THE UNIVERSITY of NEW MEXICO

World Food Resource Management

Serious Game Project

Requirements for Milestone 2

version 1.5

Countries:

1) **Country List:** The user's unit of control will be the country. This game will use the 193 countries with full membership status within the United Nations system as of Dec 31, 2014 (see http://en.wikipedia.org/wiki/Member_states_of_the_United_Nations). This does not include, for example, Vatican City and Palestine (which the UN recognizes as "observer states"). Kosovo is recognized by 108 UN member states, but it is not itself a UN member state and, therefore, not in the list of 193 countries included in the model.

   The 193 countries and their game spellings are defined in the data file: **`countryData.cvs`** on the class website.

   The data file uses the spelling: "United States of America" (not United States), "China" (not People's Republic of China), Republic of Korea (not South Korea), and Democratic People's Republic of Korea (not North Korea).

2) **Country XML**: With a few changes, the areas XML file from milestone 1 becomes, the country XML in milestone 2**.** When the program starts, it must read all .xml files within the directory:

   <*projectroot*>/resources/areas/

   Each configuration file will be a root element tag: <country></country >. Within the root element will be a name tag and a series of <area> tags that is an ordered list of the area's vertices. The last vertex is assumed to be connected to the first. For example

```
<country>
  <name>countryName</name>
  <area>
     <vertex lat="value1" lon="value2"/>
     <vertex lat="value2" lon="value3"/>
     ...
  </area>
</country>
```

   There must be exactly one configuration file for each of the 193 countries included in the game. The file names (other than ending in .xml) are not specified. Every country must contain an area tag for each major continuous area of that country. What defines a "major area" is left to the development team's discretion. For some guidelines: In the United

States, there needs to be at least 4 areas: continental US, Alaska, Hawaii, and Puerto Rico. The development team may or may not choose to include separate areas for Guam, the U.S. Virgin Islands, and some other US territory and commonwealths. The development team will not want to include separate areas for each island in the U.S. Virgin Islands or in the Florida keys. Also left to the development team's discretion is whether to leave small islands as land drawn on the map, but unassociated with a country or to define the country's outlines to include water areas that encompass groups of islands with the mainland (or largest island).

Note: the number of vertices used to define an area should be vastly lower than what is shown on your rendered background country image of a coastline or country boarder.

If a configuration file contains malformed or impossible data, then the reader must:

a) Display a scrollable window that shows the offending data file.
b) Highlight the offending line and sets the scroll bars so that the line is visible.
c) Provide a helpful error message.
d) Allow the user to edit the data file and resave it.

Impossible data includes:

a) More than one name tag in a country tag.
b) A name tag not matching the spelling of the name in **`countryData.cvs`**.
c) Latitude value greater than +90 or less than -90.
d) Longitude value greater than +180 or less than -180.
e) Polygonal region with less than three vertices.
f) Polygonal region with intersecting edges.
g) Polygonal region that is degenerate (the vertices are collinear).

3) For each of the 193 countries, you must obtain **approximate** values for each of the following 2014 data:

Population Data:

a) Population (in people).
b) Median age of population (in years).
c) Birth Rate (total number of live births per 1,000 of a population in a year).
d) Mortality Rate (units of deaths per 1,000 individuals per year).
e) Net Migration Rate (immigration - emigration per 1,000 population).
f) Prevalence of Undernourishment (percentage of population).

Food Data:

a) Total Production, Total Exports, and Total Imports of Corn (in metric tons).
b) Total Production, Total Exports, and Total Imports of Wheat (in metric tons).
c) Total Production, Total Exports, and Total Imports of Rice (in metric tons).
d) Total Production, Total Exports, and Total Imports of Soy (in metric tons).

e) Sum Total Production, Sum Total Exports, and Sum Total Imports all other foods.

Land Data:

a) Total Land area (in square kilometers).
b) Land used for Corn (in square kilometers).
c) Land used for Wheat (in square kilometers).
d) Land used for Rice (in square kilometers).
e) Land used for Soy (in square kilometers).
f) Land used for all other foods (in square kilometers).
g) Arable land still open to cultivation (in square kilometers).
h) Cultivated Land using Organic, Conventional and GMO methods (each a percentages totaling 100%).

The data you obtain should be a zeroth-order approximation. For example, you might find that 22 nations produce 90% of all corn. A zeroth-order approximation would be to assume that all of the remaining nations produce the remaining 10% in proportion to their area (with perhaps some countries, totally outside the growing area of corn, set to zero production).

More accurate values of these data will be supplied by other classes participating in this project. The more accurate data may or may not arrive before the end of this milestone.

Data (production, consumption, population, ...) from all land areas is considered to be part of one of 193 countries modeled in this game. For example, data from Kosovo is aggregated with Serbia. Data from Taiwan is aggregated with China and data form Bermuda is aggregated with the United Kingdom.

All off the above data must be read from the **`.cvs`** file:
<*projectroot*>**`/resources/data/countryData.cvs`**

Each record of the **`countryData.cvs`** file is comma separated ASCII values. The first record is list of column headers. The second record is the data type of the column: **`"String"`**, **`"int"`** or **`"double"`**. All strings fields in the file must be within double quotes. All int and double fields most not be in quotes.

The first column of this file defines the country names that must be used in milestone 2 of this game. For this milestone, all other columns are blank and for the developer to populate as described above.

When reading the CSV file if the program encounters an error, the data must be displayed into a JTable with the offending line hightlighted and an hopefully helpful error message displayed. The user needs to be able to edit the data and save it to a file.

A record is in error if the country, population or landArea fields are blank.

All other fields, if blank are assumed unknown and your program must provide an educated guess at the values for unknown data. For this milestone, as we do not know how much of

this data will actually be blank by the end of the semester, these "educated guesses" should be very simplistic. For example, if the fields imports, exports and production are all blank, then you might assume the import and export values for each crop are zero and production values are sufficient to keep the country 's population at 5% undernourished. This is very simplistic, but not stupid.

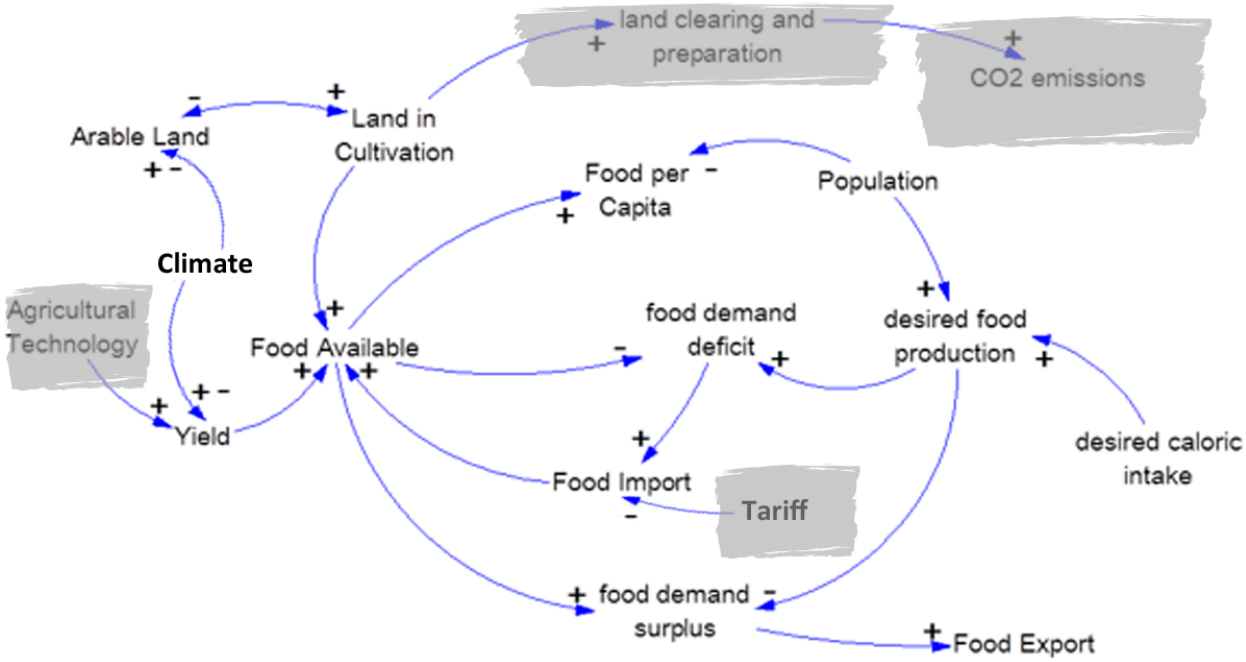
4) Country flag icons: Each country, when displayed on the map, must include its 23x15 pixel flag found on http://en.wikipedia.org/wiki/Member_states_of_the_United_Nations.

   These images must be **`.png`** and read from: <*projectroot*>**`/resources/flags/`**

5) Climate Change: This project will not model climate change, but will incorporate results of other climate change models. In this milestone, the primary source for climate change data is the Intergovernmental Panel on Climate Change Fifth Assessment Report (ipcc AR5): http://www.ipcc.ch/report/ar5/.

Assumptions:

1) All countries will maintain their current, 2014, borders through 2050.
2) The topology of all land will remain the same through 2050. If sea levels rise or lower, then less or more of that assumed fixed topology will be above sea level.
3) The model used it this game assumes the causal relationships in the image below. Arrows with (+) indicate a direct relationship and (-) indicates an indirect relationship.



The grayed relationships are not implemented in Milestone II.

Data Types:

User controllable country data: data that the user can input about a country. For this milestone, the only user controllable country data is the percent of arable land devoted to each of the five modeled crops (wheat, rice, corn and soy) and all other crops. Note: the total of these five percentages is the percentage of cultivated land in the country. In most cases, the total cultivated land will be less than 100% of the total arable land. Of course, the percentage of cultivated land can never be more than 100% of the total arable land.

Resultant country data: All other data in the required **`AbstractCountry`** class. For year 1, these are given values. For all future years, they are model results.

Scenario Assumptions data: This is all the data at higher resolution than country data used as input to each calculation of the next year's viewable country data. Examples are low/high seasonal temperatures, and rainfall. This data is based on externally supplied model results with various randomizations and probabilities applied.

Derived Internal States: This is data calculated by the model from the past year's resultant country data, the current year's user controllable country data and the scenario assumptions data that is used as intermediate steps to calculate the current year's resultant country data.

Interactive Output to the User:

1) **World Map:** The program will start by displaying the world map. This should be built on one of the selected Milestone 1 projects posted on the class website.
2) **Global Stats:** The GUI must at all times display:
   a. The current model year both as a number and as a timeline / progress bar.
   b. Current world population with percent happy people.
3) **Map Zoom:** The map must be smoothly zoomable from world scale down to one pixel being 2 kilometers on a side. Do not zoom past this level. This level will show our highest resolution climate data covering a 5 pixel $\times$ 5 pixel area. Much past this level would show too many features that are likely to change by 2050.
4) **Country Flag and Name:** The basic unit over which the user will have control is the country. At each scale of the map, all visible countries that are at least 50x25 pixels must display the country flag. Additionally, whenever a country is large enough, it must display both the flag and the country name as spelled in **`countryData.cvs`**. With continued zooming, no additional place or region names should display. All crop production data viewable by the user will be at the country granularity. Some internal parts of the model will need higher resolutions. For countries with discontinuous areas, center the label over the largest visible area. For example, if all of North America is visible, the label for the United States of America must be centered over the continental United States of America. If only part of United States of America is visible (Hawaii or a part of the continental US) then center the

label over that part. Exactly, what defines "center" is left to the development team's discretion, with the general guideline that an informed user viewing the map should not say: "that label is in the wrong place".

5) **Map Data Overlays:** The user must be able to select an overlay that displays any of the model's user *Controllable Country Data*, *Resultant Country Data* and *Scenario Assumptions data*. By contrast, *Derived Internal States* are not displayable to the user. Overlays must be applied to all visible map areas.

6) **Map Sea Level Overlay:** The Scenario Assumptions data will include sea level as rise in meters as a function of year from the 2014 level. The minimum value of rise by 2050 in a scenario will be 2 cm. The maximum value of rise in any scenario will be 10 meters. In this game, country boarders must remain constant through any rise in sea level, however, land area below sea level must be colored blue – a different shade than the ocean, but also different from any blue colors you use on land.

7) **Country Selection:** As in milestone 1, the users must be able to select a single country by clicking on the map. The user must also be able to select multiple countries. Multiple countries may be selected by click-and-drag, multi-click while depressing the ctrl or shift button, by menu (i.e. European Union Countries), by game controller, and/or by Wiimote. Note: I have a few Wiimotes with USB bluetooth connectors. Let me know if you want to try one out).

8) **Information Panel:** When a country is selected, its outline, flag, name, and *Resultant Country Data* must be displayed on some sort of information panel. Additionally, its User Controllable Country Data must be displayed within editable fields on the panel. When multiple countries are selected, the displayed numbers must be aggregated over the selected countries. Where it makes sense, the information panel must display the same data both as a quantity and as a percentage.

9) **Metric/English Units:** The user must be able to select data to be displayed in metric or English units.

Input form the User:

1) **Controllable Country Data:** For this milestone, the only user controllable country data is the percent of arable land devoted to each of the five modeled crops (wheat, rice, corn and soy) and all other crops for the upcoming year. This must be enterable form the information panel. If multiple countries are selected, then the change must apply to all of those countries.

2) **Run, Pause, Next Controls:** There must be a run simulation / pause button and a next year button. When the simulation is advanced one year, your interface must guarantee that no less than one second and no more than 10 seconds (on a moon.cs.unm.edu machine) pass before the next year's data is displayed. During this second, all GUI controls (except exit) must be disabled and some sort of progress bar or animation must display.

Climate, Crop, and Land Data:

1) **Data Resolution:** The models will be using data at the resolution of one point per one hundred square kilometers (10 km × 10 km). It is the other class's responsibility to supply us with this data, but in the meantime, each development team needs zero-order approximations. For example, if the best data found in a quick search is one point per 500 square kilometer, then a zero-order approximation can upsample the data using simple linear interpolation.

2) **Elevation Data:** Obtain approximate elevation data for all land areas. In milestone 2 of the game, elevation data will only be used to determine if a pixel is above or below sea level during each year of the game. Land elevation above 2014 sea level will be assumed to be constant throughout the game (no growing or flattened mountains). However, sea level will rise over the duration of the simulation so that some land that is above sea level in 2014 will not be above sea level in 2050.

3) **Temperature:** Obtain approximate temperature data as a function of time and location for all land areas. In particular, the model requires four temperature metrics:
   a) Annual maximum temperature.
   b) Annual minimum temperature.
   c) Average annual daytime temperature.
   d) Average annual nighttime temperature.

   When obtaining this data, keep in mind: 1) only zero-order approximation is needed for this milestone and 2) no one actually knows what these numbers will be in 2050.

   A good zero-order approximation is to find the known values for 2014, then find projected values for 2050 (or some year near that) and for maybe one other year about halfway between 2014 and 2050. It is best if different groups use different sources so we can see a range of possibilities play out. Just as linear interpolation is used to upsample in space, it

can also be used to approximate temperatures at the years between those for which you have found data.

For the final, end-of-semester rollout, we want to do better than this, but that is the job of the our partners in the other classes.

4) **Precipitation:** Obtain approximate precipitation data in cementers of rainfall per year as a function of time and location. As with the temperature data, interpolate as needed.

5) **Sea Level**: Obtain approximate sea level rise in centimeters as a function of time. Sea level rise, as it affects coastlines, is actually a function of both time and location as coastline geometry can greatly amplify tidal effects. However, in this game, it will be assumed that sea level rises evenly across the globe.

6) **Crop Zone Data**: For each of the modeled crops (wheat, corn, rice and soy) find approximate values of the crop's ideal range of:
   a) Annual maximum temperature (in degrees Celsius).
   b) Annual minimum temperature (in degrees Celsius).
   c) Average annual daytime temperature (in degrees Celsius).
   d) Average annual nighttime temperature (in degrees Celsius).
   e) Total annual precipitation (in centimeters of rainfall).

   For the fifth crop (everything else) assume that whatever are the country's current average values are optimal for that country's "everything else". This is a poor assumption as in large countries it averages very different climate zones and "other" includes widely different crops; however, it is a starting point.

   In milestone 2 of this game, it is assumed that crops can be planted in three different growth zone:

   Ideal Zone: All 5 land statistics are within the crop's ideal range. Crops planted in an ideal zone produce at 100% of that country's yield rate for that crop.

   Acceptable Zone: The land is outside the crop's ideal range in at least one of the crop's ideal requirements, but not more than ±30% out of the ideal range. Crops planted in this zone produce at 60% of that country's yield rate.

   Poor Zone: The land is outside the crop's ideal range by more than ±30% in at least one of the crop's ideal requirements. Crops in this zone produce 25% of that country's yield rate.

   For example, if rice's ideal precipitation range is 90 to 150 cm/year, then ±30% is:

   90 - (150-90)×30% through 150 + (150-90)×30% = 72 through 168 cm/year

   Of course, some countries have well developed infrastructures for irrigation, greenhouses and other environmental modifications. This will be reflected in the country's yield rate. See below for calculation of the country's yield rate.

Model Requirements at Start of Game:

1) **Randomization Percentage:** The programmer (not the user) sets the randomization percentage of the game. When this is set to zero, the game must be perfectly deterministic. This will be used for verification testing. A randomization percentage of 1.0 will be the maximum level of randomness.

2) **Base Sea Level Rise per Year:** In every scenario of this game, sea level will always rise monotonically from 2014 through 2050. The total rise from 2014 through 2050 must be at least 2.0 centimeters and no more than 1,000 centimeters. In this milestone, the developer should choose a total sea level rise that is within the range suggested by the same data source from which the developer obtained the 2050 global temperature data. The base sea level rise per year that will be used in a particular run of the model is then defined as:

$$baseSeaLevelRisePerYear = \frac{givenRise + randomizationPercentage \times (r1 - r2)}{(END\_YEAR - START\_YEAR)}$$

Where,

$givenRise$ is the value obtained from the developer's data source.
$r1$ and $r2$ are uniformly distributed random numbers from 0.0 through $0.5 \times givenRise$.
$START\_YEAR$ and END_YEAR are 2014 and 2050 respectively.

The above equation creates what is called a ***triangular distribution*** of noise with values near zero significantly more probable than values near the positive or negative extremes.

Note: this equation gives the ***base*** rise per year. The actual rise each year will vary from year to year. See the section "Model Requirements at Each Year" for details.

3) **Starting Land Use**: The input to the model includes the square kilometers of land used for each of 5 crops in 2014. This game will assume that each country's land is used optimally for the set of crops it produces in 2014. On a granularity of 100 square kilometers, the development team must determine an optimal assignment of land within the country to each crop such that the largest number of one kilometer cells contain a crop that are within its ideal growing zone. This distribution will not, in general, be unique.

This distribution is a *Derived Internal State*. It will be used in many calculations, but will not be made visible to the user. As the milestone 2 model has no data about the location of cities, protected areas, and soil, the distribution produced may be very unrealistic.

4) **Crop Yield by Country:** The input to the model includes the 2014 square kilometers of land used for each of the 5 crops being modeled. It also includes the 2014 production in metric tons of each crop. In this model, a country's yield for a particular crop is defined as:

$$yield\,(country, crop) =$$

$$= \frac{production}{idealLand + (acceptableLand \times acceptableRate) + (poorLand * poorRate)}$$

where,

production is the metric tons of the crop produced by the country in 2014.

idealLand is the square kilometers of the crop assigned to ideal land.

acceptableLand is the square kilometers of the crop assigned to acceptable land.

poorLand is the square kilometers of the crop assigned top poor land.

See "Starting Land Use" for how ideal, acceptable and poor land is assigned. Also see "Crop Zone Data" of the "Climate, Crop, and Land Data" section for the three rates.

In milestone 2, it is assumed that a country's yield for each crop in 2014 is its base yield and remains constant for that country throughout the game.

5) **2014 Annual Country Consumption:** For 2014, calculate the amount of food consumed by each country.

$$AnnualCountryConsumption(country, crop, \mathbf{2014}) = produced + import - export$$

Where *produced*, *import* and *export* are the metric tons of the particular crop produced, imported and exported by the given country in the given year.

6) **2014 Per Capita Consumption:** At the start of the model only, calculate the 2014 per capita consumption for each country for each of the 5 crops:

$$2014PerCapitaConsumption(country, crop) = \frac{AnnualCountryConsumption(country, crop, \mathbf{2014}))}{(population(country, \mathbf{2014}) - 0.5 \times undernourished(country, \mathbf{2014}))}$$

In milestone II, it is assumed that each country's 2014 per capita consumption of each crop remains constant for that country throughout all years of the simulation.

Model Requirements at Each Year:

1) **Base Annual Temperature and Precipitation:** Use the researched temperature and precipitation data values and linear interpolation to independently determine each of the following on every one hundred square kilometer area on Earth that is included in the model. Note: oceans, seas, Antarctica and other non-arable land is not included in the model.
   a) Annual maximum temperature.
   b) Annual minimum temperature.
   c) Average annual daytime temperature.
   d) Average annual nighttime temperature.
   e) Total annual precipitation.

   These values need to be stored as a table so that noise may be added and then the final result can be made available as an overlay. Most likely, the development team will want to use a table of floats (4 bytes per value) that that includes latitude and longitude along with the other data. Search times can be vastly improved if the records are grouped by proximity.

2) **Add Noise to Temperature and Precipitation:** Independently, for each temperature and precipitation array, visit a randomly selected 10 percent of the land cells in a random order. For each cell visited, calculate *delta* from the appropriate equation below.

$$delta = 0.1 \times (Tmax_i - Tmin_i) \times randomizationPercentage \times (r1 - r2)$$

$$delta = 0.1 \times (Tday_i - Tnight_i) \times randomizationPercentage \times (r1 - r2)$$

$$delta = 0.1 \times rain_i \times randomizationPercentage \times (r1 - r2)$$

   Where, $r1$ and $r2$ are uniformly distributed random numbers from 0.0 through 1.0.

   The first equation is use to when modifying the annual maximum temperature and again when modifying the Annual minimum temperature. The second equation is used when modifying the average daytime and average nighttime temperatures. The last equation is used when modifying the annual precipitation.

   Modify all of the appropriate array's land cell values within a 100 kilometer radius of the visited cell by the equation:

$$array_k = array_k + \frac{delta}{\ln(e + d(i,k) \times r3)}$$

   Where,

   $d(i,k)$ is the distance in kilometers form the visited cell i to the neighbor cell k.

   $r3$ is a uniformly distributed random number from 0.0 through 2.0. Note: choose a new $r3$ for each neighbor cell k.

   Note: Since the locations are on a sphere, the actual distance between the two cells is a

path along a great circle. However, that would be a silly amount of accuracy for adding random noise.

3) **Sea Level Rise this Year**: At the start of the model, the *Base Sea Level Rise per Year* was calculated (see above). The amount that sea level raises in each year is:

$$seaLevelRise(year) = BaseSeaLevelRisePerYear \times (1 + randomizationPercentage \times r$$

Where,

r is a uniformly distributed random number from 0.5 through 2.0.

Hint: The **`nextFloat()`** method of **`java.util.Random`** returns a uniformly distributed random number from 0.0 through 1.0. To change this range to 0.5 through 2.0, use:

```
myRandom.nextFloat()*1.5f + 0.5f;
```

4) **Arable Land Update**: Each year of the model, any land that is below sea level is no longer arable.

5) **Annual Country Birth Rate:** Input to the model includes the number of births per 1000 people for each country in 2014. In milestone II, assume this remains constant for each country throughout all years modeled in the simulation.

6) **Annual Country Migration Rate:** Input to this model includes the net migrations per 1000 people for each country in 2014. In milestone II, assume this remains constant for each country throughout all years modeled in the simulation.

7) **Annual Country Death Rate:** Input to this model includes the number of deaths per 1000 people for each country in 2014. For each country, assumes that:

a) When the number of undernourished people in previous year is ***less than or equal*** to the number of undernourished people in 2014, then the current year's death rate is the same as the 2014 death rate.

b) When the number of undernourished people in the previous year is ***greater than*** the number of undernourished people in 2014, then the number of deaths per 1000 persons is:

$$deathRate(country, year) = \frac{deathRate(country, \mathbf{2014}) + \mathbf{0.2} \times \left(hungry(country, year - \mathbf{1}) - hungry(country, \mathbf{2014})\right)}{population(country, year - 1)/\mathbf{1000}}$$

8) **Annual Country Population:** Input to the model includes the population of each country in 2014. For all future years, population is calculated using the equation:

$$population(country, year) = population(country, year - \mathbf{1}) + \left(birthRate(country, year) + migration(country, year) + deathRate(country, year)\right) \times \left(population(country, year - \mathbf{1}) / \mathbf{1000}\right)$$

9) **Desired Annual Country Consumption:** For all years after 2014, the desired annual consumption of a particular crop in a particular country is:

$$desiredAnnualCountryConsumption(country, crop, year) = 2014PerCapitaConsumption(country, crop) \times population(country, year)$$

10) Annual Crop Production:

11) **Annual Undernourished People:** Input to the model includes the number of undernourished people for each country in 2014. For all future years, this is calculated using the equation:

12) **Annual Unhappy People:** The number of unhappy people in a country is equal to whichever is smaller, the total population of the country or:

$$5 \times undernourished + 2 \times \Delta undernourished + 10 \times deaths + 5 \times \Delta deaths$$

Where,

undernourished is the country's undernourished population in the current year.

Δ*undernourished* is the change in the country's undernourished population from the previous year.

deaths is the number of deaths that occurred in the country during the current year.

Δ*deaths* is the change in the number of deaths in the country from the previous year.

Java Project Requirements:

1) Required Classes: Each development team must extend the given **`AbstractCountry`** and **`AbstractScenario`** classes.

2) **Package Structure:** No changes may be made to the given classes. This requirement dictates at least some of the package structure. For example, **`worldfoodgame.common.AbstractCountry.java`** will not compile if it is not in the directory: **`src/worldfoodgame/common/`**

   Additionally, all of your code must be in **`src/worldfoodgame/`**<*mypackagename*>

   Where *mypackagename* can be as many different names as you want for however you want

to break up your project, but must be only one level deep. That is, you may not have any directories inside a *mypackagename*.

Exception: if you have what you think is a good design reason for organizing classes into a deeper package structure, then speak with me about it. There are times when that is the best solution. What I specifically do not want are folders that contain a single folder that contains a single folder that contains a single folder, ... to a few scattered classes. Package structures were not designed to be a hunt for the minotaur.

3) XML parsing will be done using Oracle's SAX library.

4) Beyond the SAX reader and LWJGL, some groups may want to use other third party packages. Before using any such package, each group must receive explicit permission from the CS-351 course instructor (Joel). As the software developed as part of this project will be will be made open-source, all third party packages must be freely distributable (although third party packages need not be open-source).

5) Any third party images, sound or other resources must be 1) clearly sited and 2) licensed for free redistribution with or without modifications to those resources.

6) All student developed source code will meet the CS-351 coding standards (see course website for details).

7) All classes and all ***public*** methods must include JavaDoc. You can use an auto generator for creating place holders, but it will be obvious if you do not edit the auto generated text. Class JavaDoc must explain how the class is designed to be used. Method JavaDocs must list and explain input, return values and any side effects. You may also sometimes choose to include explanations of how the method does what it does if you feel it will help your classmates who may be editing your code in milestone 2 or 3.

8) During development, all groups will ***correctly*** use a non-publicly viewable Git repository (such as GitHub) for version control. Correct use of a Git repository for a project this size means:

   a) Having ONLY ONE BRANCH.

   b) Both team members need to be making frequent commits to that one branch.

   c) Only working code should be committed to the repository.

   d) Each commit must include a one or two line comment describing the changes.

   e) Commits must be frequent enough so that a line of text is sufficient to describe the changes made.

   In addition to the two team members, read access to the repository must be given to the CS-351 course instructor (GitHub account castellanos70) and lab instructor (GitHub account TBA).

9) The code ***must*** be very modular.

10) Every group's base directory (which must be included in the .zip archive) must be: **`worldfoodgame2_`***member1_member2_member3*.