```
import java.awt.*;
import java.applet.*;
/*********************
*                    *   Knight's Tour Problem
*                    *   Starting from any square on the che
*                    *   the Knight has to cover all the 64
*                    *   exactly 64 moves.
*                    *   Author  Vasanth Desai
*                    *   Email vasanthdesai@usa.net
*                    *   (c) 1998 Vasanth Desai
*********************
*            History
*            I read about this problem sometime around late
*            in a Sunday magazine as a puzzle, and solved it
*            10 minutes and noted down the path.
*            Late I thougt I would write a general program t
*            path from any starting position if it exists.
*            I tried it sometime in late seventies On CDC CY
*            (Erlangen University) and could not examine mov
*            beyond 20-25 due to combinatorial explosion. (S
*            it was obvious that there would be combinatoria
*            should not have tried it in the first place. Th
*            hit the  right path in the first few alternativ
*            have to store/examine all the alternatives. It
*            Such problems require heuristics to tackle.
*            Though I devised a heuristics I did not code it
*            was quite messy. I made another attempt in 1983
*
*            20/4/98
*            I decided to try to sove the problem using Java
*            languages make it really easy to tackle real wo
*            I could solve the entire problem in about 3 hou
*            Full credits to Java
*
*/



/*
      8 possible moves from X
      for the knight
     ___________________
    |   | 8 |   | 1 |   |
    |___|___|___|___|___|
    | 7 |   |   |   | 2 |
    |___|___|___|___|___|
    |   |   |   |   |   |
    |___|___|_X_|___|___|
    | 6 |   |   |   | 3 |
    |___|___|___|___|___|
    |   | 5 |   | 4 |   |
    |___|___|___|___|___|
*/

/*
   The class Square represents a square in a chess board. Wh
   who are are neighbours we should know about the container
    The class Board represents the chess board and each squa
    the board via the variable b which is set when 'squares'
    There is sister class of Square, CSquare which handles v
    the Square. The CSquare objects are embedded in an insta
    class. MyFrame is derived from Frame and hosts the chess
    can resize the window. The frame  is a popup window.

    The class KnightsTour is the 'main' class. It glues toge
    objects. It is designed so that it works both as an Appl
*/
//
```

```
//  The code is written in Java 1.0.2. It is straightforward
//  later versions. The conversion which requires a little ca
//  When the user clicks on a square, the CSquare objects del
//  MyFrame conveying id of the CSquare.
//
//  About the source code: The source code, alas, does not in
//  programming practices. For example accessibilty of variab
//  are not always defined. The methods of one class peek and
//  other classes directly. One should have used get/set meth
//  scarce. If you have not tackled similar problems, underst
//  is not very easy.
//  The point is, the project was to try the heuristics to so
//  the Knight's Tour problem,and to write best and efficient
//  to do so was secondary.


    class Square {
        int id;
        Board b = null;
        boolean visited = false;
        Square(int n) { id = n;}
        Square(int n,Board b){ id =n; this.b = b;}
        int row(){ return id/8+ 1;}
        int col(){ return (id%8) + 1;}
        int getId(int r, int c) { return (r-1)*8+c-1;}
        // returns  legal moves from this square
        int[] next(){
             int p[] = new int[8];
             int m = 0,r,c;
             for(int i=0; i < 8;i++){
                r = this.row(); c = this.col();
                switch(i){
                 case 0:   r -= 2; c += 1; break;
                 case 1:   r -= 1; c += 2; break;
                 case 2:   r += 1; c += 2; break;
                 case 3:    r += 2; c += 1 ; break;
                 case 4:   r += 2 ; c -= 1; break;
                 case 5:   r += 1; c -= 2; break;
                 case 6:   r -= 1; c -= 2; break;
                 case 7:   r -= 2; c -= 1; break;
                }
                if(( r < 1) || ( r > 8) || ( c < 1) || (c >
                p[m] = getId(r,c);
                m++;
             }
             int t[]= new int[m];
             System.arraycopy(p,0,t,0,m);
             return t;
        }
        // returns the number of moves from the next jump.
        // takes into account of visited squares
        int escapes(int omit){
            int nxt[];
            nxt = next();
            int e=0;
            for(int i = 0; i < nxt.length; i++){
                if(b.sq[nxt[i]].visited || (nxt[i]== omit))
                e++;
            }
            return e;
        }
        // this is the HEURISTICS
        // what is the best jump from the square ?
        // -1 if none. Note that you may get trapped into a
        // come out since all possible escape squares are al
        int goodExit(){
            int nxt[];
            nxt = next();
```

```
                int k = 8;
                int idx = -1;
                int e = 0;
                for(int i = 0; i < nxt.length; i++) {
                    if(b.sq[nxt[i]].visited) continue;
                    e = b.sq[nxt[i]].escapes(nxt[i]);
                    if((e > 0) && ( e < k)) { k = e; idx = i;}
                }
                if(idx == -1) return idx;
                else
                return nxt[idx];
            }
        }
        class Board  extends Thread{
            Square sq[];
            CSquare csq[];
            boolean running = false;  // set true while the knig
            int delayInterval = 500;
            int stsq = 0;
            Board(){
                sq = new Square[64];
                for(int i = 0; i < 64; i++) sq[i] = new Square(i
            }
            // Returns the path. The problem was first solved wi
            // encumberences. At that this method allowed me 'pr
            // path on stdout and check the logic
            int[]findPath(int startSquare){
                int i  = 0;
                int path[];
                path = new int[64];
                for(i = 0; i < 64; i++) sq[i].visited = false;
                sq[startSquare-1].visited = true;
                int nxt[];
                int n = -1;
                int moves = 0;
                int currentSquare = startSquare-1;
                path[currentSquare] = moves+1;
                for( i = 0; i < 64; i++) {
                    n = sq[currentSquare].goodExit();
                    if( n < 0 ) break;
                    sq[n].visited = true;
                    moves++;
                    currentSquare = n;
                    // remember path path
                    path[currentSquare] = moves+1;
                }
                // find out unvisited squares
                int nUnvisited = 0;
                int unvisited = -1;
                for(i = 0; i < 64; i++){
                    if(!sq[i].visited){
                        unvisited = i;
                        nUnvisited++;
                    }
                }
                if(nUnvisited == 1) {
                    nxt = sq[currentSquare].next();
                    for(i = 0;i < nxt.length; i++) {
                        if(nxt[i] == unvisited) {
                            path[unvisited]=moves+2;
                            break;
                        }
                    }
                } else {
                    // System.out.println("Failed to find soluti
                    // System.exit(1);
                    return null;
                }
```

```
    return path;
}
// Engine
public void run(){
    running = true;
   showPath(stsq);
    running = false;
    stop();
}
// Visually show the knignts moves on the chess boar
// Sleep a little after each so that user is not sus
void showPath(int startSquare){
    int i  = 0;
    int path[];
    path = new int[64];
    for(i = 0; i < 64; i++){
         sq[i].visited = false;
         csq[i].num = -1;
         csq[i].hval = -1;
         csq[i].update(csq[i].getGraphics());
    }
    sq[startSquare-1].visited = true;
    int nxt[];
    int n = -1;
    int moves = 0;
    int currentSquare = startSquare-1;
    path[currentSquare] = moves+1;
    csq[currentSquare].num = moves+1;
    csq[currentSquare].hval = 1;
 csq[currentSquare].update(csq[currentSquare].getGra
    for( i = 0; i < 64; i++) {
        try {
            sleep(delayInterval);
        } catch(InterruptedException e){
        }
        n = sq[currentSquare].goodExit();
        if( n < 0 ) break;
        sq[n].visited = true;
        csq[currentSquare].hval = -1;
    csq[currentSquare].update(csq[currentSquare].get

        moves++;
        currentSquare = n;
        // remember path
        path[currentSquare] = moves+1;
        csq[currentSquare].num = moves+1;
        csq[currentSquare].hval = 1;
    csq[currentSquare].update(csq[currentSquare].get
    }
    // Find out unvisited squares
    // Since the goodExit always looks one ahead, if
    // is OK when we arrive here theres should be ex
    // unvisited square and it should be reachale fr
    // square
    int nUnvisited = 0;
    int unvisited = -1;
    for(i = 0; i < 64; i++){
        if(!sq[i].visited){
            unvisited = i;
            nUnvisited++;
        }
    }
    if(nUnvisited == 1) {
        nxt = sq[currentSquare].next();
        for(i = 0;i < nxt.length; i++) {
            if(nxt[i] == unvisited) {
                csq[currentSquare].hval = -1;
```

```
                 csq[currentSquare].update(csq[currentSquar

                       path[unvisited]=moves+2;
                       csq[unvisited].num = moves+2;
                       csq[unvisited].repaint();
                       break;
                   }
               }
           } else {
               // System.out.println("Failed to find soluti
               // System.exit(1);
               return;
           }
           return ;
       }

   }
   public class KnightsTour extends Applet {
       boolean inApplet = false;
       int width = 280;
       int height = 180;
       public static void main(String argv[]){
       int i  = 0;
       int N = 250;
       if(argv.length > 0) {
           try {
               N = Integer.parseInt(argv[0]);
           } catch (NumberFormatException e) {
               System.out.println(e.getMessage());
               System.exit(1);
           }
       }
       int path[];
       Board board = null;
       MyFrame fr= new MyFrame();
       fr.setLayout(new GridLayout(8,8));
       fr.board = board;
       CSquare sqr[] = new CSquare[64];
       for(i = 0; i < 64; i++) {
            sqr[i] = new CSquare();
            sqr[i].id = i;
            fr.add(sqr[i]);
       }
       fr.csq = sqr;
       fr.validate();
       fr.resize(200,200);
       fr.show();
   }
   MyFrame frame;Board board;
   Button fast,slow,show,hide;
   public void  init(){
       inApplet = true;
       int i  = 0;
       int N = 0;
       setLayout(new BorderLayout());
       fast = new Button("Faster");
       slow = new Button("Slower");
       show = new Button("Show");
       hide = new Button("Hide");
       hide.disable();
       Panel p = new Panel();
       p.add(fast); p.add(slow);p.add(show); p.add(hide);
       add("South",p);
       frame = new MyFrame();
       frame.setLayout(new GridLayout(8,8));
       frame.board = board;
       frame.inApplet = true;
       frame.show = show;
```

```
            frame.hide = hide;
            CSquare sqr[] = new CSquare[64];
            frame.csq = sqr;
            for(i = 0; i < 64; i++) {
                 sqr[i] = new CSquare();
                 sqr[i].id = i;
                 frame.add(sqr[i]);
            }
            width = size().width; height = size().height;
            resize(width,height);
            frame.resize(200,200);
            board = null;
        }
        public Dimension minimumSize(){
            return new Dimension(width,height);
        }
        public Dimension preferredSize(){
            return minimumSize();
        }
        public void paint(Graphics g){
            g.drawRect(0,0,size().width-1,140);
            String s = "Knight's Tour";
            g.setFont(new Font("Serif",Font.ITALIC,24));
            FontMetrics fm = g.getFontMetrics();
            int len = fm.stringWidth(s);
            g.drawString(s,(size().width-len)/2,50);
            g.setFont(new Font("Arial",Font.BOLD,12));
            fm = g.getFontMetrics();
            s = "Click on Show button to see Chess Board";
            len = fm.stringWidth(s);
            g.drawString(s,(size().width-len)/2,75);
            s = "Click on any square to start tour from that squ
            len = fm.stringWidth(s);
            g.drawString(s,(size().width-len)/2,100);
            s = "After the start, click on any square to stop";
            len = fm.stringWidth(s);
            g.drawString(s,(size().width-len)/2,125);
        }
        int interval = 250;
        public boolean handleEvent(Event e) {
            Object target = e.target;
                if(e.id == Event.ACTION_EVENT){
                    if(e.target == show){
                        frame.show();
                        show.disable();
                        hide.enable();
                        return true;
                    }
                    if(e.target == hide){

                        frame.hide();
                        hide.disable();
                        show.enable();
                        return true;

                    }
                    if(e.target == fast){
                      if(board != null) {
                            board.delayInterval -= 50;
                            if(board.delayInterval < 50)
    board.delayInterval = 50;
                        }else {
                            frame.interval -= 50;
                            if(frame.interval < 50) frame.interv
                        }
                        return true;
                    }
                    if(e.target == slow){
                        if(board != null) {
                            board.delayInterval += 50;
                            if(board.delayInterval >2000)
    board.delayInterval = 2000;
                        } else {
                            frame.interval += 50;
                            if(frame.interval > 2000) frame.inte
    2000;
                        }

                        return true;
                    }

                }
            return super.handleEvent(e);
        }
        public void stop(){
            frame.hide();
            show.enable();
            hide.disable();
        }
    }
    class MyFrame extends Frame  {
        Board board;
        CSquare csq[];
        boolean inApplet = false;
        int interval = 250;
        Button show,hide;
        public MyFrame(){
            super("Knight's Tour");
        }
    // Deprecated in Java 1.1 onwards. Use event listners instea
        public boolean handleEvent(Event event) {
            if((event.id >= 5000)&&(event.id < 5064)){
               if((board != null)&& board.running)  {
                   csq = board.csq;
                    board.stop();
                    board = null;
                    return true;
                }
                if((board != null) && !board.running) board = nu
                if(board == null) {
                    board = new Board();
                    board.csq = csq;
                    board.stsq = event.id-5000+1;
                    board.delayInterval = interval;
                    board.start();
                }

            }
          if(!inApplet && (event.id == Event.WINDOW_DESTROY)) {
                dispose();
                System.exit(0); // necessary we should go to DOS
             return super.handleEvent(event);
          }
            if(inApplet && (event.id == Event.WINDOW_DESTROY)) {
                hide();
                show.enable();
                hide.disable();
             return true ;
          }

          return super.handleEvent(event);
        }
    }
    class CSquare extends Canvas{
        int h,w;
        int id;
        int num = -1;
        int hval = -1;
        CSquare(int w, int h){
            this.w = w; this.h = h;
        }
        CSquare(){
            h = 18;
            w = 18;
        }
        synchronized public void paint(Graphics g){
            w = size().width;h = size().height;
            int r =  id/8;
            int c = (id%8);
            if(hval == -1){
                if( ((r+c)%2) == 0) g.setColor(Color.lightGray);
                else
                g.setColor(Color.gray /* new Color(192,192,192)*
            }
            else g.setColor(Color.red);
            g.fillRect(0,0,w,h);
            g.setColor(Color.black);
            g.drawRect(1,1,w-1,h-1);

            String s ;
            if(num != -1){
                s = num+"";
                int len = g.getFontMetrics().stringWidth(s);
                int ht = g.getFontMetrics().getHeight();
                if( ((r+c)%2) == 0) g.setColor(Color.black);
                else
                g.setColor(Color.white);
                g.drawString(s,(w-len)/2,(h+ht)/2);
            }
        }
        public void update(Graphics g){

            paint(g);
        }
        // Deprecated in Java 1.1 onwards. Use new Event model
       synchronized public boolean handleEvent(Event e) {
            Object target = e.target;
            if(e.id == Event.MOUSE_UP){
//            System.out.println("up");
              getParent().deliverEvent(new Event(getParent(),500
              return true;
            }
          return super.handleEvent(e);
        }
// Deprecated in Java 1.1 onwards. Use get functions
         public Dimension minimumSize() {
            return new Dimension(w,h);
    }
        public Dimension preferredSize() {
            return minimumSize();
    }

}
// Improvements in packaging 22/04/98


// The author welcomes your comments
// V P Desai
// 6 Pushpamedh
// 39/40 Tulsibagwale Colony
// Parvati
// Pune 411 004
// India
```