Теория и технология программирования

Программирование на языке Java

Раздел 4. Разработка редактора

Глухих Михаил Игоревич, к.т.н., доц.

mailto: glukhikh@mail.ru

Задача-пример

- Необходимо разработать редактор схем коммивояжера
- Показывающий города с их названиями
- И существующие пути между ними (вид транспорта, стоимость, время в пути)

Объектная модель

- Город (City)
  - название, координаты
- Путь (Way)
  - начало, конец, стоимость, время
  - тип (kind) - автобус, поезд, самолет
- Мир (World)
  - города и пути

Класс-город

```
public class City implements Serializable {
    private String name;
    private int x, y;
    public City(String name, int x, int y) {
        // …
    }
    public boolean isInside(int x, int y) {
        // …
    }
    public void paint(Graphics g) {
        // …
    }
}
```

Класс-путь

```
public class Way implements Serializable {
    public enum WayKind {
        BUS,
        TRAIN,
        AIRCRAFT;
        // …
    };
    private City start, finish;
    private WayKind kind;
    private int cost, time;
    // …
    public boolean isInside(int x, int y, WayKind kind) {
    }
    public void paint(Graphics g) {
    }
}
```

Класс-мир

```
public class World implements Serializable {
    private Collection<City> cities;
    private Collection<Way> ways;
    public World() {
        cities = new ArrayList<City>();
        ways = new ArrayList<Way>();
    }
    // …
    public City getCityByCoord(int x, int y) {
    }
    public Way getWayByCoord(int x, int y, WayKind kind) {
    }
    public void paint(Graphics g) {
    }
}
```

Что такое Serializable

- Поддержка этого интерфейса обеспечивает возможность сохранения объекта в поток (в виде набора байт) и загрузки его из потока
- Может использоваться
  - для сохранения в файл/загрузки из файла
  - для передачи по сети
  - …

Сохранение/загрузка

- Сохранение мира (world)
  - ObjectOutputStream outputStream =
    new ObjectOutputStream(
    new FileOutputStream(file));
  - outputStream.writeObject(world);
- Загрузка мира (world)
  - ObjectInputStream inputStream =
    new ObjectInputStream(
    new FileInputStream(file));
  - world = (World)inputStream.readObject();

Интерфейс

- Меню: сохранение, загрузка, выход, добавление объектов
- Панель инструментов
- Главная панель для отображения мира
- Малая панель для отображения свойств объекта
- Панель статуса для отображения информационных сообщений

Интерфейс

- См. внешний вид

Иерархия компонентов

- MainFrame
  - JMenuBar menuBar
    - JMenu
      - JMenuItem
      - JRadioButtonMenuItem
  - contentPane (BorderLayout)
    - JToolBar toolbar (NORTH)
      - JButton
    - JPanel statusBar (SOUTH)
    - JSplitPane splitPane (CENTER)
      - JScrollPane scrollPane
        - MainPanel mainPanel
      - JPanel infoPanel

Создание меню

```
    private void initMenuBar() {
        menuBar = new JMenuBar();
        this.setJMenuBar(menuBar);
        fileMenu = new JMenu("Файл");
        menuBar.add(fileMenu);
        openMenu = new JMenuItem("Открыть");
        openMenu.addActionListener(openListener);
        fileMenu.add(openMenu);
        saveMenu = new JMenuItem("Сохранить");
        saveMenu.addActionListener(saveListener);
        fileMenu.add(saveMenu);
        fileMenu.addSeparator();
        // …
    }
```

Создание меню

```
private void initMenuBar() {
    // …
    modeMenu = new JMenu("Режим");
    modeGroup = new ButtonGroup();
    selectMenu = new JRadioButtonMenuItem("Выбор");
    selectMenu.setSelected(true);
    selectMenu.addActionListener(selectListener);
    modeMenu.add(selectMenu);
    modeGroup.add(selectMenu);
    // …
    menuBar.add(modeMenu);
}
```

Создание панели инструментов

- Традиционно, под панель инструментов выделяется северная часть BorderLayout
- Хотя она может сдвигаться и к другим границам

Создание панели инструментов

```
private void initToolBar() {
    toolbar = new JToolBar();
    toolbar.addSeparator();
    openButton = new JButton(new
        ImageIcon("open.png"));
    openButton.addActionListener(openListener);
    toolbar.add(openButton);
    saveButton = new JButton(new
        ImageIcon("save.png"));
    saveButton.addActionListener(saveListener);
    toolbar.add(saveButton);
    // …
    this.add(toolbar, BorderLayout.NORTH);
}
```

Создание слушателей

```
private void initListeners() {
    addCityListener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            onAddCity();
        }
    };
    // …
    quitListener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            onQuit();
        }
    };
}
```

Обработчики изменения режима

```
private void onSelect() {
    statusLabel.setText("Режим выбора");
    mainPanel.chooseSelectMode();
}

private void onAddCity() {
    statusLabel.setText("Режим добавления города");
    mainPanel.chooseCityMode();
}

private void onAddWay() {
    statusLabel.setText("Режим добавления пути");
    mainPanel.chooseWayMode();
}
```

Создание панели статуса

- В библиотеке Swing для панели статуса нет собственного класса
- Поэтому мы используем обычный JPanel
- И разместим на нем JLabel для вывода сообщений

Создание панели статуса

```
private void initStatusBar() {
    statusBar = new JPanel();
    statusBar.setPreferredSize(new Dimension(500, 25));
    statusBar.setBorder(new
        BevelBorder(BevelBorder.LOWERED));
    statusBar.setLayout(new
        FlowLayout(FlowLayout.LEFT));
    statusLabel = new JLabel("Режим выбора");
    statusBar.add(statusLabel);
    this.add(statusBar, BorderLayout.SOUTH);
}
```

Создание главной панели

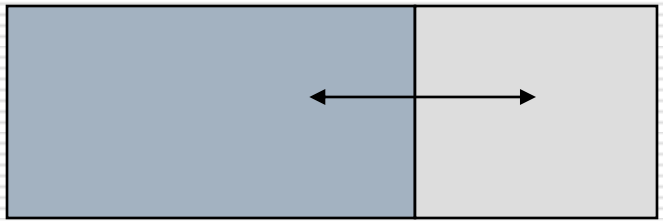
```
private void initMainPanel() {
    mainPanel = new MainPanel();
    mainPanel.setBackground(new Color(0, 0, 64));
    // Задаем размер 1000 x 1000
    mainPanel.setPreferredSize(
        new Dimension(1000, 1000));
    // Задаем «утопленную» рамку
    mainPanel.setBorder(
        new BevelBorder(
            BevelBorder.LOWERED));
}
```

Создание панели прокрутки

- Панель прокрутки используется, чтобы показать большой контейнер на меньшем участке экрана
- ```
  JScrollPane scrollPanel =
      new JScrollPane(mainPanel);
  ```
- ```
  // Минимально допустимый размер
  ```
- ```
  scrollPanel.setMinimumSize(
      new Dimension(200, 200));
  ```
- ```
  // Предпочтительный размер
  ```
- ```
  scrollPanel.setPreferredSize(
      new Dimension(500, 500));
  ```
- ```
  scrollPanel.setVerticalScrollBarPolicy(
      JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
  ```
- ```
  scrollPanel.setHorizontalScrollBarPolicy(
      JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
  ```
- Прокрутка происходит автоматически
  ```

Создание сдвоенной панели

- JSplitPane pane = new JSplitPane(
  JSplitPane.HORIZONTAL_SPLIT,
  scrollPanel, infoPanel);
- Пользователь может регулировать размеры соседних панелей



Создание простого диалога

- Хотим при нажатии на «Выход» уточнить, надо ли выходить
- Можно создать свой JDialog
- Или использовать простой готовый диалог

Диалог подтверждения выхода

```
    private void onQuit() {
        String[] vars = { "Да", "Нет" };
        int result = JOptionPane.showOptionDialog(
          this, "Действительно выйти?",
          "", JOptionPane.YES_NO_OPTION,
          JOptionPane.QUESTION_MESSAGE,
          null, vars, "Да");
        if (result==JOptionPane.YES_OPTION)
          System.exit(0);
    }
```

Подтверждение выхода

- Можно добиться, чтобы тот же запрос появлялся по нажатию X или ALT-F4:

```
setDefaultCloseOperation(
    JFrame.DO_NOTHING_ON_CLOSE);
addWindowListener(
    new WindowAdapter() {
        public void windowClosing
            (WindowEvent ev) {
            onQuit();
        }
    });
```

Диалоги открытия и закрытия файла

- Нет необходимости реализовывать вручную - существует стандартный диалог JFileChooser

Сохранение

```
private void onSave() {
    JFileChooser fileChooser = new JFileChooser(currentFile);
    int result = fileChooser.showSaveDialog(this);
    if (result==JFileChooser.APPROVE_OPTION) {
        currentFile = fileChooser.getSelectedFile();
        try {
            mainPanel.saveWorldToFile(currentFile);
            this.setTitle("Коммивояжер - " +
                currentFile.getName());
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this,
                "Ошибка открытия файла");
        }
    }
}
```

Сохранение

```
public class MainPanel extends JPanel {
    // …
    public void saveWorldToFile(File file)
            throws IOException {
        ObjectOutputStream outputStream =
            new ObjectOutputStream(
                new FileOutputStream(file));
        outputStream.writeObject(world);
    }
}
```

Аналогично, загрузка

```
public class MainPanel extends JPanel {
    // …
    public void openWorldFromFile(File file)
        throws IOException, ClassNotFoundException {
        ObjectInputStream inputStream =
            new ObjectInputStream(
                new FileInputStream(file));
        world = (World)inputStream.readObject();
    }
}
```

Файловые фильтры

```
fileChooser.addChoosableFileFilter(new FileFilter() {
    public boolean accept(File f) {
        if (f!=null) {
            if (f.isDirectory()) return true;
            String name = f.getName();
            int i = name.lastIndexOf('.');
            if (i>0 && i < name.length()-1)
                return
                    name.substring(i+1).equalsIgnoreCase("dat");
        }
        return false;
    }
    public String getDescription() {
        return "Файлы коммивояжера (*.dat)";
    }
});
```

Панель информации

- Должна позволять задать
  - название города
  - тип транспортного маршрута
  - стоимость маршрута
  - время в пути

Необходимые компоненты

```
cityName = new JTextField(15);
String[] kindNames = {
        WayKind.BUS.toString(),
        WayKind.TRAIN.toString(),
        WayKind.AIRCRAFT.toString()
    };
wayKind = new JComboBox(kindNames);
wayCost = new JSpinner(
  new SpinnerNumberModel(
    1000, 100, 10000, 100));
wayTime = new JSpinner(
  new SpinnerNumberModel(
    1, 0, 20, 1));
```

Взаимодействие панелей

- Как связаны между собой главная панель и панель информации?

Взаимодействие панелей

- Главная панель
  - текущий город
  - текущий путь
- Информационная панель
  - имя текущего города
  - тип текущего пути
  - стоимость текущего пути
  - протяженность текущего пути

Взаимодействие панелей

- Панель информации должна «знать» о том, что на главной панели выбран другой город или путь
- С другой стороны, главная панель должна «знать» о том, что на панели информации сменились какие-либо свойства города или пути
- Это значит - панели должны знать друг о друге?

Взаимодействие через интерфейсы

- В главной панели есть **два** основных события:
  - изменение текущего города
  - изменение текущего пути
- В информационной панели есть **четыре** основных события:
  - изменение имени города
  - изменение типа пути
  - изменение стоимости пути
  - изменения продолжительности пути
- Создадим интерфейсы для их обработки

Интерфейс currentListener

```
public interface CurrentListener {
    public void currentCityChanged(City city);
    public void currentWayChanged(Way way);
}

public class MainPanel … {
    CurrentListener currentListener;
}
```

Интерфейс infoListener

```
public interface InfoListener {
    public void cityNameChanged(String name);
    public void wayKindChanged(WayKind kind);
    public void wayCostChanged(int cost);
    public void wayTimeChanged(int time);
}

public class InfoPanel … {
    InfoListener infoListener;
    public void setListener(InfoListener listener) {
        infoListener = listener;
    }
}
```