CS602 Final Project

Sokoban Game

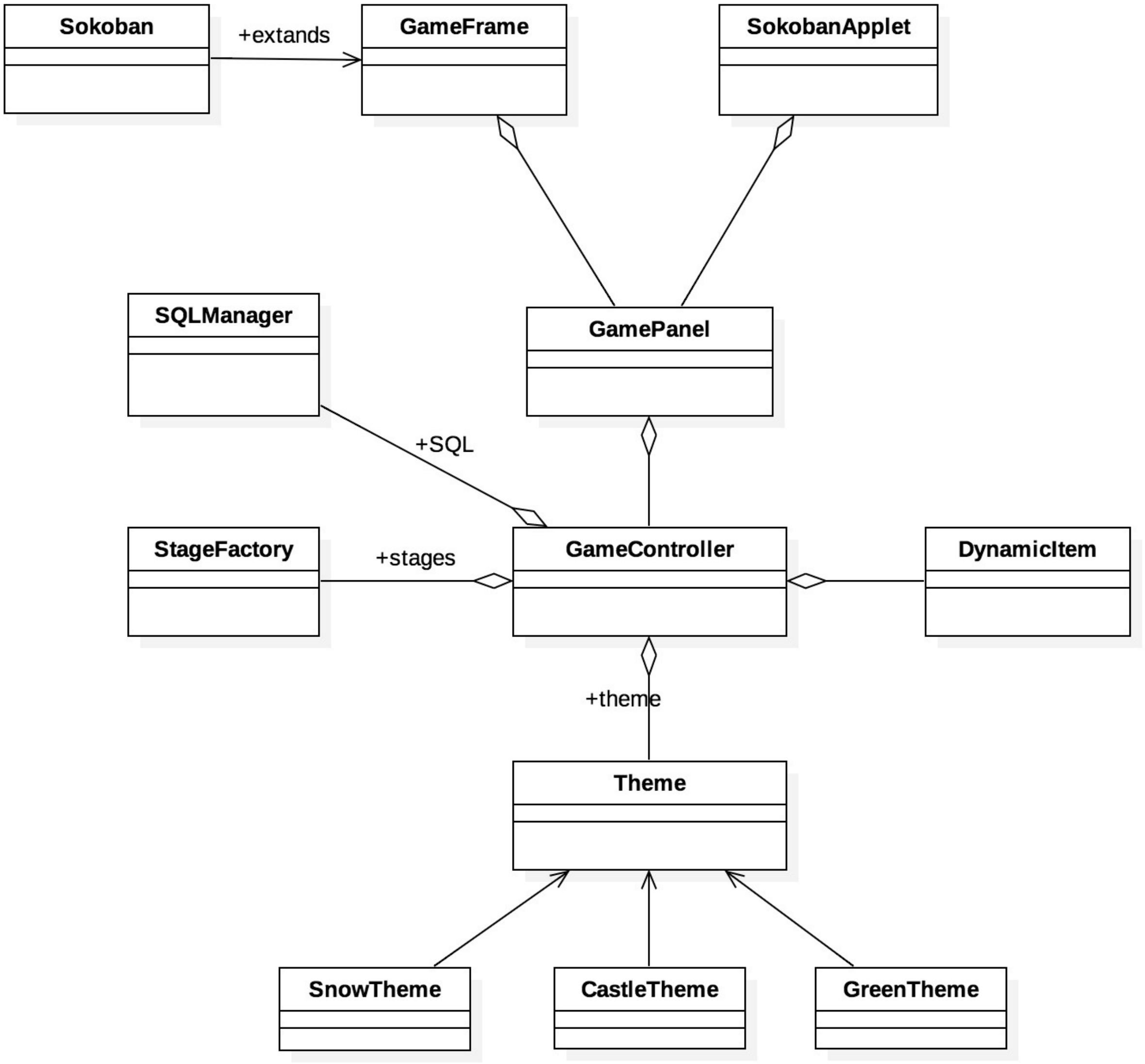
Zhonghua Qin

1. Planning

1) Features

- The Sokoban game
- Application and Applet format.
- Animation when hero walk and push box.
- User can use Up, Down, Left and Right Keys to control the hero move
- hero can not go through wall and box
- The box can not go through wall
- Sound effect when the hero walking, pushing box, hit the wall or box and victory.
- Use menu to go to the next stage or previous stage
- Use menu to select stage in 1~20 range
- Error handle. When user input number not in 1~20, provide user alert dialog.
- New game to reset game.
- Select theme from four different themes.
- Save game and load game from NJIT MySQL server. (Only supports to Application. Because Applet has accessibility limit.)

2) UML Graph


Sokoban
+extands
GameFrame
SokobanApplet
SQLManager
GamePanel
+SQL
StageFactory
+stages
GameController
DynamicItem
+theme
Theme
SnowTheme
CastleTheme
GreenTheme

3) JUnit

```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename StageFactoryTest.java 
 *  @author Zhonghua Qin
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Zhonghua Qin
 */
public class StageFactoryTest {

    public StageFactoryTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of getStages method, of class StageFactory.
     */
    @Test
    public void testGetStages() {
      System.out.println("getStages");
      int expResult = 20;
      int result = StageFactory.getStages().length;
      assertEquals(expResult, result);
    }

}
```

2. Code List

Sokoban

```java
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename Sokoban.java 
 *  @author Zhonghua Qin
 *  @datetime Nov 28, 2015  6:21:30 PM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

/**
 * Sokoban application entrance.
 * @author Zhonghua Qin
 */
public class Sokoban extends GameFrame{

}
```

GameFrame

```java
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename GameFrame.java 
 *  @author Zhonghua Qin
 *  @datetime Nov 28, 2015  6:23:48 PM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;

/**
 * Sokoban game frame.
 * @author Zhonghua Qin
 */
public class GameFrame extends javax.swing.JFrame {

    /**
     * Creates new form GameFrame
     */
    public GameFrame() {
        initComponents();
      int numOfThemes = Theme.themes.length;
      ActionListener listener = (ActionEvent e) -> {
          String command = e.getActionCommand();
          ((GamePanel)gamePanel).setTheme(Integer.valueOf(command));
      };
     for (int i = 0; i < numOfThemes; i++) {
         JMenuItem item = new JMenuItem(Theme.themes[i].getName());
         item.setActionCommand(String.valueOf(i));
         item.addActionListener(listener);
         themeMenu.add(item);
     }
    }

    /**
     * This method is called from within the constructor to initialize the form.
WARNING: Do NOT modify this code. The content of this method is always regenerated by
the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        gamePanel = new GamePanel();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu3 = new javax.swing.JMenu();
        jMenuItem5 = new javax.swing.JMenuItem();
        jMenuItem6 = new javax.swing.JMenuItem();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();
        jMenuItem3 = new javax.swing.JMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem4 = new javax.swing.JMenuItem();
        themeMenu = new javax.swing.JMenu();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Sokoban_ZhonghuaQin");
        setMaximumSize(new java.awt.Dimension(330, 370));
        setMinimumSize(new java.awt.Dimension(330, 370));
        setPreferredSize(new java.awt.Dimension(330, 370));
        setResizable(false);
        setSize(new java.awt.Dimension(330, 370));
        addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyPressed(java.awt.event.KeyEvent evt) {
                formKeyPressed(evt);
            }
        });

        javax.swing.GroupLayout gamePanelLayout = new
javax.swing.GroupLayout(gamePanel);
        gamePanel.setLayout(gamePanelLayout);
        gamePanelLayout.setHorizontalGroup(

gamePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 440, Short.MAX_VALUE)
        );
        gamePanelLayout.setVerticalGroup(

gamePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 410, Short.MAX_VALUE)
        );

        jMenu3.setText("File");

jMenuItem5.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S, java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem5.setText("Save");
        jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem5ActionPerformed(evt);
            }
        });
        jMenu3.add(jMenuItem5);

jMenuItem6.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_L, java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem6.setText("Load");
        jMenuItem6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem6ActionPerformed(evt);
            }
        });
        jMenu3.add(jMenuItem6);

        jMenuBar1.add(jMenu3);

        jMenu1.setText("Command");

jMenuItem1.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_N, java.awt.event.InputEvent.CTRL_MASK));
        jMenuItem1.setText("New Game");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem1);

jMenuItem2.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_RIGHT, java.awt.event.InputEvent.SHIFT_MASK));
        jMenuItem2.setText("Next Stage");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem2);

jMenuItem3.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_LEFT, java.awt.event.InputEvent.SHIFT_MASK));
```

```
        jMenuItem3.setText("Previous Stage");
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem3);

        jMenuBar1.add(jMenu1);

        jMenu2.setText("Select Stage");

        jMenuItem4.setText("1~20");
        jMenuItem4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem4ActionPerformed(evt);
            }
        });
        jMenu2.add(jMenuItem4);

        jMenuBar1.add(jMenu2);

        themeMenu.setText("Themes");
        jMenuBar1.add(themeMenu);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(gamePanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(gamePanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void formKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
     ((GamePanel)gamePanel).keyPressed(evt);
    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
     ((GamePanel)gamePanel).newGame();
    }
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
      ((GamePanel)gamePanel).nextStage();
    }

    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
      ((GamePanel)gamePanel).backStage();
    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
      String s = JOptionPane.showInputDialog("Input Stage",
((GamePanel)gamePanel).getController().getCurrentStage()+1);
      System.out.println(s);
      try{
          int n = Integer.parseInt(s);
          System.out.println(n);
          if (n>=1 && n<=20){
            ((GamePanel)gamePanel).setStage(n);
          }else {
            JOptionPane.showMessageDialog(gamePanel,"The number must in [1,20] range.");
          }
      }catch (NumberFormatException e2){
          JOptionPane.showMessageDialog(gamePanel,"Please input number.");
      }
    }

    private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
      ((GamePanel)gamePanel).saveGame();
    }

    private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
      ((GamePanel)gamePanel).loadGame();
    }

    /**
     * main
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code 
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default 
look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/
lookandfeel/plaf.html 
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```

```
                if ("Mac OS X".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(GameFrame.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(GameFrame.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(GameFrame.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(GameFrame.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new GameFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JPanel gamePanel;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JMenuItem jMenuItem5;
    private javax.swing.JMenuItem jMenuItem6;
    private javax.swing.JMenu themeMenu;
    // End of variables declaration
}
```

GamePanel

```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename GamePanel.java 
 *  @author Zhonghua Qin
 *  @datetime Nov 28, 2015  6:24:30 PM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JPanel;

/**
 * Sokoban game panel
 * @author Zhonghua Qin
 */
public class GamePanel extends JPanel {
    private final GameController controller;
    private static final int FRAMES = 60;

    /**
     * GamePanel Constructor
     */
    public GamePanel() {
      controller = new GameController(this);

        DisplayThread displayThread = new DisplayThread();

        displayThread.start();

    }

    /**
     * Get game controller
     * @return GameController
     */
    GameController getController() {
      return controller;
    }

    /**
     * keyPressed event
     * @param e KeyEvent
     */
    void keyPressed(KeyEvent e) {
      controller.move(e.getKeyCode());
    }

    /**
     * new game
     */
    void newGame() {
      controller.newGame();
    }

    /**
```

```
     * next stage
     */
    void nextStage() {
     controller.nextStage();
    }

    /**
     * previous stage
     */
    void backStage() {
     controller.backStage();
    }

    /**
     * set stage
     * @param n int
     */
    void setStage(int n) {
     controller.setStage(n);
    }

    /**
     * save game
     */
    void saveGame() {
     controller.saveGame();
    }

    /**
     * load game
     */
    void loadGame() {
     controller.loadGame();
    }

    /**
     * set theme
     * @param theme int
     */
    void setTheme(int theme) {
     controller.setTheme(theme);
    }

    private class DisplayThread extends Thread{
     @Override
     public void run() {
         while (true) {
                repaint();
          try {
               Thread.sleep(1000/FRAMES);
          } catch (InterruptedException ex) {
               Logger.getLogger(GamePanel.class.getName()).log(Level.SEVERE, null, ex);
          }
         }
     }
    }

    /**
     * paint method
     * @param g Graphics
     */
    @Override
    public void paint(Graphics g) {
        controller.draw((Graphics2D)g);
    }

}
```

GameController

```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename GameController.java 
 *  @author Zhonghua Qin
 *  @datetime Nov 29, 2015  8:50:21 AM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Point;
import java.awt.RenderingHints;
import java.awt.event.KeyEvent;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import javax.imageio.ImageIO;
import sun.audio.AudioPlayer;
import sun.audio.AudioStream;

/**
 * Sokoban game controller
 * @author Zhonghua Qin
 */
public class GameController {

    GamePanel gamePanel;

    /**
     * Floor index
     */
    protected static final int FLOOR = 0;
```

```java
    /**
     * Wall index
     */
    protected static final int WALL = 1;

    /**
     * Target index
     */
    protected static final int TARGET = 2;

    /**
     * Box index
     */
    protected static final int BOX = 3;

    /**
     * Player index
     */
    protected static final int PLAYER = 4;

    /**
     * Outside index
     */
    protected static final int OUTSIDE = 9;

    /**
     * Block size
     */
    protected static final int BLOCK = 55;

    /**
     * Map size
     */
    protected static final int MAPSIZE = 6;

    /**
     * Player width
     */
    protected static final int PLAYER_W = BLOCK * 2/3;

    /**
     * Player heigh
     */
    protected static final int PLAYER_H = BLOCK;

    /**
     * Target width
     */
    protected static final int TARGET_W = BLOCK * 2/3;

    /**
     * map
     */
    protected int [][] map = new int[MAPSIZE][MAPSIZE];
    private int currentStage;

    /**
     * box
     */
    protected List<DynamicItem> box = new LinkedList();
    private List<DynamicItem> targets = new LinkedList();

    /**
     * player
     */
    protected DynamicItem player;

    private Theme theme;

    /**
     * Constructor
     * @param gamePanel GamePanel
     */
    public GameController(GamePanel gamePanel) {
      this.gamePanel = gamePanel;

      theme = Theme.newTheme(0);

      currentStage = 0;
      initGame(true);
      URL imgURL = getClass().getResource("R/images/Congratulations.png");
      try{
          congratulationsImage = ImageIO.read(imgURL);
      } catch (IOException e) {
          e.printStackTrace();
      }

    }

    private void initGame(boolean withStage) {
      isPlaying = true;
      isFinished = false;

      if (withStage) {
          for (int i = 0; i < MAPSIZE; i++) {
            for (int j = 0; j < MAPSIZE; j++) {
                map[i][j] = StageFactory.getStages()[currentStage][i][j];
            }
          }
      }
      box.clear();
      targets.clear();


      for (int i = 0; i < MAPSIZE; i++) {
          for (int j = 0; j < MAPSIZE; j++) {
            switch (map[i][j]) {
                case PLAYER:
                  player = new DynamicItem(theme.getPlayerDownImages().get(0),
                      j, i,
                      j*BLOCK+(BLOCK-PLAYER_W)/2,
                      i*BLOCK+(BLOCK-PLAYER_H)/2,
                      PLAYER_W, PLAYER_H);
                break;
               case BOX:
                box.add(new DynamicItem(theme.getBoxImages().get(0), j, i, j*BLOCK,
i*BLOCK, BLOCK, BLOCK));
                break;
               case TARGET:
                targets.add(new DynamicItem(theme.getTargetImages().get(0),
                      j, i,
                      j*BLOCK+(BLOCK-TARGET_W)/2,
                      i*BLOCK+(BLOCK-TARGET_W)/2, TARGET_W, TARGET_W));
                break;
               case PLAYER+TARGET:
                player = new DynamicItem(theme.getPlayerDownImages().get(0),
                      j, i,
                      j*BLOCK+(BLOCK-PLAYER_W)/2,
                      i*BLOCK+(BLOCK-PLAYER_H)/2,
                      PLAYER_W, PLAYER_H);
                targets.add(new DynamicItem(theme.getTargetImages().get(0),
                      j, i,
                      j*BLOCK+(BLOCK-TARGET_W)/2,
                      i*BLOCK+(BLOCK-TARGET_W)/2, TARGET_W, TARGET_W));
                break;
               case BOX+TARGET:
                box.add(new DynamicItem(theme.getBoxCompletedImages().get(0), j, i,
j*BLOCK, i*BLOCK, BLOCK, BLOCK));
                targets.add(new DynamicItem(theme.getTargetImages().get(0),
                      j, i,
                      j*BLOCK+(BLOCK-TARGET_W)/2,
                      i*BLOCK+(BLOCK-TARGET_W)/2, TARGET_W, TARGET_W));
                break;
               default:
                break;
           }
          }
      }
      System.out.println("Player X:"+player.x+", Y:"+player.y);
      isPlaying = false;
    }

    /**
     * draw method
     * @param g2d Graphics2D
     */
    public void draw(Graphics2D g2d){

g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION,RenderingHints.VALUE_INTERPOLATION
_BILINEAR);

        //Background
        for (int i = 0; i < MAPSIZE; i++)
            for (int j = 0; j < MAPSIZE; j++) {
```

```
           if (map[i][j] == WALL) {
               g2d.drawImage(theme.getWallImages().get(0), j*BLOCK, i*BLOCK, BLOCK,
BLOCK, null);
           }else{
               g2d.drawImage(theme.getFloorImages().get(0), j*BLOCK, i*BLOCK, BLOCK,
BLOCK, null);
           }
            }

      //Dynamic items
      for (DynamicItem target : targets) {
          g2d.drawImage(theme.getTargetImages().get(0), target.dX, target.dY,
target.getWide(), target.getHeigh(), null);
      }
      for (DynamicItem b : box) {
          if (map[b.y][b.x] == TARGET+BOX) {
           b.image = theme.getBoxCompletedImages().get(0);
          }else if (map[b.y][b.x] == BOX){
           b.image = theme.getBoxImages().get(0);
          }
          g2d.drawImage(b.image, b.dX, b.dY, b.getWide(), b.getHeigh(), null);
      }
      g2d.drawImage(player.image, player.dX, player.dY, player.getWide(),
player.getHeigh(), null);

      g2d.setFont(new Font("Arial", Font.BOLD, 30));
      g2d.setColor(Color.white);
      g2d.drawString(String.valueOf(currentStage+1), 15, 30);

      if (isFinished) {
          g2d.drawImage(congratulationsImage, 0, 50, 330, 200, null);
      }
    }

    /**
     * Get current stage
     * @return currentStage
     */
    public int getCurrentStage() {
      return currentStage;
    }

    private boolean isPlaying = false;
    private Timer timer;
    private void playAnimation(DynamicItem item, List<Image> frames, Point
distancePoint, int cycle, int time){
      if(item == null || frames.isEmpty())
          return;
      isPlaying = true;
      timer = new Timer();

      List<Image> allFrames = new LinkedList<>();
      if (frames.size() > 1) {
          List<Image> midFrames = new LinkedList<>();
          midFrames.addAll(frames);
          midFrames.remove(midFrames.size()-1);

          for (int i = 0; i < cycle; i++) {
           allFrames.addAll(midFrames);
          }
          allFrames.add(frames.get(frames.size()-1));
      }else{
          for (int i = 0; i < cycle; i++) {
           allFrames.addAll(frames);
          }
      }

      int spanT = time/allFrames.size();
      int startX = item.dX;
      int startY = item.dY;
      double spanX = ((double)distancePoint.x)/(allFrames.size()-1);
      double spanY = ((double)distancePoint.y)/(allFrames.size()-1);
      System.out.println("playAnimation");
      timer.schedule(new TimerTask() {
          int n = 0;
          @Override
          synchronized public void run() {
//         System.out.println("Playing animation");

           item.image = allFrames.get(n);
           item.dX = startX + (int)(spanX*n);
           item.dY = startY + (int)(spanY*n);
           n++;
           if (n == allFrames.size()) {
               item.x = item.tmpX;
               item.y = item.tmpY;
               isPlaying = false;
               cancel();
               if (item == player) {
                checkFinish();
               }
           }
          }
      }, 0, spanT);
    }
    private Image congratulationsImage;
    private boolean isFinished = false;

    private void checkFinish(){
      int boxnum = 0;
      int targetnum = 0;
      for (int i = 0; i < MAPSIZE; i++) {
          for (int j = 0; j < MAPSIZE; j++) {
           if (map[i][j] == BOX) {
               boxnum++;
           }else if (map[i][j] == TARGET) {
               targetnum ++;
           }
          }
      }
      if (boxnum == 0 && targetnum == 0) {
          finishedGame();
      }
    }

    private void finishedGame() {
      isFinished = true;
      try{
          URL auURL = getClass().getResource("R/sounds/wa.wav");
          AudioStream as = new AudioStream(auURL.openStream());
          AudioPlayer.player.start(as);
      } catch (FileNotFoundException e) {
          e.printStackTrace();
      } catch (IOException e) {
          e.printStackTrace();
      }
      Timer timer = new Timer();
      timer.schedule(new TimerTask() {
          @Override
          public void run() {
           System.out.println(".run()");
           cancel();
           nextStage();
          }
      }, 2000);
    }

    private void stopAnimation(){
      if (timer != null) {
          timer.cancel();
      }
    }

    /**
     * move when key pressed
     * @param direction int
     */
    public void move(int direction){
      if (isPlaying || isFinished) {
          return;
      }
      int newX;
        int newY;
        int crossX;
        int crossY;
      List<Image> framesImages = new ArrayList<>();
        switch(direction){
            case KeyEvent.VK_UP:
                System.out.println("Up");
                newX = player.x;
                newY = player.y - 1;
                crossX = newX;
                crossY = newY - 1;
           framesImages.addAll(theme.getPlayerUpImages());
                break;
            case KeyEvent.VK_DOWN:
                System.out.println("Down");
                newX = player.x;
                newY = player.y + 1;
                crossX = newX;
                crossY = newY + 1;
           framesImages.addAll(theme.getPlayerDownImages());
                break;
            case KeyEvent.VK_LEFT:
                System.out.println("Left");
                newX = player.x - 1;
                newY = player.y;
                crossX = newX - 1;
                crossY = newY;
           framesImages.addAll(theme.getPlayerLeftImages());
                break;
            case KeyEvent.VK_RIGHT:
                System.out.println("Right");
                newX = player.x + 1;
                newY = player.y;
                crossX = newX + 1;
                crossY = newY;
           framesImages.addAll(theme.getPlayerRightImages());
                break;
            default:
                return;
        }
        if (map[newY][newX] == FLOOR || map[newY][newX] == TARGET) {
            map[newY][newX] += PLAYER;
            map[player.y][player.x] -= PLAYER;
          //Move
          playAnimation(player, framesImages,
               new Point((newX - player.x)*BLOCK, (newY - player.y)*BLOCK), 2, 1000);
          try{
           URL auURL = getClass().getResource("R/sounds/walk.wav");
           AudioStream as = new AudioStream(auURL.openStream());
           AudioPlayer.player.start(as);
          } catch (FileNotFoundException e) {
           e.printStackTrace();
          } catch (IOException e) {
           e.printStackTrace();
          }
          player.tmpX = newX;
            player.tmpY = newY;
          System.out.println("Player X:"+player.x+", Y:"+player.y);
        }else if(map[newY][newX] == BOX || map[newY][newX] == BOX+TARGET){
          DynamicItem boxT = null;
          for (DynamicItem b : box) {
           if (b.x == newX && b.y == newY) {
               boxT = b;
           }
          }
            if (map[crossY][crossX] == FLOOR || map[crossY][crossX] == TARGET) {
                map[crossY][crossX] += BOX;
                map[newY][newX] = map[newY][newX] - BOX + PLAYER;
                map[player.y][player.x] -= PLAYER;
           //Move
           playAnimation(boxT, theme.getBoxImages(),
               new Point((crossX - boxT.x)*BLOCK, (crossY - boxT.y)*BLOCK), 5*2, 1000);

           boxT.tmpX = crossX;
           boxT.tmpY = crossY;
           playAnimation(player, framesImages,
               new Point((newX - player.x)*BLOCK, (newY - player.y)*BLOCK), 2, 1000);
           try{
               URL auURL = getClass().getResource("R/sounds/walk.wav");
               AudioStream as = new AudioStream(auURL.openStream());
               AudioPlayer.player.start(as);
           } catch (FileNotFoundException e) {
               e.printStackTrace();
           } catch (IOException e) {
               e.printStackTrace();
           }
           try{
               URL auURL = getClass().getResource("R/sounds/pushbox.wav");
               AudioStream as = new AudioStream(auURL.openStream());
               AudioPlayer.player.start(as);
           } catch (FileNotFoundException e) {
               e.printStackTrace();
           } catch (IOException e) {
               e.printStackTrace();
           }
                player.tmpX = newX;
                player.tmpY = newY;
           System.out.println("Player X:"+player.x+", Y:"+player.y);
           System.out.println("Box X:"+boxT.x+", Y:"+boxT.y);
            }else{
           try{
               URL auURL = getClass().getResource("R/sounds/hitbox.wav");
               AudioStream as = new AudioStream(auURL.openStream());
               AudioPlayer.player.start(as);
           } catch (FileNotFoundException e) {
               e.printStackTrace();
           } catch (IOException e) {
               e.printStackTrace();
           }
          }
        }else{
          try{
           URL auURL = getClass().getResource("R/sounds/hitwall.wav");
           AudioStream as = new AudioStream(auURL.openStream());
           AudioPlayer.player.start(as);
          } catch (FileNotFoundException e) {
           e.printStackTrace();
          } catch (IOException e) {
           e.printStackTrace();
          }
       }
     }
     /**
      * new game
      */
     void newGame(){
       stopAnimation();
       initGame(true);
     }

     /**
      * next stage
      */
     void nextStage() {
       if (currentStage+1 < StageFactory.getStages().length) {
          currentStage ++;
          newGame();
       }else
          newGame();
     }

     /**
      * previous stage
      */
     void backStage() {
       if (currentStage > 0) {
          currentStage --;
          newGame();
       }else
          newGame();
     }

     /**
      * set stage
      * @param n
      */
     void setStage(int n) {
       currentStage = n-1;
       newGame();
     }

     /**
      * save game
      */
     void saveGame() {
       SQLManager.save(map);
     }

     /**
      * load game
      */
     void loadGame() {

       int[][] loadMap = SQLManager.load();
       if (loadMap != null) {
          stopAnimation();
          isPlaying = true;
```

```
         isFinished = false;
         for (int i = 0; i < MAPSIZE; i++) {
           for (int j = 0; j < MAPSIZE; j++) {
               map[i][j] = loadMap[i][j];
           }
         }
         initGame(false);
      }
    }

    /**
     * set theme
     * @param t int 
     */
    void setTheme(int t) {
      theme = Theme.newTheme(t);
    }

}
```

DynamicItem

```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename DynamicItem.java 
 *  @author Zhonghua Qin
 *  @datetime Nov 29, 2015  10:09:33 AM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.awt.Image;

/**
 * Sokoban map item model
 * @author Zhonghua Qin
 */
public class DynamicItem {
    Image image;
    int x;
    int y;
    int tmpX;
    int tmpY;
    int dX;
    int dY;
    private int wide;
    private int heigh;

    /**
     * get wide
     * @return 
     */
    public int getWide() {
      return wide;
    }
    /**
     * get heigh
     * @return
     */
    public int getHeigh() {
     return heigh;
    }

    /**
     * Constructor
     * @param image Image
     * @param x int
     * @param y int
     * @param dX int
     * @param dY int
     * @param wide int
     * @param heigh  int
     */
    public DynamicItem(Image image, int x, int y, int dX, int dY, int wide, int heigh) {
     this.image = image;
     this.x = x;
     this.y = y;
     this.tmpX = x;
     this.tmpY = y;
     this.dX = dX;
     this.dY = dY;
     this.wide = wide;
     this.heigh = heigh;
    }

}
```

StageFactory

```
/*
 *   Copyright (C) 2015 Zhonghua Qin
 *   @filename StageFactory.java
 *   @author Zhonghua Qin
 *   @datetime Nov 29, 2015  6:53:07 PM
 *   @version 1.0
 *   @Description
 */
package Zhonghua;

/**
 * Stage data factory.
 * @author Zhonghua Qin
 */
public final class StageFactory {
    private static final int[][] stage1 = {
        {9, 9, 9, 9, 9, 9},
        {9, 1, 1, 1, 1, 1},
        {1, 1, 0, 0, 0, 1},
        {1, 6, 3, 0, 0, 1},
        {1, 1, 1, 1, 1, 1},
```

```java
        {9, 9, 9, 9, 9, 9}
    };
    private static final int[][] stage2 = {
        {9, 9, 9, 9, 9, 9},
        {1, 1, 1, 1, 1, 1},
        {1, 5, 5, 2, 2, 1},
        {1, 0, 3, 3, 5, 1},
        {1, 4, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage3 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 2, 0, 0, 1},
        {1, 0, 3, 4, 0, 1},
        {1, 0, 1, 3, 2, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage4 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 2, 0, 0, 1},
        {1, 0, 3, 4, 0, 1},
        {1, 0, 1, 3, 2, 1},
        {1, 0, 3, 0, 2, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage5 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 0, 3, 4, 2, 1},
        {1, 0, 1, 3, 2, 1},
        {1, 0, 3, 0, 2, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage6 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 2, 0, 1},
        {1, 0, 3, 4, 0, 1},
        {1, 2, 1, 3, 0, 1},
        {1, 0, 3, 0, 2, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage7 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 2, 3, 4, 0, 1},
        {1, 2, 1, 3, 0, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage8 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 2, 3, 2, 0, 1},
        {1, 0, 1, 3, 0, 1},
        {1, 2, 3, 4, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage9 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 0, 2, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 0, 1, 3, 4, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage10 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 2, 0, 0, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 0, 1, 3, 4, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage11 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 0, 2, 1},
        {1, 2, 3, 2, 0, 1},
        {1, 0, 1, 3, 3, 1},
        {1, 2, 3, 0, 4, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage12 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 2, 2, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 0, 1, 3, 3, 1},
        {1, 2, 3, 0, 4, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage13 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 2, 0, 2, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 0, 1, 3, 3, 1},
        {1, 2, 3, 0, 4, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage14 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 2, 0, 2, 1},
        {1, 0, 3, 0, 0, 1},
        {1, 2, 1, 3, 3, 1},
        {1, 2, 3, 0, 4, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage15 = {
        {1, 1, 1, 1, 1, 9},
        {1, 2, 6, 0, 1, 9},
        {1, 0, 0, 3, 1, 1},
        {1, 0, 3, 0, 0, 1},
        {1, 1, 0, 0, 0, 1},
        {9, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage16 = {
        {1, 1, 1, 1, 1, 1},
        {1, 6, 5, 0, 0, 1},
        {1, 2, 3, 0, 0, 1},
        {1, 0, 3, 0, 0, 1},
        {1, 1, 1, 0, 0, 1},
        {9, 9, 1, 1, 1, 1}
    };
    private static final int[][] stage17 = {
        {1, 1, 1, 1, 1, 1},
        {1, 2, 2, 6, 2, 1},
        {1, 3, 3, 3, 3, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage18 = {
        {9, 1, 1, 1, 1, 1},
        {9, 1, 4, 2, 2, 1},
        {1, 1, 3, 5, 3, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage19 = {
        {1, 1, 1, 1, 1, 9},
        {1, 2, 2, 0, 1, 9},
        {1, 0, 4, 3, 1, 1},
        {1, 0, 3, 0, 0, 1},
        {1, 1, 0, 0, 0, 1},
        {9, 1, 1, 1, 1, 1}
    };
    private static final int[][] stage20 = {
        {1, 1, 1, 1, 1, 1},
        {1, 0, 0, 2, 2, 1},
        {1, 0, 3, 0, 0, 1},
        {1, 0, 3, 5, 0, 1},
        {1, 1, 4, 0, 1, 1},
        {9, 1, 1, 1, 1, 9}
    };


    private static final int [][][] stages = {
     stage1, stage2,   stage3, stage4, stage5,
     stage6, stage7,   stage8, stage9, stage10,
     stage11, stage12, stage13, stage14, stage15,
     stage16, stage17, stage18, stage19, stage20
    };

    /**
     * get stages data
     * @return int[][][]
     */
    public static int[][][] getStages() {
     return stages;
    }
}
```

Theme

```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename Theme.java
 *  @author Zhonghua Qin
 *  @datetime Nov 29, 2015  9:38:04 AM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.awt.Image;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import javax.imageio.ImageIO;

/**
 * Theme class and theme static factory
 * @author Zhonghua Qin
 */
public class Theme {

    /**
     *name
     */
    protected String name = "Default";

    /**
     * playerUpImages
     */
    protected List<Image> playerUpImages = new ArrayList<>();

    /**
     * playerDownImages
     */
    protected List<Image> playerDownImages = new ArrayList<>();

    /**
     * playerLeftImages
     */
    protected List<Image> playerLeftImages = new ArrayList<>();

    /**
     * playerRightImages
     */
    protected List<Image> playerRightImages = new ArrayList<>();
    /**
     * wallImages
     */
    protected List<Image> wallImages = new ArrayList<>();

    /**
     * floorImages
     */
    protected List<Image> floorImages = new ArrayList<>();

    /**
     * boxImages
     */
    protected List<Image> boxImages = new ArrayList<>();

    /**
     * boxCompletedImages
     */
    protected List<Image> boxCompletedImages = new ArrayList<>();

    /**
     * targetImages
     */
    protected List<Image> targetImages = new ArrayList<>();
    /**
     * themes
     */
    public static final Theme[] themes = {
      new Theme(),
      new SnowTheme(),
      new CastleTheme(),
      new GreenTheme()
    };

    /**
     * new theme with index
     * @param t index
     * @return Theme
     */
    public static Theme newTheme(int t){
      return themes[t];
    }

    /**
     * Get name
     * @return
     */
    public String getName() {
      return name;
    }

    /**
     * Get wall images
     * @return List of Image
     */
    public List<Image> getWallImages() {
      return wallImages;
    }

    /**
     * Get floor images
     * @return List of Image
     */
    public List<Image> getFloorImages() {
      return floorImages;
    }

    /**
     * Get box images
     * @return List of Image
     */
    public List<Image> getBoxImages() {
      return boxImages;
    }
    /**
     * Get completed box images
     * @return List of Images
     */
    public List<Image> getBoxCompletedImages() {
      return boxCompletedImages;
    }

    /**
     * Get target Images
     * @return List of Image
     */
    public List<Image> getTargetImages() {
      return targetImages;
    }

    /**
     * Get player up images
     * @return List of image
     */
    public List<Image> getPlayerUpImages() {
      return playerUpImages;
    }

    /**
     * Get player down images
     * @return List of image
     */
    public List<Image> getPlayerDownImages() {
      return playerDownImages;
    }

    /**
     * Get player left images
     * @return List of image
     */
    public List<Image> getPlayerLeftImages() {
     return playerLeftImages;
    }

    /**
     * Get player right images
     * @return List of image
     */
    public List<Image> getPlayerRightImages() {
     return playerRightImages;
    }

    /**
     * Constructor
     */
    protected Theme() {
     initItems();
     initPlayer();

    }

    /**
     * Initial Items res
     */
    protected void initItems() {
     URL imgURL = getClass().getResource("R/images/Wall_Brown.png");
     try{
         wallImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/GroundGravel_Dirt.png");
     try{
         floorImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Crate_Beige.png");
     try{
         boxImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Crate_Blue.png");
     try{
         boxCompletedImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/EndPoint_Blue.png");
     try{
         targetImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
    }

    /**
     * Initial player res
     */
    protected void initPlayer() {
     URL imgURL;
     //Player Up
     imgURL = getClass().getResource("R/images/Character7.png");
     try{
         playerUpImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Character8.png");
     try{
         playerUpImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Character7.png");
     try{
         playerUpImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Character9.png");
     try{
         playerUpImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Character7.png");
     try{
         playerUpImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     //Player Down
     imgURL = getClass().getResource("R/images/Character4.png");
     try{
         playerDownImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Character5.png");
     try{
         playerDownImages.add(ImageIO.read(imgURL));
     } catch (IOException e) {
         e.printStackTrace();
     }
     imgURL = getClass().getResource("R/images/Character4.png");
     try{
         playerDownImages.add(ImageIO.read(imgURL));
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character6.png");
        try{
            playerDownImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character4.png");
        try{
            playerDownImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        //Player Left
        imgURL = getClass().getResource("R/images/Character1.png");
        try{
            playerLeftImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character10.png");
        try{
            playerLeftImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character1.png");
        try{
            playerLeftImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character10.png");
        try{
            playerLeftImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character1.png");
        try{
            playerLeftImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        //Player Right
        imgURL = getClass().getResource("R/images/Character2.png");
        try{
            playerRightImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character3.png");
        try{
            playerRightImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character2.png");
        try{
            playerRightImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character3.png");
        try{
            playerRightImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
        imgURL = getClass().getResource("R/images/Character2.png");
        try{
            playerRightImages.add(ImageIO.read(imgURL));
        } catch (IOException e) {
            e.printStackTrace();
        }
      }
  }
```

SnowTheme

```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename NewClass.java
 *  @author Zhonghua Qin
 *  @datetime Dec 2, 2015  6:15:31 PM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.io.IOException;
import java.net.URL;
import javax.imageio.ImageIO;

/**
 * Snow Theme
 * @author Zhonghua Qin
 */
public class SnowTheme extends Theme{

    @Override
    protected void initItems() {
      this.name = "Snow";
      URL imgURL = getClass().getResource("R/images/Wall_Gray.png");
      try{
          wallImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/GroundGravel_Sand.png");
      try{
          floorImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/CrateDark_Brown.png");
      try{
          boxImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/Crate_Red.png");
      try{
          boxCompletedImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/EndPoint_Red.png");
      try{
          targetImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
    }

}
```

CastleTheme

```
/*
 *   Copyright (C) 2015 Zhonghua Qin
 *   @filename CastleTheme.java
 *   @author Zhonghua Qin
 *   @datetime Dec 2, 2015  9:20:04 PM
 *   @version 1.0
 *   @Description
 */
package Zhonghua;

import java.io.IOException;
import java.net.URL;
import javax.imageio.ImageIO;

/**
 * Castle Theme
 * @author Zhonghua Qin
 */
public class CastleTheme extends Theme{
    @Override
    protected void initItems() {
      this.name = "Castle";
      URL imgURL = getClass().getResource("R/images/Wall_Black.png");
      try{
          wallImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/GroundGravel_Concrete.png");
      try{
          floorImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/Crate_Black.png");
      try{
          boxImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/Crate_Purple.png");
      try{
          boxCompletedImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/EndPoint_Purple.png");
      try{
          targetImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
    }
}
```

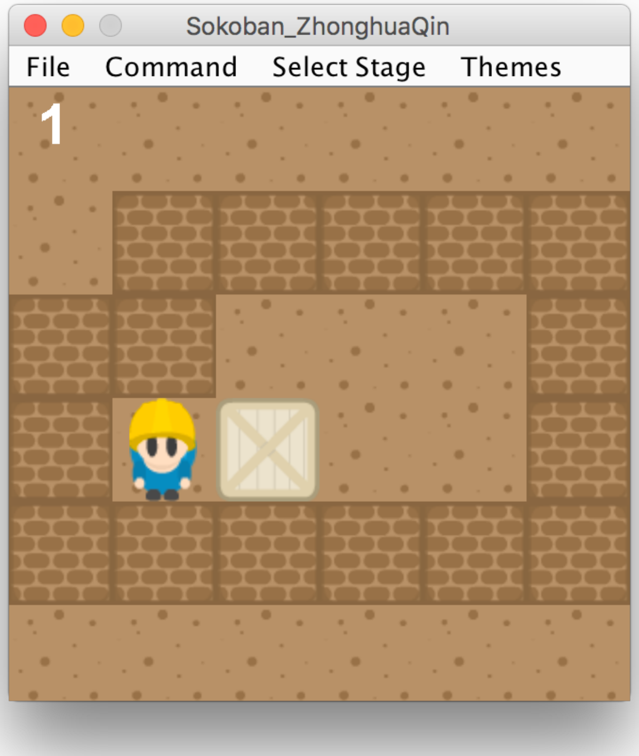
GreenTheme

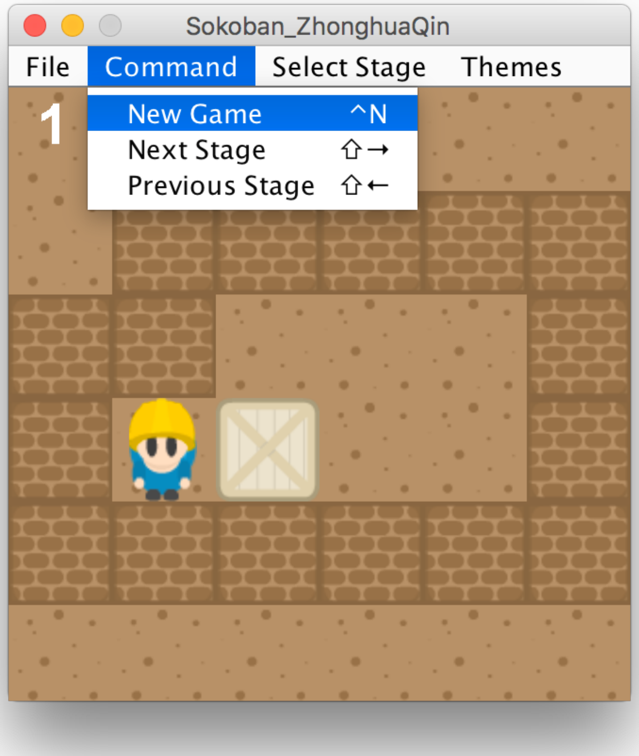
```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename GreenTheme.java
 *  @author Zhonghua Qin
 *  @datetime Dec 2, 2015  9:59:14 PM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.io.IOException;
import java.net.URL;
import javax.imageio.ImageIO;

/**
 * Green Theme
 * @author Zhonghua Qin
 */
public class GreenTheme extends Theme{
    @Override
    protected void initItems() {
      this.name = "Green";
      URL imgURL = getClass().getResource("R/images/Wall_Beige.png");
      try{
          wallImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/GroundGravel_Grass.png");
      try{
          floorImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/Crate_Yellow.png");
      try{
          boxImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/Crate_Gray.png");
      try{
          boxCompletedImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
      imgURL = getClass().getResource("R/images/EndPoint_Gray.png");
      try{
          targetImages.add(ImageIO.read(imgURL));
      } catch (IOException e) {
          e.printStackTrace();
      }
    }
}
```

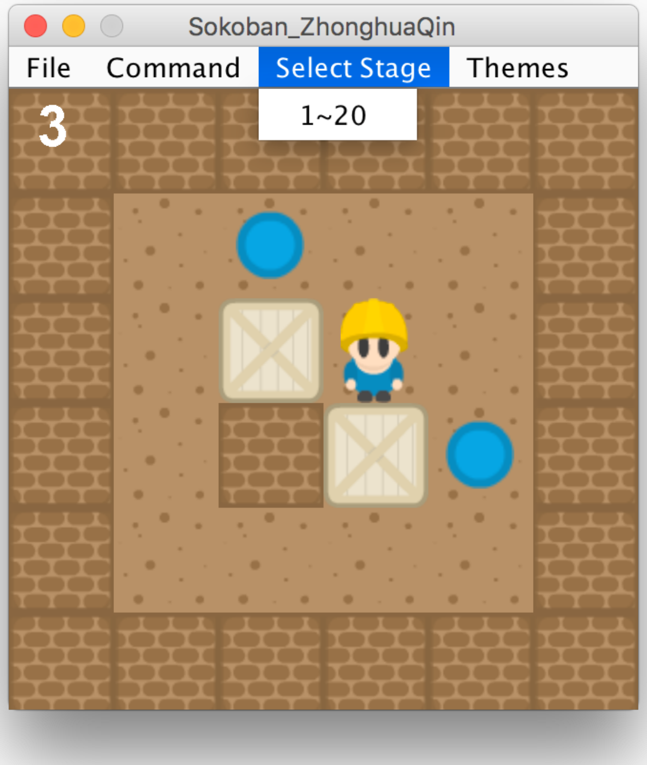
SokobanApplet

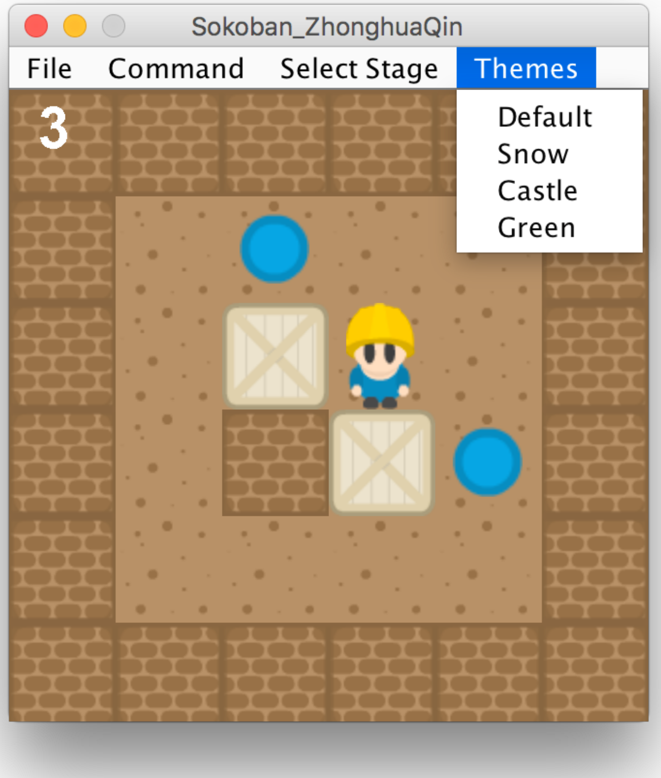
```
/*
 *  Copyright (C) 2015 Zhonghua Qin
 *  @filename SokobanApplet.java
 *  @author Zhonghua Qin
 *  @datetime Dec 1, 2015  5:45:11 PM
 *  @version 1.0
 *  @Description
 */
package Zhonghua;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JApplet;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;

/**
 * Sokoban Applet extends JApplet
 * @author Zhonghua Qin
 */
public class SokobanApplet extends JApplet{

    // Variables declaration - do not modify
    private GamePanel gamePanel;
    // End of variables declaration

    /**
     * Initialization method that will be called after the applet is loaded into the
browser.
     */
    public void init() {
      // TODO start asynchronous download of heavy resources
      add(initMenu(), BorderLayout.NORTH);

      gamePanel = new GamePanel();
      add(gamePanel, BorderLayout.CENTER);

      addKeyListener(new KeyAdapter() {
          @Override
          public void keyPressed(KeyEvent e) {
            gamePanel.keyPressed(e);
          }

      });
      setFocusable(true);

    }

    // TODO overwrite start(), stop() and destroy() methods

    private JMenuBar initMenu(){
      JMenuBar menu = new JMenuBar();

        JMenu menuCommand = new JMenu("Command");
        JMenuItem newGameItem = new JMenuItem("New Game");
      newGameItem.addActionListener((ActionEvent e) -> {
          gamePanel.newGame();
      });
        JMenuItem nextStageItem = new JMenuItem("Next Stage");
      nextStageItem.addActionListener((ActionEvent e) -> {
          gamePanel.nextStage();
      });
        JMenuItem preStageItem = new JMenuItem("Previous Stage");
      preStageItem.addActionListener((ActionEvent e) -> {
          gamePanel.backStage();
      });
      menuCommand.add(newGameItem);
      menuCommand.add(nextStageItem);
      menuCommand.add(preStageItem);

        JMenu menuStage = new JMenu("Stages");
        JMenuItem selectStageItem = new JMenuItem("1~20");
      selectStageItem.addActionListener((ActionEvent e) -> {
          String s = JOptionPane.showInputDialog("Input 
Stage",gamePanel.getController().getCurrentStage()+1);
      System.out.println(s);
      try{
          int n = Integer.parseInt(s);
          System.out.println(n);
          if (n>=1 && n<=20){
            gamePanel.setStage(n);
          }else {
            JOptionPane.showMessageDialog(gamePanel,"The number must in [1,20] range.");
          }
      }catch (NumberFormatException e2){
          JOptionPane.showMessageDialog(gamePanel,"Please input number.");
      }
      });
      menuStage.add(selectStageItem);

      JMenu menuTheme = new JMenu("Themes");
      int numOfThemes = Theme.themes.length;
      ActionListener listener = new ActionListener() {
          @Override
          public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand();
            ((GamePanel)gamePanel).setTheme(Integer.valueOf(command));
          }
      };

      for (int i = 0; i < numOfThemes; i++) {
          JMenuItem item = new JMenuItem(Theme.themes[i].getName());
          item.setActionCommand(String.valueOf(i));
          item.addActionListener(listener);
          menuTheme.add(item);
      }

      menu.add(menuCommand);
      menu.add(menuStage);
      menu.add(menuTheme);

      return menu;
    }

}
```

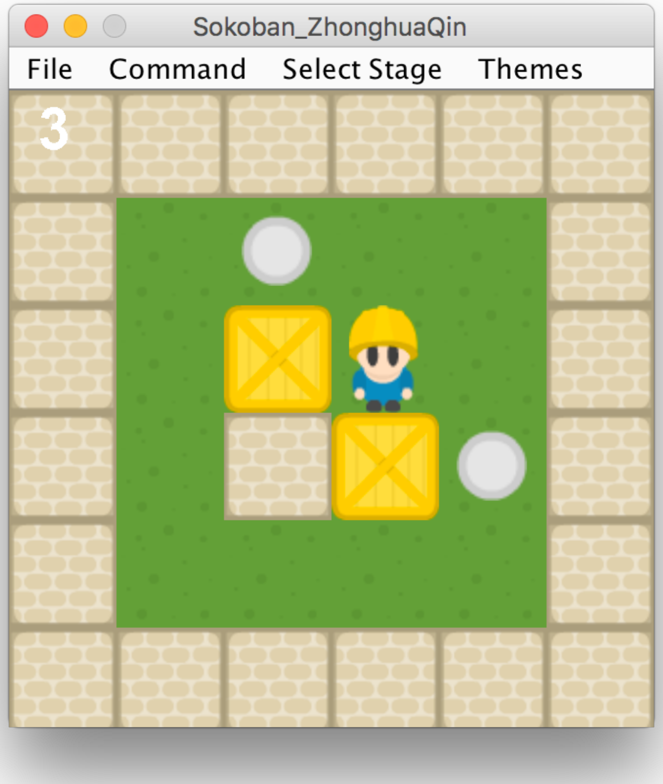
3.Screen Shots

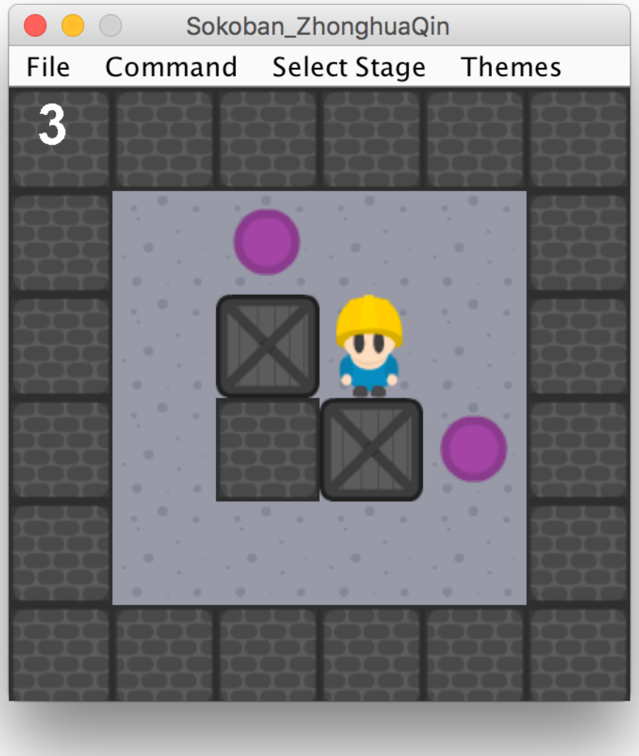


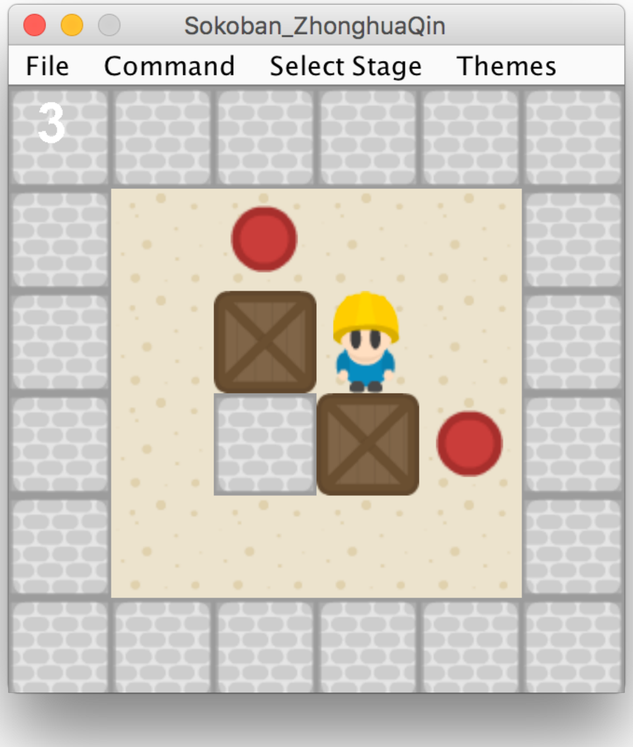


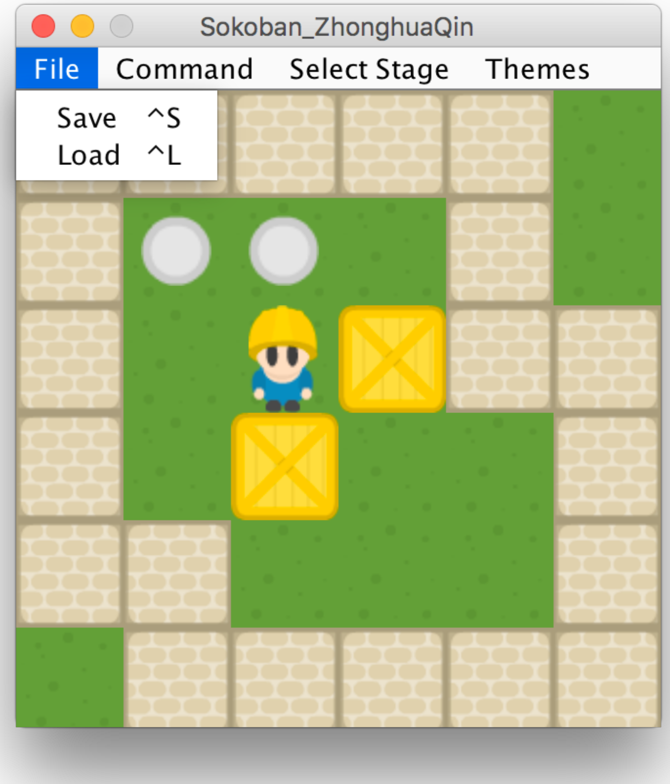


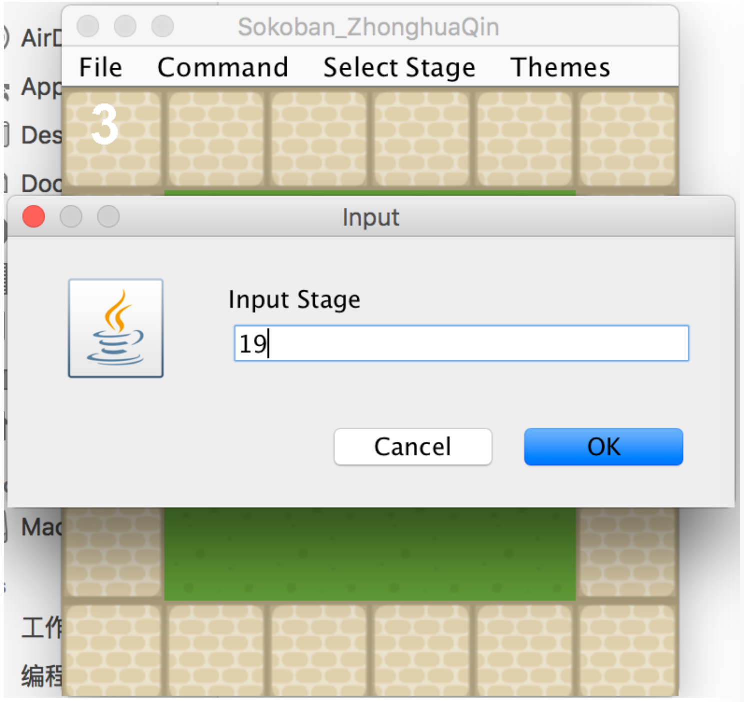



Sokoban_ZhonghuaQin
File
Command
Select Stage
Themes
3


Sokoban_ZhonghuaQin
File
Command
Select Stage
Themes
3

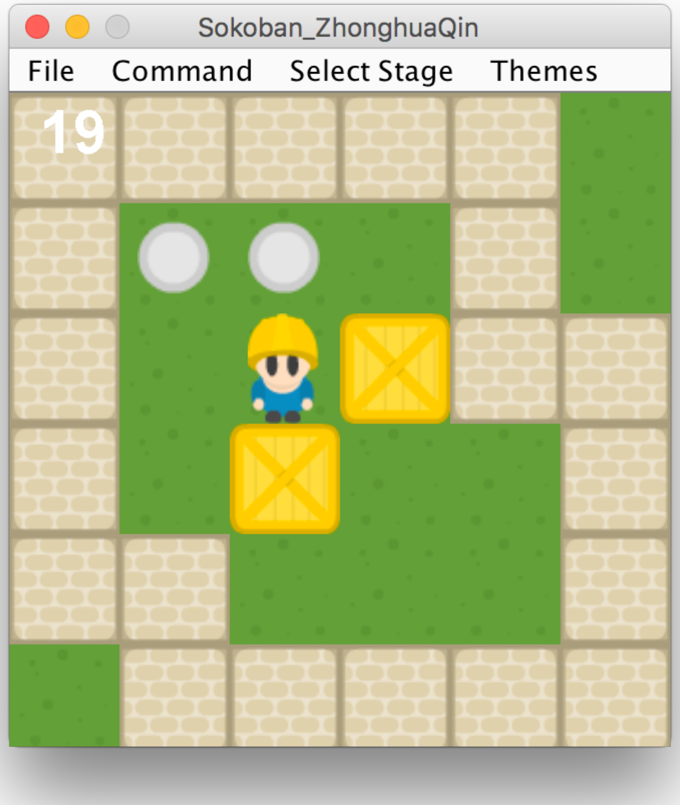

Sokoban_ZhonghuaQin
File
Command
Select Stage
Themes
3

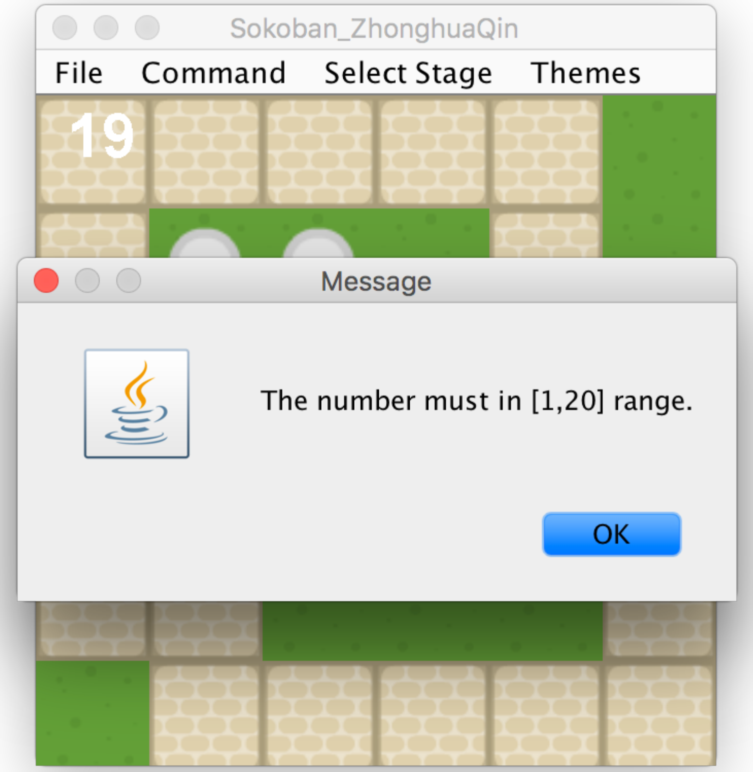

Sokoban_ZhonghuaQin
File
Command
Select Stage
Themes
Save ^S
Load ^L



Input Stage

19





The number must in [1,20] range.