

Event-Driven Programming

1 Introduction

Through this training, the students should master GUI design using JavaFX and event handling.

1.1 Evaluation

- Code Correctness: 60%
- Experimental Report: 40%

1.2 Knowledge Points

- Event Object
- Event Source
- Event Handler
- Inner Class and Anonymous Class
- Lambda Expression for Event Handling

2 Demonstration

2.1 Man Moving

Implement a program simulating man moving.

```
public class TestApplication extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

        Man m = new Man(new Image("file:mario.png"));
        m.setX(150);
        m.setY(150);

        Pane pane = new Pane();
        pane.getChildren().add(m);
        Scene scene = new Scene(pane, 500, 500);
        // Set event handler for keypressed event
        scene.setOnKeyPressed(new MoveHandler(m));

        primaryStage.setScene(scene);
        primaryStage.getIcons().add(new Image("file:mario.png"));
        primaryStage.setTitle("Man Moving");
        primaryStage.show();

    }

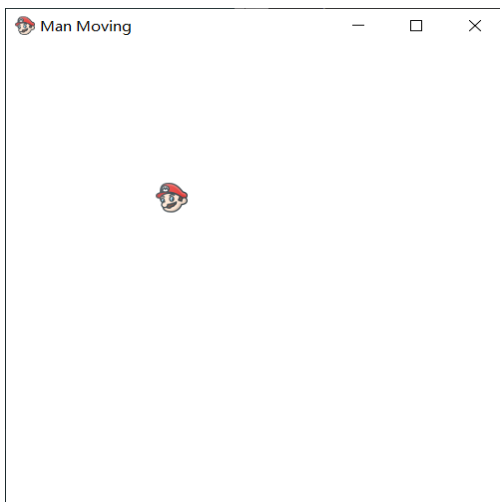
    public static void main(String[] args) {
        launch(args);
    }
}
```

```

    }
}

// The handler class for moving
class MoveHandler implements EventHandler<KeyEvent>{
    private Man m;
    public MoveHandler(Man m) {
        this.m = m;
    }
    @Override
    public void handle(KeyEvent event) {
        switch(event.getCode()) {
            case LEFT: m.setX(m.getX() - 32);break;
            case RIGHT: m.setX(m.getX() + 32);break;
            case UP: m.setY(m.getY() - 32);break;
            case DOWN: m.setY(m.getY() + 32);break;
            default: break;
        }
    }
}
}

```



2.2 Simplify Program Using Anonymous Class

```

public class TestApplication extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

        Man m = new Man(new Image("file:mario.png"));
        m.setX(150);
        m.setY(150);

        Pane pane = new Pane();
        pane.getChildren().add(m);
        Scene scene = new Scene(pane, 500, 500);
        scene.setOnKeyPressed(new EventHandler<KeyEvent>(){

```

```

        @Override
        public void handle(KeyEvent event) {
            switch(event.getCode()) {
                case LEFT: m.setX(m.getX() - 32);break;
                case RIGHT: m.setX(m.getX() + 32);break;
                case UP: m.setY(m.getY() - 32);break;
                case DOWN: m.setY(m.getY() + 32);break;
                default: break;
            }
        }
    });

    primaryStage.setScene(scene);
    primaryStage.getIcons().add(new Image("file:mario.png"));
    primaryStage.setTitle("Man Moving");
    primaryStage.show();
}

public static void main(String[] args) {
    launch(args);
}
}

```

2.2 Simplify Program Using Lambda Expression

```

public class TestApplication extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

        Man m = new Man(new Image("file:mario.png"));
        m.setX(150);
        m.setY(150);

        Pane pane = new Pane();
        pane.getChildren().add(m);
        Scene scene = new Scene(pane, 500, 500);
        scene.setOnKeyPressed(e -> {
            switch(e.getCode()) {
                case LEFT: m.setX(m.getX() - 32);break;
                case RIGHT: m.setX(m.getX() + 32);break;
                case UP: m.setY(m.getY() - 32);break;
                case DOWN: m.setY(m.getY() + 32);break;
                default: break;
            }
        });

        primaryStage.setScene(scene);
        primaryStage.getIcons().add(new Image("file:mario.png"));
        primaryStage.setTitle("Man Moving");
    }
}

```

```

        primaryStage.show();

    }

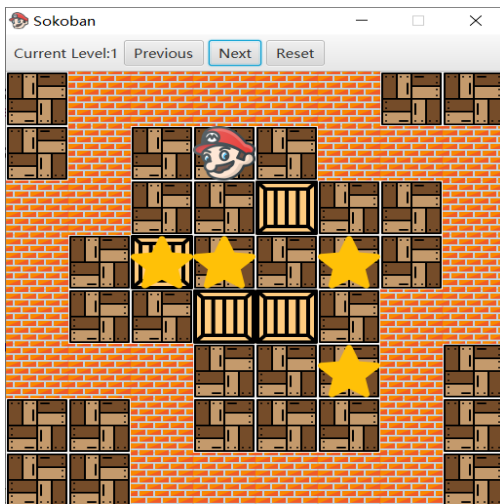
    public static void main(String[] args) {
        launch(args);
    }
}

```

3 Experiment Content

3.1 Sokoban

Sokoban is a game genre in which the player pushes crates or boxes around in a warehouse, trying to get them to storage locations. The game interface is shown as follows:



1. Define image classes `Element`, `MovingElement`, `Man`, `Box`, `Wall` and `Target`.

```

public abstract class Element extends ImageView {
    // Image element types
    public final static int MAN = 0;
    public final static int BOX = 1;
    public final static int WALL = 2;
    public final static int TARGET = 3;
    public final static int BACKGROUND = 4;

    protected Element(Image img) {
        super(img);
        // Set the display size
        this.setFitHeight(MapPane.CELL_SIZE);
        this.setFitWidth(MapPane.CELL_SIZE);
    }
}

// The element can be moved
public class MovingElement extends Element {
    protected MovingElement(Image img) {

```

```

        super(img);
    }
    public void left() {
        this.setX(this.getX() - MapPane.CELL_SIZE);
    }
    public void right() {
        this.setX(this.getX() + MapPane.CELL_SIZE);
    }
    public void up() {
        this.setY(this.getY() - MapPane.CELL_SIZE);
    }
    public void down() {
        this.setY(this.getY() + MapPane.CELL_SIZE);
    }
}

// Two moving element classes
public class Man extends MovingElement {
    public Man(Image manImg) {
        super(manImg);
    }
}
public class Box extends MovingElement {
    public Box(Image img) {
        super(img);
    }
}

// Two fixed element classes
public class Target extends Element {
    public Target(Image img) {
        super(img);
    }
}
public class Wall extends Element {
    public Wall(Image img) {
        super(img);
    }
}
}

```

2. Define the map class `MapPane`.

```

public class MapPane extends Pane {
    // The cell size
    public final static int CELL_SIZE = 64;

    // All images for element displaying
    private Image[] icons;
    // Logical storage space for all elements
    private ImageView[][] map;
    // Controlled object
    private Man man;
}

```

```
// Mission points
private List<ImageView> targets;
// Number of cells
private int xlength;
private int ylength;

// Constructor.
public MapPane(Image[] iconList, File mapFile) {
    icons = iconList;
    // Load map
    this.loadMap(mapFile);

    // Set background
    Background bg = new Background(
        new BackgroundImage(
            iconList[Element.BACKGROUND],
            BackgroundRepeat.REPEAT,
            BackgroundRepeat.REPEAT,
            BackgroundPosition.DEFAULT,
            new BackgroundSize(
                BackgroundSize.AUTO,
                BackgroundSize.AUTO,
                true, true, false, false)
        )
    );
    setBackground(bg);
}

// Clear old data and load new map data
public void loadMap(File mapFile) {
    try(Scanner input = new Scanner(mapFile)){
        // Get length information from the first line in the map file
        String[] items = input.nextLine().split(",");
        int xlen = Integer.parseInt(items[0]);
        int ylen = Integer.parseInt(items[1]);
        xlength = xlen;
        ylength = ylen;

        // Initialize and clear the map pane
        this.map = new ImageView[xlen][ylen];
        targets = new ArrayList<ImageView>();
        this.getChildren().clear();

        // Read file and add elements to the map pane
        while(input.hasNextLine()) {
            // Get information of a element from string parsing
            items = input.nextLine().split(",");
            int x = Integer.parseInt(items[0]);
            int y = Integer.parseInt(items[1]);
            int type = Integer.parseInt(items[2]);

            // Create an element
            Element e = null;
            switch(type) {
```

```

        case Element.MAN: e = this.man = new Man(icons[Element.MAN]);
break;

        case Element.BOX: ...;break;
        case Element.WALL: ...;break;
        case Element.TARGET: ...;break;
        default:break;
    }

    // Set the position and display the element
    e.setX(x * CELL_SIZE);
    e.setY(y * CELL_SIZE);
    getChildren().add(e);
    // Add element to the logical map
    if(e instanceof Target) {
        targets.add(e);
    }else {
        this.map[x][y] = e;
    }

    }
}catch(IOException e) {
    e.printStackTrace();
}
}

// Judge the state of the game
public boolean judge() {
    boolean win = true;
    for(...) {
        int x = (int) (img.getX() / CELL_SIZE);
        int y = (int) (img.getY() / CELL_SIZE);
        if(...)) {
            win = false;
            break;
        }
    }
    return win;
}

// How to move the man and boxes in the upward direction
public void moveManUp(){
    int manX = (int) (man.getX() / CELL_SIZE);
    int manY = (int) (man.getY() / CELL_SIZE);
    if(manY > 0) {
        if(map[manX][manY - 1] == null) {
            man.up();
            map[manX][manY] = null;
            map[manX][manY - 1] = man;
        }else if(map[manX][manY - 1] instanceof Box) {
            System.out.println("hahah");
            if(manY - 1 > 0 && map[manX][manY - 2] == null) {
                Box b = (Box) map[manX][manY - 1];
                b.up();
                map[manX][manY - 2] = b;
            }
        }
    }
}

```

```

        man.up();
        map[manX][manY] = null;
        map[manX][manY - 1] = man;
    }
}

// How to move the man and boxes in the downward direction
public void moveManDown() {
    ...
}
// How to move the man and boxes in the leftward direction
public void moveManLeft(){
    ...
}
// How to move the man and boxes in the rightward direction
public void moveManRight() {
    ...
}
}

```

3.The structure of the map file is as follows:

```

xlength,ylength
x0,y0,elementType1
x1,y1,elementType2
x2,y2,elementType3

```

There is a example:

```

6,6
3,1,0
2,3,1
4,4,2
5,5,3

```

4.Define the main class **Sokoban**:

```

public class Sokoban extends Application {
    // File root paths
    private final String mapDir = "maps";
    private final String imgDir = "imgs";

    // All map files
    private List<File> mapFiles;
    // Current game level

```



```

private int currentLevel = 0;
// Current map
private MapPane currentMap = null;

// Load all map files in the application
public void loadMapFiles() {
    mapFiles = new ArrayList<File>();
    File dir = new File(mapDir);
    for(File f:dir.listFiles()) {
        mapFiles.add(f);
    }
}

public void start(Stage primaryStage) throws Exception {

    loadMapFiles();

    VBox vb = new VBox();

    ToolBar tb = new ToolBar();
    Label label = new Label("Current Level:" + currentLevel);
    // Click button for changing to previous level map
    Button preBtn = new Button("Previous");
    preBtn.setOnAction(e -> {
        if(currentLevel > 0) {
            currentLevel--;
            currentMap.loadMap(mapFiles.get(currentLevel));
            label.setText("Current Level:" + currentLevel);
            primaryStage.sizeToScene();
        }
    });
    // Click button for changing to next level map
    Button nextBtn = new Button("Next");
    nextBtn.setOnAction(e -> {
        ...
    });
    // Click button for resetting this map
    Button resetBtn = new Button("Reset");
    resetBtn.setOnAction(e -> {
        ...
    });
    tb.getItems().addAll(label, preBtn, nextBtn, resetBtn);

    // Add an alert dialog box
    Alert a = new Alert(Alert.AlertType.CONFIRMATION);
    a.setHeaderText("You have won");
    a.setContentText("Next level?");
    a.setOnCloseRequest(e -> {
        if(currentLevel < mapFiles.size() - 1) {
            currentLevel++;
            currentMap.loadMap(mapFiles.get(currentLevel));
            label.setText("Current Level:" + currentLevel);
            primaryStage.sizeToScene();
        }
    });
}

```

```

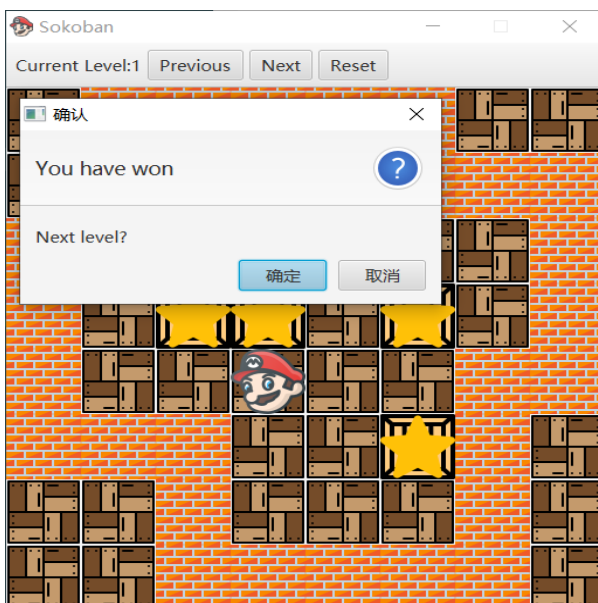
});

// Add a map pane and load the first level map data
Image[] icons = new Image[5];
icons[Element.MAN] = new Image("file:" + imgDir + "/man.png");
icons[Element.BOX] = new Image("file:" + imgDir + "/box.png");
icons[Element.WALL] = new Image("file:" + imgDir + "/wall.png");
icons[Element.TARGET] = new Image("file:" + imgDir + "/target.png");
icons[Element.BACKGROUND] = new Image("file:" + imgDir + "/floor.png");
currentMap = new MapPane(icons, mapFiles.get(currentLevel));

vb.getChildren().addAll(tb, currentMap);
Scene scene = new Scene(vb);
scene.setOnKeyPressed(e -> {
    switch(e.getCode()) {
        case LEFT: currentMap.moveManLeft();break;
        case RIGHT: ...;break;
        case UP: ...;break;
        case DOWN: ...;break;
        default: break;
    }
    if(currentMap.judge()) {
        a.show();
    }
});
primaryStage.sizeToScene();
primaryStage.setResizable(false);
primaryStage.setScene(scene);
primaryStage.getIcons().add(new Image("file:" + imgDir + "/man.png"));
primaryStage.setTitle("Sokoban");
primaryStage.show();
}
}

```

5.Implement and test the program.



4 Experiment Report Requirements

4.1 Think and answer the question

- (1) What else can be improved in the program?
- (2) How to develop a map editing program for this game?

4.2 Experiment report content

- (1) Answer the above questions.
- (2) All codes.

4.3 Submission method

- (1) Upload the report by ftp:(Address:ftp://172.18.5.102; UserName:wangxiaomeng; Password: wangxiaomeng)
- (2) File name format: StudentID+Name. For example, 20191234小明.docx

4.4 Other Instructions

You can obtain experiment course resources through the web platform (URL: <https://www.lanqiao.cn>;
InvitationCode: ZF0XA4Y1)