

> Activity Title: The Camel Game

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 the objective is for players to cross 200 miles of desert in seven days. They must achieve this goal while keeping themselves and their camels sufficiently fed and hydrated. During the task, they will also have to outrun bandits while avoiding overly fatiguing the camels. The challenge here will not be simply to write code but to make the code implement an algorithm. As a result, students will gain some experience in algorithm development. Given the parameters of the game, they will need to make choices between competing needs, such as travel over rest or speed over fatigue, in attempting to successfully cross the desert.

> 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
- Interfaces

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"/> Full Scale Programming
<input type="checkbox"/> HTML	
<input type="checkbox"/> Action Scripting	
<input type="checkbox"/> Java Scripting	

> Curriculum-Framing Questions

Essential Questions	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Do students understand how the game works? • Have students played the game on Internet sites to get a feel for how it works?
Sample Content Questions	<ul style="list-style-type: none"> • What must be included in the <code>Camel</code> 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 equations and linear inequalities in one variable and provide justification for each step.

International Society of Technology Education (ISTE):

Information Literacy Standards:

- Standard 1 – The student who is information literate accesses information efficiently and effectively.
- Standard 2 – The student who is information literate evaluates information critically and competently.

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.

- 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.
- Probability and Statistics: Standard 2.0 – Students know the definition of conditional probability and use it to solve for probabilities in finite sample spaces.

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

- Computer for each student with Java SDK installed

Consumable Supplies

- Diskettes and CD-ROMS

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.

**Textbooks/Lesson
Guides**

- There are many possible texts, as this unit requires only basic Java. Some possibilities are included as follows
 - *Objects First With Java: A Practical Introduction to BlueJ* by David J. Barnes and Michael Kölling. Prentice Hall, 2003.
 - *Java Methods* by Maria Litvin and Gary Litvin. Skylight Publishing, 2001.
 - *Java Software Solutions: Foundations of Program Design (Fourth Edition)* by John Lewis and William Loftus. Addison-Wesley, 2004.

Technology

- Windows-based machines running Windows 98 SE or higher

Internet Resources

- <http://www.atariarchives.org/morebasicgames/showpage.php?page=24>

> Activity Outline

Overview: For this activity, students will undertake several tasks. First, they will code the `Camel` class with provided data settings and all of the required methods to initialize the Camel Game. Second, they will code an applet in which players can input their selections and experiment to immediately see what they have produced. Third, they will develop a test suite with predefined results to adequately test how the code has implemented the assumptions of the game. Finally, they will fully test the program to see that it fulfills all of the required specifications.

The purpose of the game is to cross the desert before any of several possible fatal events occurs. These include: running out of water, having a camel die, and being captured by outlaw treasure seekers (OTS). At the start of the game, the attributes should be set as follows:

- Distance Traveled = 0
- Available Commands = 4
- Available Drinks = 6
- Camel Days Used = 0
- Turns Made = 0
- OTS_distance = 0

The object of the game is to travel 200 miles across the desert, keeping the camel alive and the water supply intact. At the outset, there are four commands available. The camel has seven “strength-days.” This is a measurement tool to track the stamina and energy level of the camel. Traveling at maximum speed will naturally deplete the camel’s strength at a faster rate than traveling at moderate speed. The camel’s strength-days can be replenished by stopping.

Each time a command is selected, the logic associated with that command is executed, and a subsequent method then reduces the available commands. This

reduction in turn begins the process of calling forth an “adventure,” or random event, which will be discussed in more detail later.

The command process “stop” will completely replenish the camel’s strength, returning him to the level of having seven days of travel available.

Taking a drink of water will reduce the number of drinks remaining by one thus resetting the available commands to four.

Traveling at a moderate pace will increase the days used by the camel. If the camel uses up seven days, it will die, a message will appear and the game will end. On any given turn, if the camel has not died and no other bad fortune has befallen the traveler, he/she will gain a random number of miles – something between 1 and 10.

Traveling at full speed will reduce by three the number of days left for the camel, but it will approximately double the speed made on this leg of the trip. The life expectancy of the camel drops drastically but the player will gain between 1 and 19 miles.

The number of available days can be reset to 7 by opting to stop. Of course stopping uses up a turn and increases the likelihood of being set upon by the OTS (Outlaw Treasure Seekers.)

The player may also request to view his/her status. This option will display the number of camel-days left, available drinks of water, and remaining commands - but doing so will count as a command selected.

Seeking help is a desperation selection. When this option is selected, there is a 90 percent chance that the player will perish and only a 10 percent chance that help will arrive. Should help arrive, the drink level will be reset to three and the number of commands to four. This will also count as a command and the usual next step will ensue.

After issuing a command, one of several adventures (random events) may occur:

- There is a 20 percent probability of finding an oasis. This replenishes the water supply and resets the available command level to four.

- There is a 30 percent chance of encountering a sandstorm. In a sandstorm, a player has a 50 percent chance of advancing some random number of miles between 0 and 10. There is also a 50 percent chance of retreating that same number of miles.
- There is a 25 percent chance that the OTS have increased their distance by one. The OTS are assumed to be moving in the same direction as the player, so any time their distance increases by more than the player's distance, then they are getting closer. Should the OTS overtake the player, the game will end with the player's (and perhaps the camel's) demise.
- There is a 25 percent chance that none of the above happens.

</applet>

After each command is implemented, the code will check to see whether the player has traveled 200 or more miles. If so, the player should get a congratulatory message indicating that he/she has won the game. Otherwise, the number of commands is decremented and the number of turns taken incremented. What happens next depends on how many commands are left.

- With only one command left, the player will get a warning about needing water.
- With less than zero commands left, the player will die. The code will process an out-of-water method at this point, followed by a random message about how the player perished generated in the `gonegone()` method.
- If the player has two or more commands left (and has taken more than four turns), the OTS will advance some random number of miles between 3 and 12. If at this point the OTS have traveled more miles than the player has, the player will be killed.

Your `Camel` class must model the behavior detailed above.

Once the `Camel` class has been coded, you can then implement an applet.

The applet should include a text area. Initially, this area will display the workings of the game—the objective and obstacles. There should also be a number of buttons allowing the player to do the following:

- Drink water
- Move at a moderate pace
- Move at full speed
- Stop
- Check status
- Hope for help
- Clear the text area
- Start a new game

> **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 game 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 Camel Class

```
/* Code for Camel Game
*/

public class Camel
{
    String OutputStr;
    private int distance_traveled;
    private int commands_left;
    private int drinks_left;
    private int camel_strength;
    private int turns;
    private int ots_distance;
    private int i;

    public Camel()
    {
        distance_traveled=0;
        commands_left=4;
        drinks_left=6;
        camel_strength=0;
        turns=0;
        ots_distance=0;
        OutputStr="";
    }
    public String processDrink( )
    {
        drinks_left--;
        if (drinks_left < 0)
        {
            out_of_water();
        }
        else
        {
            OutputStr += "Better watch for an oasis.\n";
            commands_left = 4;
            next_step();
        }
        return (OutputStr);
    }
    public String processModerate( )
    {
        camel_strength++;
        if (camel_strength>7)
        {
```

```

        camel_died();
    }
    else
    {
        any_adventures();
        distance_traveled += (int)(Math.random()*10)+1;
        OutputStr += "Your camel likes this pace.\n";
        next_step();
    }
    return (OutputStr);
}
public String processFull( )
{
    camel_strength += 3;;
    if (camel_strength>7)
    {
        camel_died();
    }
    else
    {
        any_adventures();
        distance_traveled += 2*((int)(Math.random()*10))+1;
        OutputStr += "Your camel is burning across the desert ";
        OutputStr += "sands.\n";
        next_step();
    }
    return (OutputStr);
}
public String processStop( )
{
    OutputStr += "Your camel thanks you!\n";
    camel_strength=0;
    next_step();
    return (OutputStr);
}
public String processStatus( )
{
    OutputStr += "Your camel has "+ (7-camel_strength);
    OutputStr += " good days left,\n";
    OutputStr += "You have "+ drinks_left;
    OutputStr += " drinks left in your canteen,\n";
    OutputStr += "You can go "+ commands_left;
    OutputStr += " commands without drinking,\n";
    next_step();
    return (OutputStr);
}
public String processHope( )
{
    i = (int)(10*Math.random());

```

```

        if (i != 1)
        {
            gonegone();
        }
        else
        {
            OutputStr += "Help has found you ";
            OutputStr += "in a state of unconsciousness,\n";
            drinks_left = 3;
            commands_left = 4;
            next_step();
        }
        return (OutputStr);
    }
    public void camel_died()
    {
        OutputStr += "You dirty rascal! You ran ";
        OutputStr += " your poor camel to death!!\n";
        gonegone();
        return;
    }
    public void out_of_water()
    {
        OutputStr += "You ran out of water.....Sorry chum!!!!\n";
        gonegone();
        return;
    }
    public void any_adventures ()
    {
        i = (int)(Math.random()*10)+1;
        if (i <=2)
        {
            found_oasis();
        }
        else
        {
            i = (int)(100 * Math.random()) +1;
            if (i <= 5)
            {
                found_sandstorm();
            }
            else
            {
                i = (int)(100*Math.random()+1;
                if (i <=5)
                {
                    ots_distance++;
                    OutputStr += "Your camel hurt his hump.\n";
                    OutputStr += "Luckily, the OTS were ";
                }
            }
        }
    }

```

```

        OutputStr += "footweary!!\n";
    } // no else needed here
    }
}
return;
}

public void found_oasis()
{
    OutputStr += "You have arrived at an oasis. ";
    OutputStr += " =====Your camel is\n";
    OutputStr += "filling your canteen and eating figs.\n";
    commands_left = 4;
    drinks_left = 6;
    return;
}

public void found_sandstorm()
{
    OutputStr += "You have been caught in a sandstorm ";
    OutputStr += "...Good luck!\n";
    int x5 = (int)(Math.random()*10);
    int x6 = (int)(Math.random()*10);
    if (x6 < 5)
    {
        distance_traveled += x5;
    }
    else
    {
        distance_traveled -= x5;
    }
    OutputStr += "Your new position is "+ distance_traveled;
    OutputStr += " miles so far!\n";
    return;
}

public String processNew( )
{
    OutputStr = "Welcome to Camel.\n";
    OutputStr += "The object is to travel 200 miles across
the\n";
    OutputStr += "Great Gobi Desert to reach the
authorities.\n";
    OutputStr += "A group of outlaw treasure seekers ";
    OutputStr += " (OTS, for short) \n";
    OutputStr += "will be chasing you and will prevent ";
    OutputStr += " you from seeing\n";
    OutputStr += "civilization again. You have ";
    OutputStr += " a 4-command head start.\n\n";
    OutputStr += "You will click on the buttons for ";
    OutputStr += "your commands.\n";
}

```

```

        OutputStr += "\nYou have one quart of water which will ";
        OutputStr += " last you six drinks.\n";
        OutputStr += "You can renew your water supply completely ";
        OutputStr += " at an oasis.\n";
        OutputStr += "You get a half a quart if found by help.\n";
        OutputStr += "If help does not find you after hoping ";
        OutputStr += " for help, you lose.\n";
        OutputStr += "Good luck and good cameling !!\n\n";
        OutputStr += "You are in the middle of the desert at an ";
        OutputStr += "oasis.\n\n";
        start_over();
        return (OutputStr);
    }
    public String clearScreen()
    {
        OutputStr = "";
        return (OutputStr);
    }
    public void next_step()
    {
        if (distance_traveled > 199)
        {
            OutputStr += "\nYou reached civilization!!!\n";
            OutputStr += "A party is being given in your honor.\n";
        }
        else
        {
            commands_left--;
            turns++;
            if (commands_left == 1)
            {
                OutputStr += "----WARNING-----Get a drink.\n";
            }
            else
            {
                if (commands_left < 0)
                {
                    out_of_water();
                }
                else
                {
                    OutputStr += "You have traveled";
                    OutputStr += " "+distance_traveled;
                    OutputStr += " miles altogether.\n";
                    if (turns >4)
                    {
                        i = (int)(10*Math.random()+2.5);
                        ots_distance += i;
                        if (ots_distance > distance_traveled)

```

```

        {
            OutputStr += "The OTS have captured";
            OutputStr += " you.\n";
            OutputStr += "Camel soup is their ";
            OutputStr += " favorite dish.\n\n";
            start_over();
        }
    else
    {
        int behind = distance_traveled-
            ots_distance;
        OutputStr += "The OTS are " + behind;
        OutputStr += " miles behind you.\n";
    }
}
}
}
}
return;
}
}
public void gonegone()
{
    OutputStr += "You died in the desert.\n";
    i = (int)(Math.random()*5)+1;
    switch (i)
    {
        case 1:
            OutputStr += "The National Camel's Union ";
            OutputStr += " is not attending your funeral!!!\n";
            break;
        case 2:
            OutputStr += "Your body was eaten by vultures!!!\n";
            break;
        case 3:
            OutputStr += "The chief of the OTS now uses your ";
            OutputStr += " skull for a change purse!!!\n";
            break;
        case 4:
            OutputStr += "People with little intelligence ";
            OutputStr += " should stay out of the desert!!!\n";
            break;
        case 5:
            OutputStr += "Turkeys should fly, not ride ";
            OutputStr += "camels!!!\n";
            break;
    }
    OutputStr += "\nThat's all!!\n";
    start_over();
    return;
}

```

```

    }
    public void start_over()
    {
        distance_traveled=0;
        commands_left=4;
        drinks_left=6;
        camel_strength=0;
        turns=0;
        ots_distance=0;
        return ;
    }
}

```

> Addendum #2 – Sample Code Camel Applet

```

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

public class CamelApplet extends Applet implements
ActionListener
{
    private TextArea resultArea;
    private Label prompt1;
    private Label prompt2 ;
    private Button buttonDrink;
    private Button buttonModerate;
    private Button buttonFull;
    private Button buttonStop;
    private Button buttonStatus;
    private Button buttonHope;
    private Button buttonNew;
    private Button buttonClear;
    Camel cam;
    String OutputStr;

    private JPanel prompts;
    private JPanel buttons;

    public void init()
    {

        setLayout(new BorderLayout());

```

```

prompts = new JPanel();
prompts.setLayout(new GridLayout(1,2));
buttons = new JPanel();
buttons.setLayout(new GridLayout(2,4));
//c.setLayout;
resultArea = new TextArea (20,60);
prompt1 = new Label("Game of Camel");
prompt2 = new Label("Click on the button of
your
                                choice");
prompts.add(prompt1);
prompts.add(prompt2);
buttonDrink = new Button("Drink from your
canteen");
buttonModerate = new Button("Ahead Moderate
Speed");
buttonFull = new Button("Ahead Full Speed");
buttonStop = new Button("Stop for the Night");
buttonStatus = new Button("Status Check");
buttonHope = new Button("Hope for Help");
buttonNew = new Button("New Game");
buttonClear = new Button("Clear Textarea");
buttons.add(buttonDrink);
buttons.add(buttonModerate);
buttons.add(buttonFull);
buttons.add(buttonStop);
buttons.add(buttonStatus);
buttons.add(buttonHope);
buttons.add(buttonNew);
buttons.add(buttonClear);

add(prompts, BorderLayout.NORTH);
add(buttons, BorderLayout.SOUTH);
add(resultArea);

buttonDrink.addActionListener(this); //
Listeners
buttonModerate.addActionListener(this);
buttonFull.addActionListener(this);
buttonStop.addActionListener(this);
buttonStatus.addActionListener(this);
buttonHope.addActionListener(this);
buttonNew.addActionListener(this);
buttonClear.addActionListener(this);
cam = new Camel();
OutputStr = cam.processNew();
resultArea.setFont(new Font("Courier",
Font.PLAIN,
                                12));

```

```

        resultArea.setText(OutputStr);
    }

    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == buttonDrink)
        {
            OutputStr = cam.processDrink();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonModerate)
        {
            OutputStr = cam.processModerate();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonFull)
        {
            OutputStr = cam.processFull();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonStop)
        {
            OutputStr = cam.processStop();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonStatus)
        {
            OutputStr = cam.processStatus();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonHope)
        {
            OutputStr = cam.processHope();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonNew)
        {
            OutputStr = cam.processNew();
            resultArea.setText(OutputStr);
        }
        else if (e.getSource() == buttonClear)
        {
            cam.clearScreen();
            resultArea.setText("");
        }
    }
}

```