAL, true );
473
474                 }
475
476
477             } else {
478
479                 System.out.print    ( "\nYou don't have any animals here.\n" );
480                 pressEnterToContinue();
481
482                 decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
483
484             }
485
486         } else {
```

```
487
488                 System.out.print    ( "\nYou don't have a barn here.\n" );
489                 pressEnterToContinue();
490
491                 decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
492
493             }
494
495         }
496
497     }
498
499     /**
500      * Submenu to prepare animal produce for sale
501      */
502
503     private void prepareProduceForSale()    {
504
505         // The variable blnAnimalSelected is "true" when an animal has already been selected in another part of the program
506         if( checkTurnCost( TURN_COST_PREPARE_PRODUCE ) )    {
507
508             if( ( barnsOnFarm[intSelectedBarn] != null ) || ( blnAnimalSelected ) ) {
509
510                 if( barnsOnFarm[intSelectedBarn].getCurrentNumberOfAnimals() > 0 )  {
511
512                     boolean blnCompleteLoop = false;
513                     int intAnimalIndex      = 0;
514                     while( true )   {
515
516                         if( ( blnAnimalSelected == false ) && ( blnCompleteLoop ) ) {
517
518                             // Prompt user to prepare more produce for sale (without a turn penalty)
519                             System.out.print    ( "\nWould you like to prepare more produce for sale in this barn? (y/n)  " );
520
521                             chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
522
523                             if( chrUserInput == 'n' )   {   break;  }
524
525                         }
526
527                         if( blnAnimalSelected == false )    {
528
529                             // Select an animal
530                             showAnimals();
531                             System.out.print    ( "Which animal would you like to prepare produce for?  " );
532
533                             intAnimalIndex  = (driverKeyboard.readInt( 1, barnsOnFarm[intSelectedBarn].getMaxNumberOfAnimals() )
                            - 1);
534
535                         } else {
```

```
536
537                         // Animal has already been selected in another part of the program
538                         intAnimalIndex  = intSelectedAnimal;
539
540                     }
541
542                     if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ) == null )  {
543
544                         System.out.print    ( "\nYou don't have an animal there.\n" );
545                         pressEnterToContinue();
546
547                     } else {
548
549                         if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).getAnimalProduce() == null )   {
550
551                             // Some animals don't produce and can only be sold
552                             System.out.print    ( "\nThis animal doesn't produce and can only be sold.\n" );
553                             pressEnterToContinue();
554
555                         } else if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).getProduceAvailable() ) {
556
557                             // Prepare the maximum number of units produce to sell from the available from the currently
                                selected animal
558                             intNumberOfUnitsSold        = barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).
                                unitsOfProduceAvailable();
559                             dblAmountToEarn         = intNumberOfUnitsSold * currentValueOfProduceForSelectedAnimal(
                                barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).getAnimalProduce() );
560
561                             System.out.printf   ( "\nYou have prepared %d units of produce from this animal.\n",
                                intNumberOfUnitsSold );
562                             System.out.printf   ( "At current prices this will sell for $%s.\n", currencyFormatter.format(
                                dblAmountToEarn ) );
563                             System.out.print    ( "This produce will be sold the next time you visit the market.\n" );
564                             pressEnterToContinue();
565
566                             // Add this total to the total amount of money to give the player the next time she visits the
                                market
567                             dblMoneyToEarnOnNextSell    += dblAmountToEarn;
568                             barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).unitsOfProduceSold( intNumberOfUnitsSold
                                 );
569
570                         } else {
571
572                             System.out.print( "\nThis animal has no produce available to be sold.\n" );
573                             pressEnterToContinue();
574
575                         }
576
577                     }
578
```

```
579                         if( blnAnimalSelected == true ) {   break;  }
580
581                         blnCompleteLoop     = true;
582
583                     }
584
585                     if( blnAnimalSelected == false )    {   decrementRemainingTurns( TURN_COST_PREPARE_PRODUCE, true ); }
586
587                 } else {
588
589                     System.out.print    ( "\nYou don't have any animals here.\n" );
590                     pressEnterToContinue();
591
592                     decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
593
594                 }
595
596             } else {
597
598                 System.out.print    ( "\nYou don't have a barn here.\n" );
599                 pressEnterToContinue();
600
601                 decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
602
603             }
604
605         }
606
607     }
608
609     /**
610      * Submenu to prepare an animal for sale.
611      */
612
613     private void prepareAnimalForSale() {
614
615         // The variable blnAnimalSelected is "true" when an animal has already been selected in another part of the program
616         if( checkTurnCost( TURN_COST_PREPARE_PRODUCE ) )    {
617
618             if( ( barnsOnFarm[intSelectedBarn] != null ) || ( blnAnimalSelected ) ) {
619
620                 if( barnsOnFarm[intSelectedBarn].getCurrentNumberOfAnimals() > 0 )  {
621
622                     boolean blnCompleteLoop = false;
623                     int intAnimalIndex      = 0;
624                     while( true )   {
625
626                         if( ( blnAnimalSelected == false ) && ( blnCompleteLoop ) ) {
627
628                             // Prompt user to prepare more animals for sale (without a turn penalty)
```

```
629                         System.out.print    ( "\nWould you like to prepare more animals for sale in this barn? (y/n)  " );
630
631                         chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
632
633                         if( chrUserInput == 'n' )   {   break;  }
634
635                     }
636
637                     if( blnAnimalSelected == false )    {
638
639                         // Select an animal
640                         showAnimals();
641                         System.out.print    ( "Which animal would you like to prepare for sale?  " );
642
643                         intAnimalIndex  = (driverKeyboard.readInt( 1, barnsOnFarm[intSelectedBarn].getMaxNumberOfAnimals() )
                            - 1);
644
645                     } else {
646
647                         // Animal has already been selected in another part of the program
648                         intAnimalIndex  = intSelectedAnimal;
649
650                     }
651
652                     if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ) == null )  {
653
654                         System.out.print    ( "\nYou don't have an animal there.\n" );
655                         pressEnterToContinue();
656
657                     } else {
658
659                         if( barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).getMature() )  {
660
661                             // Tell the player the amount of money he will make if the animal is sold; provide option to not
                                sell
662                             dblAmountToEarn     = currentValueOfSelectedAnimal( barnsOnFarm[intSelectedBarn].getAnimal(
                                intAnimalIndex ).getAnimalType() );
663                             System.out.printf   ( "At current prices this animal will sell for $%s.\n", currencyFormatter.
                                format( dblAmountToEarn ) );
664                             System.out.print    ( "Are you sure you want to ready this animal for sale? (y/n)  " );
665
666                             chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
667
668                             if( chrUserInput == 'y' )   {
669
670                                 // For future implementation
671                                 //barnsOnFarm[intSelectedBarn].getAnimal( intAnimalIndex ).setAnimalPreparedToSell( true );
672
673                                 // Remove animal from barn
674                                 barnsOnFarm[intSelectedBarn].removeAnimal( intAnimalIndex, false );
```

```
675                                 // Add amount to money to be paid the next time the player visits the market
676                                 dblMoneyToEarnOnNextSell += dblAmountToEarn;
677
678                                 System.out.print    ( "\nYou have removed this animal from your barn.\n" );
679                                 System.out.print    ( "It will be sold the next time you visit the market.\n" );
680                                 pressEnterToContinue();
681
682                             }
683
684                         } else {
685
686                             System.out.print    ( "\nThis animal is not mature enough to be sold.\n" );
687                             System.out.print    ( "You must continue to feed it until it is full grown.\n" );
688                             pressEnterToContinue();
689
690                         }
691
692                     }
693
694                     if( blnAnimalSelected == true ) {   break;  }
695
696                     // Loop has completed at least once
697                     blnCompleteLoop     = true;
698
699                 }
700
701                 if( blnAnimalSelected == false )    {   decrementRemainingTurns( TURN_COST_PREPARE_ANIMAL, true );  }
702
703             } else {
704
705                 System.out.print    ( "\nYou don't have any animals here.\n" );
706                 pressEnterToContinue();
707
708                 decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
709
710             }
711
712         } else {
713
714             System.out.print    ( "\nYou don't have a barn here.\n" );
715             pressEnterToContinue();
716
717             decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
718
719         }
720
721     }
722
723 }
724
```

```
725         /**
726          * Subroutine to allow the player to visit a "market" to sell produce and animals.
727          */
728 
729         private void sellAtMarket() {
730 
731             if( checkTurnCost( intMaxTurnsPerDay ) )    {
732 
733                 System.out.print    ( "\nTime to load everything up and head to the market...\n" );
734                 decrementRemainingTurns( intMaxTurnsPerDay, true );
735 
736                 System.out.printf   ( "\nYou earned $%s at the market today!\n", currencyFormatter.format( dblMoneyToEarnOnNextSell )
                     );
737                 pressEnterToContinue();
738 
739                 // Pay player for total produce prepared for sale on the farm since the last visit to the market
740                 dblMoney += dblMoneyToEarnOnNextSell;
741 
742                 // Reset amount to pay the player back to zero
743                 dblMoneyToEarnOnNextSell    = 0.00;
744 
745             }
746 
747         }
748 
749         /**
750          * Submenu to allow the player to purchase additional feed to help his animals and make produce.
751          */
752 
753         private void buyFeed()  {
754 
755             if( checkTurnCost( TURN_COST_BUY_FEED ) )   {
756 
757                 int intAmountOfFeed;
758                 double dblTotalCost;
759 
760                 System.out.printf   ( "\nThe current price for feed is $%s per pound.\n", currencyFormatter.format( dblFeedPrice ) );
761                 System.out.printf   ( "You currently have %d pounds of feed at your farm.\n", intAvailableFeed );
762                 System.out.print    ( "\nHow many pounds would you like to buy?  " );
763 
764                 intAmountOfFeed = driverKeyboard.readInt( 1, 2147483647 );
765                 // Cost of feed
766                 dblTotalCost    = intAmountOfFeed * dblFeedPrice;
767 
768                 if( ( intAmountOfFeed > 0 ) && ( dblTotalCost <= dblMoney ) )   {
769 
770                     System.out.printf   ( "%d pounds of feed will cost $%s.  Are you sure? (y/n)  ", intAmountOfFeed,
                         currencyFormatter.format( dblTotalCost ) );
771                     chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
772 
```

```
                if( chrUserInput == 'y' )   {

                    // Subtract cost of feed from total money
                    dblMoney            -= dblTotalCost;
                    intAvailableFeed    += intAmountOfFeed;

                    System.out.printf   ( "\n%d pounds of feed has been delivered to your farm.\n", intAmountOfFeed );
                    pressEnterToContinue();

                    decrementRemainingTurns( TURN_COST_BUY_FEED, true );

                } else {

                    System.out.print    ( "Undecided?  Come back whenever you want.\n" );
                    pressEnterToContinue();

                    decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );

                }

            } else {

                // Player does not have enough money
                System.out.print    ( "Sorry, you don't have enough money.  Come back later.\n\n" );
                pressEnterToContinue();

                decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );

            }

        }

    }

    /**
     * Submenu to purchase a new animal and add it to the currently selected farm.
     */

    private void buyAnimal()    {

        // The variable blnAnimalSelected is "true" when an animal has already been selected in another part of the program
        if( ( checkTurnCost( TURN_COST_BUY_ANIMAL ) ) || ( blnAnimalSelected ) )    {

            if( ( barnsOnFarm[intSelectedBarn] != null ) || ( blnAnimalSelected ) ) {

                if( checkForBarnVacancy() ) {

                    String strAnimalType                    = "";
                    String strAnimalProduce                 = "";
                    String strSellAnimalProduce             = "";
```

```
823                     double dblAnimalPrice                   = 0.00;
824                     int intAnimalFeedNeededToProduce        = 0;
825                     int intAnimalFeedNeededFullGrown        = 0;
826                     int intMaxFeedPerDay                    = 0;
827
828                     while( true )   {
829
830                         System.out.print    ( "\nThe current prices for barn animals are as follows:\n" );
831                         System.out.printf   ( "\t(C)ow:  $%s for one cow\n", currencyFormatter.format( dblCowPrice ) );
832                         System.out.printf   ( "\tCh(i)ckens:  $%s for two dozen chickens\n", currencyFormatter.format(
                            dblChickenPrice ) );
833                         System.out.printf   ( "\t(T)urkeys:  $%s for six turkeys\n", currencyFormatter.format( dblTurkeyPrice ) );
834                         System.out.printf   ( "\t(H)orses:  $%s for one horse\n", currencyFormatter.format( dblHorsePrice ) );
835                         System.out.printf   ( "\n\t(R)eturn to the main menu...\n" );
836
837                         System.out.print    ( "\nWhat kind of animal would you like to buy today?  " );
838
839                         chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
840
841                         switch( chrUserInput )  {
842
843                             // Buy a cow
844                             case 'c':
845                                 dblAnimalPrice                  = dblCowPrice;
846
847                                 strAnimalType                   = "Cow";
848                                 strAnimalProduce                = "Milk";
849                                 strSellAnimalProduce            = "Beef";
850                                 intAnimalFeedNeededToProduce    = FEED_NEEDED_TO_PRODUCE_COW;
851                                 intAnimalFeedNeededFullGrown    = FEED_NEEDED_FULL_GROWN_COW;
852                                 intMaxFeedPerDay                = MAX_FEED_PER_DAY_COW;
853
854                                 break;
855
856                             // Buy some chickens
857                             case 'i':
858                                 dblAnimalPrice                  = dblChickenPrice;
859
860                                 strAnimalType                   = "Chicken";
861                                 strAnimalProduce                = "Eggs";
862                                 strSellAnimalProduce            = "Chicken Meat";
863                                 intAnimalFeedNeededToProduce    = FEED_NEEDED_TO_PRODUCE_CHICKEN;
864                                 intAnimalFeedNeededFullGrown    = FEED_NEEDED_FULL_GROWN_CHICKEN;
865                                 intMaxFeedPerDay                = MAX_FEED_PER_DAY_CHICKEN;
866
867                                 break;
868
869                             // Buy some turkeys
870                             case 't':
871                                 dblAnimalPrice                  = dblTurkeyPrice;
```

```
872
873                             strAnimalType                   = "Turkey";
874                             strAnimalProduce                = null;
875                             strSellAnimalProduce            = "Turkey Meat";
876                             intAnimalFeedNeededToProduce    = FEED_NEEDED_TO_PRODUCE_TURKEY;
877                             intAnimalFeedNeededFullGrown    = FEED_NEEDED_FULL_GROWN_TURKEY;
878                             intMaxFeedPerDay                = MAX_FEED_PER_DAY_TURKEY;
879
880                             break;
881
882                         // Buy a horse
883                         case 'h':
884                             dblAnimalPrice                  = dblHorsePrice;
885
886                             strAnimalType                   = "Horse";
887                             strAnimalProduce                = null;
888                             strSellAnimalProduce            = "Transportation";
889                             intAnimalFeedNeededToProduce    = FEED_NEEDED_TO_PRODUCE_HORSE;
890                             intAnimalFeedNeededFullGrown    = FEED_NEEDED_FULL_GROWN_HORSE;
891                             intMaxFeedPerDay                = MAX_FEED_PER_DAY_HORSE;
892
893                             break;
894
895                         case 'r':
896                             break;
897
898                         default:
899                             System.out.print    ( "You entered '" + chrUserInput + "' which is not a valid option.\n\n" );
900                             pressEnterToContinue();
901                             continue;
902
903                     }
904
905                     if( chrUserInput == 'r' )   {
906
907                         decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
908                         break;
909
910                     }
911
912                     // Intermediate funds value:  subtract cost of animal from total money
913                     double dblMoneyRemaining    = (dblMoney - dblAnimalPrice);
914                     if( dblMoneyRemaining >= 0 )    {
915
916                         // Print the cost and allow the player one last chance to not buy
917                         System.out.printf   ( "\nBuying this animal will leave you with $%s leftover.\n", currencyFormatter.
                            format( dblMoneyRemaining ) );
918                         System.out.printf   ( "Are you sure you want to buy this animal?  (y/n)  " );
919                         chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
920
```

```
                    if( chrUserInput == 'y' )   {

                        // Make intermediate funds value permanent total funds value
                        dblMoney = dblMoneyRemaining;

                        boolean blnAnimalDelivered  = false;
                        while( blnAnimalDelivered == false )    {

                            // Show the player the current vacancies in her barn
                            System.out.print    ( "\nYou currently have the following vacancies:\n" );

                            int intBarnIndex    = 0;
                            int intAnimalIndex  = 0;
                            for( Barn barn : barnsOnFarm )  {

                                intBarnIndex++;

                                if( barn != null )  {

                                    intAnimalIndex = 0;

                                    for( Animal animal : barn.someAnimals ) {

                                        intAnimalIndex++;

                                        if( animal == null )    {

                                            System.out.printf   ( "\tBarn [%d] has a vacancy in fenced area [%d].\n",
                                            intBarnIndex, intAnimalIndex );

                                        }

                                    }

                                }

                            }

                            while( blnAnimalDelivered == false )    {

                                System.out.print    ( "\nWhich barn should we make the delivery to?  " );

                                // Allow player to select where to deliver the animal
                                intBarnIndex = (driverKeyboard.readInt( 1, MAX_NUMBER_OF_BARNS ) - 1);
                                if( barnsOnFarm[intBarnIndex] != null ) {

                                    System.out.print    ( "And in which fenced area?  " );

                                    intAnimalIndex = (driverKeyboard.readInt( 1, barnsOnFarm[intBarnIndex].someAnimals.
                                    length ) - 1);
```

```
969                                 if( barnsOnFarm[intBarnIndex].someAnimals[intAnimalIndex] == null ) {
970
971                                     System.out.print    ( "\nOkay!  We'll take care of it right away.\n" );
972
973                                     barnsOnFarm[intBarnIndex].putAnimal( new Animal( strAnimalType, strAnimalProduce,
                                         strSellAnimalProduce, intAnimalFeedNeededToProduce, intAnimalFeedNeededFullGrown
                                        , intMaxFeedPerDay ), intAnimalIndex );
974                                     // Animal has been delivered; exit containing loops
975                                     blnAnimalDelivered = true;
976                                     pressEnterToContinue();
977
978                                     decrementRemainingTurns( TURN_COST_BUY_ANIMAL, true );
979
980                                 } else {
981
982                                     System.out.print    ( "\nThat space is not vacant.\n" );
983                                     pressEnterToContinue();
984
985                                 }
986
987
988
989                             } else {
990
991                                 System.out.print    ( "\nYou don't have a barn there.\n" );
992                                 pressEnterToContinue();
993                                 // "continue" handled by if statement below
994
995                             }
996
997                             // When a barn does not exist at the supplied location OR the space is already filled by
                            another animal
998                             if( ( barnsOnFarm[intBarnIndex] == null ) || ( barnsOnFarm[intBarnIndex].someAnimals[
                            intAnimalIndex] != null ) )    {
999
1000                                break;
1001
1002                            }
1003
1004                        }
1005
1006                    }
1007
1008                } else {
1009
1010                    System.out.print    ( "\nDoesn't seem like you're ready to buy.  Maybe next time.\n" );
1011                    pressEnterToContinue();
1012
1013                    decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1014
```

```
                            }

                            break;

                        } else {

                            System.out.printf   ( "You don't have enough money!  Come back later.\n" );
                            pressEnterToContinue();

                            decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
                            break;

                        }

                    }


                } else {

                    System.out.print    ( "\nYou don't have any spaces available for new animals.\n" );
                    pressEnterToContinue();

                }

            } else {

                System.out.print    ( "\nYou don't have a barn here.\n" );
                pressEnterToContinue();

                decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );

            }

        }

    }

    /**
     * Submenu to move an animal from one barn to another (currently unused; for future implementation)
     */

    private void moveAnimal()   {

        if( checkTurnCost( TURN_COST_MOVE_ANIMAL ) )    {


        }

    }

```

```
1065        /**
1066         * Submenu to select a particular animal in the currently selected barn and perform many tasks like prepare produce for
             sale, prepare animal for sale,
1067         * and feed the animal.
1068         */
1069
1070        private void selectAnimal() {
1071
1072            if( checkTurnCost( TURN_COST_SELECT_ANIMAL ) )  {
1073
1074                if( barnsOnFarm[intSelectedBarn] != null )  {
1075
1076                    if( barnsOnFarm[intSelectedBarn].getCurrentNumberOfAnimals() > 0 )  {
1077
1078                        boolean blnCompleteLoop     = false;
1079                        while( true )   {
1080
1081                            if( blnCompleteLoop )   {
1082
1083                                // Allow player to perform more tasks in this barn without a turns penalty
1084                                System.out.print    ( "\nWould you like to perform another task in this barn? (y/n)  " );
1085
1086                                chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
1087
1088                                // Exit loop and go back to main menu
1089                                if( chrUserInput == 'n' )   {   break;  }
1090
1091                            }
1092
1093                            // Select an animal
1094                            showAnimals();
1095                            System.out.print    ( "\nWhich animal would you like to select?  " );
1096                            intSelectedAnimal = (driverKeyboard.readInt( 1, barnsOnFarm[intSelectedBarn].getMaxNumberOfAnimals() ) -
                            1);
1097
1098                            if( barnsOnFarm[intSelectedBarn].getAnimal( intSelectedAnimal ) == null )   {
1099
1100                                System.out.print    ( "\nYou don't have an animal there.\n" );
1101                                pressEnterToContinue();
1102
1103                                continue;
1104
1105                            } else {
1106
1107                                blnAnimalSelected   = true;
1108
1109                            }
1110
1111                            // Tasks to be performed on the animals in the currently selected barn
1112                            System.out.print    ( "\nWhat would you like to do?\n" );
```

```
1113                    System.out.print    ( "\t(F)eed this animal\n" );
1114                    System.out.print    ( "\t(P)repare produce for sale\n" );
1115                    System.out.print    ( "\tReady (a)nimal for sale\n" );
1116                    System.out.print    ( "\n\t(R)eturn to the main menu...\n" );
1117
1118                    System.out.print    ( "\nWhat kind of animal would you like to buy today?  " );
1119
1120                    chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
1121
1122                    switch( chrUserInput )  {
1123
1124                        // Feed this animal
1125                        case 'f':
1126                            feedAnimal();
1127                            break;
1128
1129                        // Prepare this animal produce for sale
1130                        case 'p':
1131                            prepareProduceForSale();
1132                            break;
1133
1134                        // Prepare this animal for sale
1135                        case 'a':
1136                            prepareAnimalForSale();
1137                            break;
1138
1139                        // Return to the main menu
1140                        case 'r':
1141                            break;
1142
1143                        default:
1144                            System.out.print    ( "You entered '" + chrUserInput + "' which is not a valid option.\n\n" );
1145                            pressEnterToContinue();
1146                            continue;
1147
1148                    }
1149
1150                    if( chrUserInput == 'r' )   {   break;  }
1151
1152                    blnCompleteLoop     = true;
1153
1154                }
1155
1156                decrementRemainingTurns( TURN_COST_SELECT_ANIMAL, true );
1157
1158                intSelectedAnimal       = -1;
1159                blnAnimalSelected       = false;
1160
1161
1162            } else {
```

```
1163
1164                    System.out.print    ( "\nYou don't have any animals here.\n" );
1165                    pressEnterToContinue();
1166
1167                    decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1168
1169                }
1170
1171            } else {
1172
1173                System.out.print    ( "\nYou don't have a barn here!\n" );
1174
1175                pressEnterToContinue();
1176                decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1177
1178            }
1179
1180        }
1181
1182    }
1183
1184    /**
1185     * Select a new part of the farm's property (allows the player to go from one barn to another).
1186     */
1187
1188    private void selectBarn()   {
1189
1190        if( checkTurnCost( TURN_COST_SELECT_BARN ) )    {
1191
1192            // Cycle through the farm's plots of land and display barn and animal information to the player
1193            int intBarnCount = 0;
1194            for( Barn barn : barnsOnFarm )  {
1195
1196                intBarnCount++;
1197
1198                if( barn != null )  {
1199
1200                    System.out.printf   ( "Barn on plot of land #%d:\n", intBarnCount );
1201                    System.out.print    ( barn );
1202
1203                } else {
1204
1205                    System.out.printf   ( "Vacant plot of land #%d.\n", intBarnCount );
1206
1207                }
1208
1209            }
1210
1211            // Select another plot of land/barn
1212            System.out.print    ( "\nWhich part of your farm would like to do some work at?  " );
```

```
1213
1214            intSelectedBarn = driverKeyboard.readInt( 1, MAX_NUMBER_OF_BARNS ) - 1;
1215
1216            decrementRemainingTurns( TURN_COST_SELECT_BARN, true );
1217
1218        }
1219
1220    }
1221
1222    /**
1223     * Submenu to allow the player to purchase a new barn.
1224     */
1225
1226    private void buyBarn()  {
1227
1228        if( checkTurnCost( intMaxTurnsPerDay ) )    {
1229
1230            int intCount = 0;
1231            for( Barn barn : barnsOnFarm )  {
1232
1233                if( barn == null )  {   intCount++; }
1234
1235            }
1236
1237            if( intCount == 0 ) {
1238
1239                System.out.print    ( "You don't have any more space on your farm for a new barn!" );
1240                System.out.print    ( "Try expanding a current barn or tearing down an old one to create more room." );
1241
1242            } else {
1243
1244                int intNewBarnSize      = 0;
1245                double dblBarnPrice     = 0.00;
1246
1247                while( true )   {
1248
1249                    // Menu for different sizes of barn to purchase
1250                    System.out.print    ( "\nThe current prices for new barns are:\n" );
1251                    System.out.printf   ( "\t(S)mall Barn:  $%s -- holds %d kinds of animals\n", currencyFormatter.format(
                        COST_SMALL_BARN ), SIZE_SMALL_BARN );
1252                    System.out.printf   ( "\t(M)edium Barn:  $%s -- holds %d kinds of animals\n", currencyFormatter.format(
                        COST_MEDIUM_BARN ), SIZE_MEDIUM_BARN );
1253                    System.out.printf   ( "\t(L)arge Barn:  $%s -- holds %d kinds of animals\n", currencyFormatter.format(
                        COST_LARGE_BARN ), SIZE_LARGE_BARN );
1254                    System.out.printf   ( "\n\t(R)eturn to the main menu...\n" );
1255
1256                    System.out.printf   ( "\nYou currently have space on your farm for %d more barns.\n", intCount );
1257                    System.out.print    ( "Which size barn would you like to buy?  " );
1258
1259                    chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
```

```
1260
1261                    switch( chrUserInput )  {
1262
1263                        // Buy a small barn
1264                        case 's':
1265                            intNewBarnSize      = SIZE_SMALL_BARN;
1266                            dblBarnPrice        = COST_SMALL_BARN;
1267                            break;
1268
1269                        // Buy a medium barn
1270                        case 'm':
1271                            intNewBarnSize      = SIZE_MEDIUM_BARN;
1272                            dblBarnPrice        = COST_MEDIUM_BARN;
1273                            break;
1274
1275                        // Buy a large barn
1276                        case 'l':
1277                            intNewBarnSize      = SIZE_LARGE_BARN;
1278                            dblBarnPrice        = COST_LARGE_BARN;
1279                            break;
1280
1281                        // Return to the main menu
1282                        case 'r':
1283                            break;
1284
1285                        default:
1286                            System.out.print    ( "You entered '" + chrUserInput + "' which is not a valid option.\n\n" );
1287                            pressEnterToContinue();
1288                            continue;
1289
1290                    }
1291
1292                    if( chrUserInput == 'r' )   {
1293
1294                        decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1295                        break;
1296
1297                    }
1298
1299                    // Intermediate funds value:  the amount of total money leftover if the barn is purchased
1300                    double dblMoneyRemaining    = (dblMoney - dblBarnPrice);
1301                    if( dblMoneyRemaining >= 0 )    {
1302
1303                        // Show the player the amount of money she will have if the barn is purchased and confirm she wants to buy
1304                        System.out.printf   ( "\nThis barn is going to leave you with $%s leftover.\n", currencyFormatter.format(
                             dblMoneyRemaining ) );
1305                        System.out.printf   ( "Are you sure you want to buy this barn?  (y/n)  " );
1306                        chrUserInput    = Character.toString( driverKeyboard.readChar() ).toLowerCase().charAt( 0 );
1307
1308                        if( chrUserInput == 'y' )   {
```

```
1309
1310                        // Make intermediate funds value the new value for total funds
1311                        dblMoney = dblMoneyRemaining;
1312
1313                        // Prompt user for where he would like to put the new barn
1314                        System.out.print    ( "\nWhere on your farm would you like to put your new barn?\n" );
1315
1316                        while( true )   {
1317
1318                            System.out.print    ( "\nThe following plots of land are available:\n" );
1319
1320                            intCount = 0;
1321                            for( Barn barn : barnsOnFarm )  {
1322
1323                                intCount++;
1324                                if( barn == null )  {
1325
1326                                    System.out.printf   ( "\t[%d] Empty\n", intCount );
1327
1328                                }
1329
1330                            }
1331
1332                            System.out.print    ( "Construct barn on plot # " );
1333
1334                            int intIndex = (driverKeyboard.readInt( 1, MAX_NUMBER_OF_BARNS ) - 1);
1335                            if( barnsOnFarm[intIndex] == null ) {
1336
1337                                barnsOnFarm[intIndex] = new Barn( intNewBarnSize );
1338                                intCurrentNumberOfBarns++;
1339                                System.out.printf   ( "\nCongratulations!  You have constructed a new barn on plot #%d!\n", (
                                    intIndex + 1) );
1340                                pressEnterToContinue();
1341
1342                                decrementRemainingTurns( /* TURN_COST_BUY_BARN */ intMaxTurnsPerDay, true );
1343                                break;
1344
1345                            } else {
1346
1347                                System.out.print    ( "You already have a barn on that plot of land.  Try again.\n" );
1348                                pressEnterToContinue();
1349                                continue;
1350
1351                            }
1352
1353                        }
1354
1355                    } else {
1356
1357                        System.out.print    ( "\nNot so sure you want to buy a barn today?  Come back anytime.\n" );
```

```
1358                            pressEnterToContinue();
1359
1360                            decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1361
1362                        }
1363
1364                        break;
1365
1366                    } else {
1367
1368                        System.out.printf   ( "You don't have enough money!  Come back later.\n" );
1369                        pressEnterToContinue();
1370
1371                        decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1372                        break;
1373
1374                    }
1375
1376
1377                }
1378
1379            }
1380
1381        }
1382
1383    }
1384
1385    /**
1386     * Submenu to allow the player to expand a barn and increase the number of animals it can contain (currently unused; for
        future implementation)
1387     */
1388
1389    private void expandBarn()   {
1390
1391        if( checkTurnCost( /* TURN_COST_EXPAND_BARN */ intMaxTurnsPerDay ) )    {
1392
1393
1394
1395        }
1396
1397    }
1398
1399    /**
1400     * Submenu to allow the player to remove a barn from his farm (currently unused; for future implementation)
1401     */
1402
1403    private void tearDownBarn() {
1404
1405        if( checkTurnCost( /* TURN_COST_TEAR_DOWN_BARN */ intMaxTurnsPerDay ) ) {
1406
```

```
1407
1408
1409            }
1410
1411
1412        }
1413
1414        /**
1415         * Finish the current day by disposing of any remaining turns
1416         */
1417
1418        private void finishDay()    {
1419
1420            decrementRemainingTurns( intRemainingTurns, false );
1421
1422        }
1423
1424        /**
1425         * Console prompt for the player to press the enter button to continue playing
1426         */
1427
1428        private void pressEnterToContinue() {
1429
1430            System.out.print( "Press enter to continue..." );
1431            driverKeyboard.readString();
1432
1433        }
1434
1435        /**
1436         * Create a new instance of Barn.class with a supplied capacity of animals.  Instance is stored within an element of
             barnsOnFarm array.
1437         *
1438         * @param intAnimalCapacity the max number of animals that can be contained within this new instance of Barn.class
1439         */
1440
1441        private void createNewBarn( int intAnimalCapacity ) {
1442
1443            // Create a new barn
1444            barnsOnFarm[intCurrentNumberOfBarns] = new Barn( intAnimalCapacity );
1445            // Increment the helper variable for the partially filled array barnsOnFarm everytime a new barn is purchased
1446            intCurrentNumberOfBarns++;
1447
1448        }
1449
1450        /**
1451         * Check if the player has enough turns left for the requested task.  Returns true if enough turns remain, but returns false
             and prints a message
1452         * to the console if the player does not have enough left in the current day.
1453         *
1454         * @param intTurnCost The number of turns it will cost to perform a given task on the farm.
```

```
1455        * @return Boolean value indicating whether the player has enough turns left to continue.
1456        */
1457
1458       private boolean checkTurnCost( int intTurnCost )    {
1459
1460           if( intTurnCost <= intRemainingTurns )  {
1461
1462               return true;
1463
1464           } else {
1465
1466               System.out.print    ( "\nYou don't have enough turns left for today.\n" );
1467               System.out.print    ( "Try again tomorrow after you get some rest.\n" );
1468
1469               pressEnterToContinue();
1470
1471               return false;
1472
1473           }
1474
1475       }
1476
1477       /**
1478        * Decrements the number of turns the player has left after a task has been completed.
1479        *
1480        * @param intTurnCost The number of turns required for a task.
1481        * @param blnDisplayMessage Displays a message to the console communicating to the user the number of turns that were used
           when true; suppresses
1482        * when false.
1483        */
1484
1485       private void decrementRemainingTurns( int intTurnCost, boolean blnDisplayMessage )  {
1486
1487           // Subtract turns from total turns remaining
1488           intRemainingTurns -= intTurnCost;
1489
1490           if( blnDisplayMessage ) {
1491
1492               System.out.printf   ( "\n[ -- You used %d turns -- ]\n", intTurnCost );
1493               pressEnterToContinue();
1494
1495           }
1496
1497       }
1498
1499       /**
1500        * Determine the current day of the week from the total number of days completed by the player.
1501        *
1502        * @param intDayOfWeek The total number of days completed by the player.
1503        */
```

```
1504
1505        private String changeDay( int intDayOfWeek )    {
1506
1507            switch( (intDayOfWeek % 7) )    {
1508
1509                case 0:
1510                    return "Sunday";
1511
1512                case 1:
1513                    return "Monday";
1514
1515                case 2:
1516                    return "Tuesday";
1517
1518                case 3:
1519                    return "Wednesday";
1520
1521                case 4:
1522                    return "Thursday";
1523
1524                case 5:
1525                    return "Friday";
1526
1527                case 6:
1528                    return "Saturday";
1529
1530            }
1531
1532            // Cannot happen
1533            return null;
1534
1535        }
1536
1537        /**
1538         * Dynamically alters a price by comparing the current value to the average price.  The larger the price gets the more
                likely it is to begin
1539         * decreasing in price the next time the method is called.  Conversely, the smaller the price gets the more likely it is to
                begin increasing in price
1540         * the next time the method is called.
1541         *
1542         * @param dblLastWeeksPrice The current value for the price.
1543         * @param dblStartingPrice The average price.
1544         * @return Double representing a new price.
1545         */
1546
1547        private double changePrice( double dblLastWeeksPrice, double dblStartingPrice ) {
1548
1549            // Pick a random number between 1 and 100
1550            int intPercentageNumerator = (randomNumber.nextInt( 99 ) + 1);
1551
```

```
1552            // Coin flip as to whether the price should increase or decrease
1553            if( randomNumber.nextInt( 2 ) == 0 )    {   intPercentageNumerator *= -1;   }
1554
1555            // If amount is to decrease and the value is already less than the typical starting price, increase the probability of
                the price being increased
1556            if( ( dblLastWeeksPrice < dblStartingPrice ) && ( intPercentageNumerator < 0 ) )    {
1557
1558                if( ( (randomNumber.nextInt( 999 ) + 1) > (((dblStartingPrice - dblLastWeeksPrice) / dblStartingPrice) * 1000) ) ==
                    false ) {
1559
1560                    intPercentageNumerator *= -1;
1561
1562                }
1563
1564            }
1565
1566            // If amount is to increase and the value is already more than the typical starting price, increase the probability of
                the price being decreased
1567            if( ( dblLastWeeksPrice < dblStartingPrice ) && ( intPercentageNumerator < 0 ) )    {
1568
1569                if( ( (randomNumber.nextInt( 999 ) + 1) > (((dblLastWeeksPrice - dblStartingPrice) / dblStartingPrice) * 1000) ) ==
                    false ) {
1570
1571                    intPercentageNumerator *= -1;
1572
1573                }
1574
1575            }
1576
1577            // Return the new price
1578            return dblLastWeeksPrice + (dblStartingPrice * COST_MODIFIER * ((float) intPercentageNumerator / 100.00f));
1579
1580        }
1581
1582        /**
1583         * Verifies that the currently selected barn contains at least one animal within it.
1584         *
1585         * @return Boolean indicating whether an animal exists in the given instance of Barn.class.
1586         */
1587
1588        private boolean checkForBarnVacancy()   {
1589
1590            int intBarnIndex    = 0;
1591            int intAnimalIndex  = 0;
1592
1593            for( Barn barn : barnsOnFarm )  {
1594
1595                intBarnIndex++;
1596                if( barn != null )  {
1597
```

```
1598                for( Animal animal : barn.someAnimals ) {
1599
1600                    intAnimalIndex++;
1601                    if( animal == null )    {   return true;    }
1602
1603                }
1604
1605            }
1606
1607        }
1608
1609        return false;
1610
1611    }
1612
1613    /**
1614     * Packs the someAnimals array for the currently selected instance of Barn.class.  Messages are printed to the console to
         indicate
1615     * whether or not the operation was successful.
1616     */
1617
1618    private void packSelectedBarn() {
1619
1620        if( barnsOnFarm[intSelectedBarn] != null )  {
1621
1622            if( barnsOnFarm[intSelectedBarn].getCurrentNumberOfAnimals() > 0 )  {
1623
1624                barnsOnFarm[intSelectedBarn].packBarn();
1625
1626                System.out.print    ( "\nYou have moved all of your animals to the front areas of this barn.\n" );
1627                pressEnterToContinue();
1628
1629                decrementRemainingTurns( TURN_COST_PACK_BARN, true );
1630
1631            } else {
1632
1633                System.out.print    ( "\nYou don't have any animals here.\n" );
1634                pressEnterToContinue();
1635
1636                decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1637
1638            }
1639
1640        } else {
1641
1642            System.out.print    ( "\nYou don't have a barn here!\n" );
1643
1644            pressEnterToContinue();
1645            decrementRemainingTurns( TURN_COST_INCOMPLETE_TASK, true );
1646
```

```
1647            }
1648
1649        }
1650
1651        /**
1652         * Return the current value of the type of produce for the currently selected instance of Animals.class (in the currently
         selected
1653         * instance of Barn.class).
1654         *
1655         * @param strAnimalProduce The kind of produce for the currently selected animal.
1656         * @return The current price for the produce of the currently selected animal.
1657         */
1658
1659        private double currentValueOfProduceForSelectedAnimal( String strAnimalProduce )    {
1660
1661            // --> rewrite this section using an enumeration
1662            if( strAnimalProduce.equals( "Milk" ) ) {
1663
1664                return dblCowMilkPrice;
1665
1666            }
1667
1668            if( strAnimalProduce.equals( "Eggs" ) ) {
1669
1670                return dblChickenEggPrice;
1671
1672            }
1673
1674            return 0.00;
1675
1676        }
1677
1678        /**
1679         * Return the current value of the kind of animal for the currently selected instance of Animals.class (in the currently
         selected
1680         * instance of Barn.class).
1681         *
1682         * @param strAnimalProduce The kind of animal for the currently selected animal.
1683         * @return The current price for the animal of the currently selected animal.
1684         */
1685
1686        private double currentValueOfSelectedAnimal( String strAnimalType ) {
1687
1688            if( strAnimalType.equals( "Cow" ) ) {
1689
1690                return dblCowMeatPrice;
1691
1692            }
1693
1694            if( strAnimalType.equals( "Chicken" ) ) {
```

```
1695
1696                return dblChickenMeatPrice;
1697
1698            }
1699
1700            if( strAnimalType.equals( "Turkey" ) )  {
1701
1702                return dblTurkeyMeatPrice;
1703
1704            }
1705
1706            if( strAnimalType.equals( "Horse" ) )   {
1707
1708                return dblHorseTransportationPrice;
1709
1710            }
1711
1712            return 0.00;
1713
1714        }
1715
1716        /**
1717         * Prints to the console a list of animals and vacancies in the currently selected instance of Barn.class.
1718         */
1719
1720        private void showAnimals()  {
1721
1722            System.out.print    ( "\nYou have the following animals in this barn:\n" );
1723            System.out.print    ( barnsOnFarm[intSelectedBarn] );
1724
1725        }
1726
1727    }
```