

> Summary and Outcomes

Students will learn to use Java to solve logical problems. To do this, they will implement a classic “turn”-based strategy game. In this game, players will act as feudal lords making decisions about buying and selling land and planting specific acreage. To succeed, they must successfully balance population demands with available crops. They must achieve this task while trying to stave off random occurrences such as the plague and the ever-present rat problem. Students will have to develop algorithms to model all of the parameters involved in this game. To do this, they will write a `Hammurabi` class with all of the required attributes and methods provided. Driving this class will be an applet created with input to reflect the combination of decisions made by the feudal lord. Students will also need to develop a test suite to adequately test their code.

> Student Assessment

How well do students perform each task:

- Design and implement classes with all required attributes
- Implement all necessary methods included in the classes
- Develop a test suite to fully explore the many possible scenarios

> Prerequisite Knowledge and Skills

Java:

- Basic control structures
- Basic graphics (methods of the `java.awt.Graphics` class)
- Applets

Logic:

- Ability to code game conditions
- Ability to understand how the game proceeds
- Ability to develop and implement a robust test plan

> Subject Area(s): This activity targets the following:

Pre-Java	Java Programming
<input type="checkbox"/> Hardware Basics	<input checked="" type="checkbox"/> Applet Programming
<input checked="" type="checkbox"/> Software Basics	<input type="checkbox"/> Subroutine Programming
<input type="checkbox"/> Networks and Servers	<input type="checkbox"/> Application Programming
<input type="checkbox"/> HTML	
<input type="checkbox"/> Action Scripting	
<input type="checkbox"/> Java Scripting	

> Curriculum-Framing Questions

Fundamental Questions

- What is the initial state of the game and how are those elements stored?
- What methods (choices/possible scenarios) appear throughout and what are the consequences of each?

Activity Questions

- Do students understand how the game works?
- Have they played the game on Internet sites to get a feel for how it works?

Sample Content Questions

- What must be included in the `Hammurabi` class?
- How should the interface (applet) be designed?

> Targeted Content Standards, Benchmarks, or State Frameworks

California State Standards:

Mathematics:

- **Algebra I: Standard 5.0** – Students solve multi-step problems, including word problems, involving linear

International Society of Technology Education (ISTE):

Information Literacy Standards:

- Standard 1 – The student who is

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by the East Side Union High School 2004.

equations and linear inequalities in one variable and provide justification for each step.

- **Algebra I: Standard 15.0** – Students apply algebraic techniques to solve rate problems, work problems, and percent-mixture problems.
- **Probability and Statistics: Standard 1.0** – Students know the definition of the notion of independent events and can use the rules for addition, multiplication, and complementation to solve for probabilities of particular events in finite sample spaces.

information literate accesses information efficiently and effectively.

- Standard 2 – The student who is information literate evaluates information critically and competently.

Technology Foundation Standards for Students:

- **Standard 3** – Students use technology tools to enhance learning, increase productivity, and promote creativity. Students use productivity tools to collaborate in constructing technology-enhanced models, prepare publications, and produce other creative works.

Science Standards for Students Grades 9–12:

- Content Standard F: Science in Personal and Social Perspectives
 - **F1** – Personal and community health
 - **F2** – Population growth
 - **F3** – Natural resources
 - **F4** – Environmental quality
 - **F5** – Natural and human-

induced hazards

> Materials and Resources Required for Activity

Equipment	<ul style="list-style-type: none">• Computer for each student with Java SDK installed
Consumable Supplies	<ul style="list-style-type: none">• Diskettes or CD-ROMs
Textbooks/Lesson Guides	<ul style="list-style-type: none">• There are many applicable texts, as this unit requires only basic Java. Some reference guide possibilities are as follows:<ul style="list-style-type: none">○ <i>Objects First With Java: A Practical Introduction to BlueJ</i> by David J. Barnes and Michael Kölling. Prentice Hall, 2003.○ <i>Java Methods</i> by Maria Litvin and Gary Litvin. Skylight Publishing, 2001.○ <i>Java Software Solutions: Foundations of Program Design (Fourth Edition)</i> by John Lewis and William Loftus. Addison-Wesley, 2004.
Technology	<ul style="list-style-type: none">• Windows-based machines running Windows 98 SE or higher
Internet	<ul style="list-style-type: none">• http://www.atariarchives.org/basicgames/showpage.php?page=78

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by the East Side Union High School 2004.

> Activity Outline

Overview: For this activity, students will undertake several tasks. First, they will code the `Hammurabi` class with provided data settings and all of the required methods to initialize the `Hammurabi` game. In this game, players are feudal lords making agricultural decisions—buying and selling acreage, determining how much area to plant crops on, etc. The main objective is to increase the population under their control, which is achieved by both providing enough food for the growing population and minimizing the activity of rats that might destroy the crops. Unless fatal events occur, the game can continue for 10 game years. Students should include error checking in the code to ensure, for example, that players do not try to plant more acreage than they have. The code must also calculate the results of any given “turn,” keeping track of how much land was bought and sold, how much was planted, the crop yield that resulted, and how much was eaten by rats. This report should be generated at the end of each turn.

The second task for students is to develop a simple applet in which players can enter their choices: acreage to buy, sell, and plant as well as the number of bushels used to feed the population. Once the `Hammurabi` class and the applet are compiled and running, the third task will be to develop a test suite. To do this, students will need to understand how the game works and be able to calculate the results of several sets of selections. The fourth and final task will be to completely test the application to make sure it fulfills all of the required specifications.

At the start of the game, set the initial data attributes as follows:

- Year = 1
- Number Starved This Year = 0

- Number Moving to the Territory This Year = 5
- Population = 95
- Acres Owned = 1000
- Yield = 3
- Number of Rats = 200
- Number of Bushels = 2400 (bushels available for trade to buy land)
- Trade = 24 (Multiplier for buy or sell transactions—number of bushels per acre)
- Total Starved to Date = 0
- Percent Starved = 0
- Plague = 0

When taking a turn, the player will indicate the following:

- Acres to buy
- Acres to sell
- Acres to plant
- Bushels to be used to feed the population

After defining the attributes, develop corresponding methods and add error checking. The code should check the input to make sure that there are enough bushels available to trade in exchange for the acreage to be bought. The amount of acres planted should not exceed the number owned, and the amount allocated to feed the population should not exceed the amount remaining after either purchasing or selling land. You need to use one or more methods in the code to calculate statistics at the end of each turn.

At the end of a cycle, the following should be true:

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by the East Side Union High School 2004.

- The acreage is equal to the beginning acreage plus any land bought minus land sold
- The number of bushels available is decreased by trade times acres bought
- The number of bushels available is then increased by trade times acres sold
- The number of bushels is further decreased by half the acreage planted
- The number of bushels is further decreased by the amount used to feed the population
- Yield for the year is a random number between 1 and 5
- The number of bushels is incremented by yield times number of bushels planted
- The rat population is the product of a random number between 0 and 1 multiplied by the number of bushels
- The number of bushels is decremented by the number of rats (each rat eats a bushel of grain each year)
- The population increases by a random amount calculated as follows:

R = random number from 1 to 5

Increase = (R * Acres + Number of Bushels)/Population/100 + 1

- The plague factor is a random number between -3 and 17

Formula: 10 * (2 * Math.random() - .3)

- To calculate the number who have starved, create a temporary variable:
FeedEnough

FeedEnough = Number of Bushels Allocated for Food/20

Number who Starved this Year = Population – FeedEnough

- If Number Starved exceeds 45 percent of the population, generate a message about impeachment and send a set of messages about the impeachment
- If the number who starved is less than 45 percent of the population, then recalculate the population as follows:

**Population = Old Population value – Number who Starved +
Number who Moved to Town**

- Total Starved is incremented by the Number Starved this year
- Percent Starved is calculated as follows:

**(Year–1) * percentstarved + numstarved *
100/population)/numyear**

Finally, the code must generate an annual report indicating the following end-of-year results:

- Year
- Number who Starved
- Population, recalculated as Old Population minus Number Starved plus New Arrivals
- If plague is negative, then generate a message about a horrible plague and reduce the population by half
- Population

- Acres owned
- Yield
- Amount eaten by rats (equal to number of rats)
- Current trading price: an amount equal to 17 plus a random number between 0 and 10

Students should calculate the expected results for various combinations of inputs. To do this, students need to understand the logic of the game and be able to determine the effects of certain sets of selections. They should develop a suite of test sets for which they have already calculated the results. After they put the suite in place, they should use it to thoroughly test the application.

> Pacing/Timeline

- Algorithm development: 5–10 hours
- Class development: 3–6 hours
- Test suite development: 2–4 hours
- Complete testing: 2–4 hours

> Teacher Reflection (For example, what worked well in this activity? What would you change if you were to teach it again?)

- Did students understand how the game worked and how to play it?
- Did students develop more than a surface curiosity for the activities?
- What teaching strategies could be used to enhance the activities and classroom discussions?
- Are there cross-learning opportunities to work collaboratively with other educators in science, math, economics, or social studies? If yes, then how?
- What would be the next step? Should students develop a strategy game of their own or a Java version of an existing game?

> Addendum 1: Sample Code Hammurabi Class

```

public class Hammurabi
{
    private String OutputStr;
    private int numyear;
    private int numstarved;
    private int numentered;
    private int population;
    private int acresowned;
    private int yield;
    private int rats;
    private int numbushels;
    private int trade;
    private int totalstarved;
    private int percentstarved;
    private float plague;

    public Hammurabi() // default constructor
    {
        OutputStr=" ";
        numyear=1;
        numstarved=0;
        numentered=5;
        population=95;
        acresowned=1000;
        yield=3;
        rats=200;
        numbushels=2800;
        trade=24;
        totalstarved=0;
        percentstarved=0;
        plague=1.0f;
    }
    public String processData(int buy, int sell, int feed, int plant)
    {
        int feedenough;
        int ranFive = ((int)(Math.random()*5)+1);
        double starved = ((numyear-
1)*percentstarved+numstarved*100/population);
        if (numyear>11) startover();
        if ((buy<0)|| (sell<0)|| (feed<0)|| (plant<0))
        {
            cannotdo();
        }
        if (buy*trade>numbushels)
            notenoughbushels("to buy");
        else if (sell > acresowned)

```

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by the East Side Union High School 2004.

```

        notenoughacres("to sell");
    else if (feed > (numbushels+(sell-buy)*trade))
        notenoughbushels("to feed");
    else if (plant > (acresowned-sell+buy))
        notenoughacres("to plant");
    else if (plant*.5>numbushels)
        notenoughbushels("to plant");
    else if (plant/10>population)
        notenoughpeople();
    else
    {
        numbushels -= buy*trade;
        numbushels += sell*trade;
        numbushels -=(plant/2);
        numbushels -= feed;
        acresowned = acresowned + buy - sell;
        yield = (int)(Math.random()*5)+1;
        numbushels += yield*plant;
        rats = (int)(numbushels*Math.random());
        numbushels -= rats;
        numentered = (int)((ranFive *
acresowned+numbushels)/population/100+1);
        feedenough = feed/20;
        plague = (int)(10*(2*Math.random()-.3));
        numstarved = population - feedenough;
        if (numstarved > .45*population)
        {
            OutputStr += "You starved "+numstarved+" people in
one year!!!!";
            impeachedresults();
        }
        else
        {
            percentstarved = (int)starved/numyear;
            population -= numstarved;
            population += numentered;
            totalstarved += numstarved;
            numyear++;
            annualresults();
            if (numyear==11)
                finalresults();
        }
    }
    return (OutputStr);
}

public String annualresults( )
{
    OutputStr += "\n\nHammurabi: I beg to report to you,\n";

```

```

        OutputStr += "In year " + numyear + ", " + numstarved;
        OutputStr += " people starved, " + numentered + " came to the
city,\n";
        population += numentered;
        if (plague <= 0.0f)
        {
            OutputStr += "A horrible plague struck! Half the people
died.\n";
            population /=2;
        }
        OutputStr += "The population of the city is now "+ population
+ ".\n";
        OutputStr += "The city now own " + acresowned + " acres.\n";
        OutputStr += "The harvest is "+ yield + " bushels per
acre.\n";
        OutputStr += "Rats ate " + rats + " bushels.\n";
        OutputStr += "The city now has " + numbushels + " bushels in
store.\n\n";
        trade = 17 + (int)(10*Math.random());
        OutputStr += "Land is trading at " + trade + " bushels per
acre.\n\n";
        OutputStr += "\nWhat do you wish to do for the coming
year?\n";
        return (OutputStr);
    }

    public String finalresults( )
    {
        OutputStr += "\nIn your 10-term of office,
"+percentstarved+"% of the\n";
        OutputStr += "population starved per year on the average, ";
        OutputStr += " i.e. a total of\n";
        OutputStr += totalstarved + " people died!!\n";
        int l=acresowned/population;
        if ((percentstarved>33) || (l<7))
        {
            impeachedresults();
        }
        else
        {
            if ((percentstarved>10) || (l < 9))
            {
                heavyhandedresults();
            }
            else
            {
                if ((percentstarved>3) || (l < 10))
                {
                    okresults();
                }
            }
        }
    }

```

```

        }
        else
        {
            greatresults();
        }
    }
}
return (OutputStr);
}

public String impeachedresults( )
{
    OutputStr += "Due to this extreme mismanagement, you have not
only\n";
    OutputStr += "been impeached and thrown out of office, but
you have\n";
    OutputStr += "also been declared National Fink!!!!\n";
    return (OutputStr);
and Ivan IV.\n";
    OutputStr += "The people (remaining) find you an unpleasant
ruler, and,\n";
    OutputStr += "frankly, hate your guts!!\n";
    return (OutputStr);
}
public String okresults( )
{
    OutputStr += "Your performance could have been somewhat
better, but\n";
    OutputStr += "really wasn't too bad at all. ";
    OutputStr += (int)(population*.8*Math.random()+ " people
would\n";
    OutputStr += "dearly like to see you assassinated, but we all
have our\n";
    OutputStr += "trivial problems.\n";
    return (OutputStr);
}
public String greatresults( )
{
    OutputStr += "A fantastic performance!!! Charlemagne,
Disraeli, and\n";
    OutputStr += "Jefferson combined could not have done
better!\n";
    return (OutputStr);
}
public String cannotdo( )
{
    OutputStr += "Hammuarabi: I cannot do what you wish.\n";
    OutputStr += "Get yourself another Steward!!!!!\n";
    return (OutputStr);
}

```

```

    }
    public String notenoughbushels(String str)
    {
        OutputStr += "Hammurabi: Think again. You have only\n";
        OutputStr += numbushels + " bushels of grain "+str+". Now
then\n";
        return (OutputStr);
    }
    public String notenoughacres(String str)
    {
        OutputStr += "Hammurabi: Think again. You own only\n";
        OutputStr += acresowned + " acres "+str+". Now then,\n";
        return (OutputStr);
    }
    public String notenoughpeople()
    {
        OutputStr += "Hammurabi: Think again. You have only\n";
        OutputStr += population + " people to tend the fields!! Now
then,\n";
        return (OutputStr);
    }
    public String clearScreen()
    {
        OutputStr = "";
        return (OutputStr);
    }
    public void startover()
    {
        OutputStr="";
        numyear=1;
        numstarved=0;
        numentered=5;
        population=95;
        acresowned=1000;
        yield=3;
        rats=200;
        numbushels=2800;
        trade=24;
        totalstarved=0;
        percentstarved=0;
        plague=1.0f;
    }
}

```

> Addendum 2: Sample Code Hammurabi Applet

```

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HammurabiApplet extends Applet implements
ActionListener
{
    private TextField inBuy;
    private TextField inSell;
    private TextField inFeed;
    private TextField inPlant;
    private TextArea resultArea;
    private Label prompt1;
    private Label prompt2;
    private Label prompt3;
    private Label prompt4;
    private Button buttonProcess;
    private Button buttonClear;
    Hammurabi ham;
    String OutputStr;
    private JPanel input;
    private JPanel buttons;

    public void init()
    {
        setLayout(new BorderLayout());
        input = new JPanel();
        input.setLayout(new GridLayout(4,2));
        buttons = new JPanel();
        buttons.setLayout(new GridLayout(2,4));
        inBuy = new TextField(10);
        inSell = new TextField(10);
        inFeed = new TextField(10);
        inPlant = new TextField(10);
        resultArea = new TextArea (20,60);
        prompt1 = new Label("How many acres do you wish to Buy?");
        prompt2 = new Label("How many acres do you wish to Sell?");
        prompt3 = new Label("How many bushels do you wish to feed
your people?");
        prompt4 = new Label("How many acres do you wish to plant
with seed?");
        buttonProcess = new Button("Buy, Sell, Plant, Feed");
    }
}

```

Sun, Sun Microsystems, the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. This Java activity was developed for The Sun Foundation by the East Side Union High School 2004.

```

buttonClear = new Button("Clear Screen");

input.add(prompt1);
input.add(inBuy);
input.add(prompt2);
input.add(inSell);
input.add(prompt3);
input.add(inFeed);
input.add(prompt4);
input.add(inPlant);

buttonProcess.addActionListener(this); // Listeners
buttonClear.addActionListener(this);

buttons.add(buttonProcess);
buttons.add(buttonClear);

ham = new Hammurabi();
OutputStr = "Try your hand at governing Ancient Sumeria\n";
OutputStr += "for a Ten-year Term of office.\n\n";
OutputStr += ham.annualresults();
resultArea.setFont(new Font("Courier", Font.PLAIN, 12));
resultArea.setText(OutputStr);

add(input, BorderLayout.NORTH);
add(buttons, BorderLayout.SOUTH);
add(resultArea);
}
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == buttonProcess)
    {
        int Buy = Integer.parseInt(inBuy.getText());
        int Sell = Integer.parseInt(inSell.getText());
        int Feed = Integer.parseInt(inFeed.getText());
        int Plant = Integer.parseInt(inPlant.getText());
        OutputStr = ham.processData(Buy, Sell, Feed, Plant);
        resultArea.setText(OutputStr);
    }
    else if (e.getSource() == buttonClear)
    {
        ham.clearScreen();
        resultArea.setText("");
    }
}
}

```