**RobotWorldApp.java:**

```
import java.io.*;

public class RobotWorldApp {

    public static void main( String[] args ) throws IOException {

      RobotWorldGame game = new RobotWorldGame();
      game.show();
    } // end main

} // end class RobotWorldApp
```

**RobotWorldGame.java:**

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class RobotWorldGame extends Frame {

    private final int BOARD_DIM = 12;

    private final int EVIL_NUMBER = 3;

    private final int JUNK_NUMBER = 3;

    private RobotWorldBoard board;

    private MessagePanel messagePanel;

    private Button playAgainButton;

    public RobotWorldGame() {
      addWindowListener(new Closer());

      setSize( 640, 640 );
      setTitle( "Robot World Game " );

      messagePanel = new MessagePanel( "   Click on adjacent square to move.    " );
      messagePanel.setSize( 400, 30 );
      add( "North", messagePanel );

      board = new RobotWorldBoard( messagePanel, BOARD_DIM, BOARD_DIM,
                                   JUNK_NUMBER, EVIL_NUMBER  );
      add( "Center", board );


      playAgainButton = new Button( "New Game"  );
      playAgainButton.addActionListener( new PlayAgainButtonListener() );
      add( "South", playAgainButton );

    } // end RobotWorldGame constructor


    /******************************  Inner-classes  ************************/

    private class PlayAgainButtonListener implements ActionListener {
      public void actionPerformed( ActionEvent e ) {
          messagePanel.setMessage( "    Click on adjacent square to move.      " );

          board.newGame(BOARD_DIM, BOARD_DIM, JUNK_NUMBER, EVIL_NUMBER);
      } // end actionPerformed
    } // end inner-class NumberedButtonListener
```

```
    private class Closer extends WindowAdapter {
      public void windowClosing(WindowEvent e) {
          System.exit(0);
      } // end windowClosing
    } // end inner-class Closer

} // end class RobotWorldGame
```

**RobotWorldBoard.java:**

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.sound.sampled.*;


public class RobotWorldBoard extends Canvas {

    private final int FREE   = 0;
    private final int GOOD   = 1;
    private final int EVIL   = 2;
    private final int PORTAL = 3;
    private final int JUNK   = 4;

    private int numRows;
    private int numCols;
    private int numFunctioningEvilRobots;

    private int clickX;
    private int clickY;

    private int [] [] board;
    private Robot goodRobot;
    private Robot [] evilRobots;
    private Robot [] junkPiles;

    private MessagePanel message;

    private Clip breakingGlassSound;
    private Clip gongSound;

    public RobotWorldBoard(MessagePanel msg, int rows,
                    int columns, int junks, int evils) {
      message = msg;

      //    gongSound = loadSoundEffect("gong.au");
      gongSound = loadSoundEffect("gong.au");
      breakingGlassSound = loadSoundEffect("breaking_glass.au");

      newGame(rows, columns, junks, evils);

      addMouseListener( new MouseKeeper() );

    } // end constructor RobotWorldBoard


    public void newGame ( int rows, int columns, int junks, int evils) {
      int row;
      int column;

      numRows = rows;
      numCols = columns;

      board = new int [numRows][numCols];

      for (int r = 0; r < board.length; r++ ) {
          for (int c = 0; c < board[0].length; c++ ) {
```

```
        board[r][c] = FREE;
      } // end for (int c...
  } // end for (int r...

  //    initializeBoard(junks);

  board[0][numCols-1] = PORTAL;
  board[0][0] = PORTAL;
  board[numRows-1][numCols-1] = PORTAL;
  board[numRows-1][0] = PORTAL;

  do {
      row = (int) (Math.random() * numRows);
      column = (int) (Math.random() * numCols);
  } while (board[row][column] != FREE);

  goodRobot = new Robot( this, "leiaid.GIF", row, column);
  board[row][column] = GOOD;

  evilRobots = new Robot [evils];
  for (int r = 0; r < evils; r++) {

      do {
        row = (int) (Math.random() * numRows);
        column = (int) (Math.random() * numCols);
      } while (board[row][column] != FREE);

      evilRobots[r] = new Robot( this, "vader.gif", row, column);
  } // end for

  numFunctioningEvilRobots = evils;

  junkPiles = new Robot [junks];
  for (int r = 0; r < junks; r++) {

      do {
        row = (int) (Math.random() * numRows);
        column = (int) (Math.random() * numCols);
      } while (board[row][column] != FREE);

      junkPiles[r] = new Robot( this, "vader.gif", row, column);
      board[row][column] = JUNK;

  } // end for
  repaint();
} // end newGame


private Clip loadSoundEffect( String audioFile ) {

  Clip clip = null;
  try {

      AudioInputStream stream = AudioSystem.getAudioInputStream(new File(audioFile));

      // At present, ALAW and ULAW encodings must be converted
      // to PCM_SIGNED before it can be played
      AudioFormat format = stream.getFormat();
      if (format.getEncoding() != AudioFormat.Encoding.PCM_SIGNED) {
        format = new AudioFormat(
                            AudioFormat.Encoding.PCM_SIGNED,
                            format.getSampleRate(),
                            format.getSampleSizeInBits()*2,
                            format.getChannels(),
                            format.getFrameSize()*2,
                            format.getFrameRate(),
                            true);        // big endian
```

```
        stream = AudioSystem.getAudioInputStream(format, stream);
      }

      // Create the clip
      DataLine.Info info = new DataLine.Info(
            Clip.class, stream.getFormat(),
            ((int)stream.getFrameLength()*format.getFrameSize()));
      clip = (Clip) AudioSystem.getLine(info);

      // This method does not return until the audio file is completely loaded
      clip.open(stream);

  }catch (UnsupportedAudioFileException ex) {
      System.out.println("Error: Unsupported");

  }catch (IOException ex) {
      System.out.println("Error: Unsupported");

  } catch (Exception e) {
      System.out.println("Error in Audio stuff");
  }

  return clip;
} // end loadSoundEffects


private void printBoard() {
  System.out.println("");
  for (int r = 0; r < board.length; r++ ) {
      System.out.print("row " + r + ":  ");
      for (int c = 0; c < board[0].length; c++ ) {
        System.out.print(" " + board[r][c]);
      } // end for (int c...
      System.out.println("");
  } // end for (int r...

} // end printBoard


public void paint(Graphics g) {
  drawGrid(g);
  drawJunkPiles(g);
  drawGoodRobot(g);
  drawEvilRobots(g);
} // end paint


private void drawGoodRobot(Graphics g) {
  int width = getWidth();
  int height = getHeight();

  int rowHeight = height / numRows;
  int columnWidth = width / numCols;

  // draw good robot
  goodRobot.drawRobot(g, columnWidth, rowHeight, this);

} // end drawGoodRobots


private void drawEvilRobots(Graphics g) {
  int width = getWidth();
  int height = getHeight();

  int rowHeight = height / numRows;
  int columnWidth = width / numCols;
```

```
  // draw evil robots
  for (int r = 0; r < evilRobots.length; r++ ) {
      evilRobots[r].drawRobot(g, columnWidth, rowHeight, this);
  } // end for (int r...

} // end drawEvilRobots


private void drawJunkPiles(Graphics g) {
  int width = getWidth();
  int height = getHeight();

  int rowHeight = height / numRows;
  int columnWidth = width / numCols;

  g.setColor (Color.red);
  // draw junk piles
  for (int r = 0; r < junkPiles.length; r++ ) {
      //   junkPiles[r].drawRobot(g, columnWidth, rowHeight, this);
      g.fillRect(junkPiles[r].getColumn()*columnWidth+2,
               junkPiles[r].getRow()*rowHeight+2,
               columnWidth-4, rowHeight-4);
  } // end for (int r...

} // end drawJunkPiles


private void drawGrid(Graphics g) {
  int width = getWidth();
  int height = getHeight();

  int rowHeight = height / numRows;
  int columnWidth = width / numCols;

  // draw vertical lines
  for (int c = 0; c <= board[0].length; c++ ) {
      g.drawLine(c * columnWidth, 0, c * columnWidth, height);
  } // end for (int c...

  // draw horizontal lines
  for (int r = 0; r <= board.length; r++ ) {
      g.drawLine(0, r * rowHeight, width, r * rowHeight);
  } // end for (int c...

} // end drawGrid


/***************** inner classes ************************/
private class MouseKeeper extends MouseAdapter {

  public void mousePressed( MouseEvent e ) {

      if (!goodRobot.isWorking()) {
        return;
      } // end if
      int width = getWidth();
      int height = getHeight();

      int rowHeight = height / numRows;
      int columnWidth = width / numCols;

      clickX = e.getX();
      clickY= e.getY();

      int row = clickY/rowHeight;
      int column = clickX/columnWidth;
```

```
        // check if the click was in an adjacent or current square
        if (Math.abs(goodRobot.getRow()-row) <= 1 &&
          Math.abs(goodRobot.getColumn()-column) <= 1) {

          if (board[row][column] == JUNK) {
              goodRobot.move(row, column);
              message.setMessage("Evil Robots Win.  You ran into a junk pile!!!");
              goodRobot.turnToJunk();
                breakingGlassSound.stop();
                breakingGlassSound.flush();
                breakingGlassSound.setFramePosition(0);
                breakingGlassSound.start();

              repaint();
              return;
          } // end if

          message.setMessage("  Click an adjacent square to move!" );
          if ((row == 0 && column == 0) || (row == 0) && (column == numCols-1) ||
              (row == numRows-1 && column == 0) || (row == numRows-1) && (column ==
numCols-1)) {


              do {
                row = (int) (Math.random() * numRows);
                column = (int) (Math.random() * numCols);
              } while (board[row][column] != FREE);

          } // end if

          board[row][column] = FREE;
          goodRobot.move( row, column );
          board[row][column] = GOOD;

          moveEvilRobots(row, column);

        } else {
          message.setMessage("  You must click in an adjacent square to move!" );
        } // end if

        repaint();

      } // end mousePressed

      private void moveEvilRobots( int goodRobotRow, int goodRobotColumn ) {

        // move evil robots toward good robots new position
        for (int r = 0; r < evilRobots.length; r++ ) {
          int nextEvilRobotRow;
          int nextEvilRobotColumn;
          if (evilRobots[r].isWorking()) {
              if (goodRobotColumn < evilRobots[r].getColumn()) {
                nextEvilRobotColumn = evilRobots[r].getColumn()-1;
              } else if (goodRobotColumn > evilRobots[r].getColumn()) {
                nextEvilRobotColumn =  evilRobots[r].getColumn()+1;
              } else {
                nextEvilRobotColumn = goodRobotColumn;
              } // end if

              if (goodRobotRow < evilRobots[r].getRow()) {
                nextEvilRobotRow = evilRobots[r].getRow()-1;
              } else if (goodRobotRow > evilRobots[r].getRow()) {
                nextEvilRobotRow = evilRobots[r].getRow()+1;
              } else {
                nextEvilRobotRow = goodRobotRow;
              } // end if
```

```
                board[evilRobots[r].getRow()][evilRobots[r].getColumn()] = FREE;
                board[nextEvilRobotRow][nextEvilRobotColumn] = EVIL;
                evilRobots[r].move(nextEvilRobotRow,
                               nextEvilRobotColumn);
                if (nextEvilRobotRow == goodRobotRow &&
                  nextEvilRobotColumn == goodRobotColumn) {
                  message.setMessage("                           Evil Robots Win");
                  goodRobot.turnToJunk();
                  breakingGlassSound.stop();
                  breakingGlassSound.flush();
                  breakingGlassSound.setFramePosition(0);
                  breakingGlassSound.start();
                } // end if
              } // end if (evilRobots[r].isWorking()...
            } // end for (int r...

            checkForCollisions();

        } // end moveEvilRobots

        private void checkForCollisions() {
            for (int r = 0; r < evilRobots.length; r++ ) {
              for (int test = r+1; test < evilRobots.length; test++ ) {
                  if (evilRobots[r].getRow() == evilRobots[test].getRow() &&
                    evilRobots[r].getColumn() == evilRobots[test].getColumn()) {
                    if (evilRobots[r].isWorking()) {
                        evilRobots[r].turnToJunk();
                        board[evilRobots[r].getRow()][evilRobots[r].getColumn()] = JUNK;
                        numFunctioningEvilRobots--;
                    } // end if
                    if (evilRobots[test].isWorking()) {
                        evilRobots[test].turnToJunk();
                        board[evilRobots[test].getRow()][evilRobots[test].getColumn()] =
JUNK;
                        numFunctioningEvilRobots--;
                    } // end if
                  } // end if
              } // end for
            } // end for

            for (int r = 0; r < evilRobots.length; r++ ) {
              for (int j = 0; j < evilRobots.length; j++ ) {
                  if (evilRobots[r].getRow() == junkPiles[j].getRow() &&
                    evilRobots[r].getColumn() == junkPiles[j].getColumn()) {
                    if (evilRobots[r].isWorking()) {
                        board[evilRobots[r].getRow()][evilRobots[r].getColumn()] = JUNK;
                        evilRobots[r].turnToJunk();
                        numFunctioningEvilRobots--;
                    } // end if
                  } // end if
              } // end for

            } // end for

            if (numFunctioningEvilRobots == 0) {
              message.setMessage("                        Good Robot Wins!!!");
              gongSound.stop();
              gongSound.flush();
              gongSound.setFramePosition(0);
              gongSound.start();
            } // end if

        } // end checkForCollisions

    } // end inner-class MouseKeeper

} // end class RobotWorldBoard
```

**Robot.java:**

```
import java.awt.*;

public class Robot {

    private RobotWorldBoard board;     // reference to the board
    private Image robotImage;
    private int row;
    private int column;
    private boolean working;

    public Robot(RobotWorldBoard board, String imageFile, int r, int c) {
        robotImage = Toolkit.getDefaultToolkit().getImage( imageFile );
      this.board = board;
      row = r;
      column = c;
      working = true;
    } // end constructor GoodRobot


    public int getRow( ) {
      return row;
    } // end getRow


    public int getColumn() {
      return column;
    } // end getColumn


    public void move (int newRow, int newColumn ) {
      if (working) {
          row = newRow;
          column = newColumn;
      } // end if
    } // end move


    public void drawRobot( Graphics g, int width, int height, Canvas observer ) {
      if (working) {
          g.drawImage( robotImage, column*width+2, row*height+2, width-4, height-4,
observer );
      } else {
          g.setColor(Color.black);
          g.fillRect(column*width+2,row*height+2,
                   width-4, height-4);

      }
    } // end drawRobot


    public void turnToJunk() {
      working = false;
    } // end turnToJunk


    public boolean isWorking () {
      return working;
    }

} // end class Robot
```