

Теория и технология программирования

Программирование на языке Java

Раздел 5. Разработка редактора (продолжение)

Глухих Михаил Игоревич, к.т.н., доц.

[mailto: glukhikh@mail.ru](mailto:glukhikh@mail.ru)

Задача-пример

- ❑ Необходимо разработать редактор схем коммивояжера
- ❑ Показывающий города с их названиями
- ❑ И существующие пути между ними

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;  
□     }  
□ }
```

Обработка событий - JTextField

```
□ public void initListeners() {  
□     cityName.addKeyListener(new KeyAdapter() {  
□         public void keyReleased(KeyEvent e) {  
□             if (infoListener != null)  
□                 infoListener.cityNameChanged  
□                     (cityName.getText());  
□         }  
□     });  
□ }
```

Обработка событий - JComboBox

```
□ public void initListeners() {
□     wayKind.addActionListener(new ActionListener() {
□         public void actionPerformed(ActionEvent e) {
□             if (infoListener != null)
□                 infoListener.wayKindChanged(getWayKind());
□         }
□     });
□ }
□ public WayKind getWayKind() {
□     switch (wayKind.getSelectedIndex()) {
□         case 0: return WayKind.BUS;
□         case 1: return WayKind.TRAIN;
□         default: return WayKind.AIRCRAFT;
□     }
□ }
```

Обработка событий - JSpinner

```
□ public void initListeners() {  
□     wayCost.addChangeListener(new ChangeListener() {  
□         public void stateChanged(ChangeEvent e) {  
□             if (infoListener != null)  
□                 infoListener.wayCostChanged(  
□                     (Integer)wayCost.getValue());  
□         }  
□     });  
□ }
```

Обработка событий информационной панели

```
❑ class MainPanel extends JPanel
    implements InfoListener {
❑     public void cityNameChanged(String name) {
❑         if (currentCity != null) {
❑             currentCity.setName(name);
❑             repaint();
❑         }
❑     }
❑     public void wayCostChanged(int cost) {
❑         if (currentWay != null)
❑             currentWay.setCost(cost);
❑     }
❑ }
```

Выбор объектов в главной панели

```
□ private void onPressSelect(int x, int y) {  
□     City city = world.getCityByCoord(x, y);  
□     if (city != null) {  
□         currentCity = city;  
□         currentListener.currentCityChanged(city);  
□         repaint();  
□     } else {  
□         // ...  
□     }  
□ }
```


Обработка событий главной панели

```
□ public class InfoPanel extends JPanel
                                implements CurrentListener {
□     public void currentCityChanged(City city) {
□         if (city==null) {
□             cityName.setEnabled(false);
□         } else {
□             this.setCityName(city.getName());
□         }
□     }
□     public void setCityName(String name) {
□         cityName.setEnabled(true);
□         cityName.setText(name);
□     }
□ }
```

Обработка событий главной панели

```
□ public class InfoPanel extends JPanel
                                implements CurrentListener {
□     public void currentWayChanged(Way way) {
□         if (way == null) {
□             wayKind.setEnabled(false);
□             wayCost.setEnabled(false);
□             wayTime.setEnabled(false);
□         } else {
□             this.setWayKind(way.getKind());
□             this.setWayCost(way.getCost());
□             this.setWayTime(way.getTime());
□         }
□     }
□ }
```

Обработка событий главной панели

```
 public class InfoPanel extends JPanel implements CurrentListener {  
     public void setWayKind(WayKind kind) {  
         wayKind.setEnabled(true);  
         switch (kind) {  
             case BUS:  
                 wayKind.setSelectedIndex(0);  
                 break;  
             case TRAIN:  
                 wayKind.setSelectedIndex(1);  
                 break;  
             case AIRCRAFT:  
                 wayKind.setSelectedIndex(2);  
                 break;  
         }  
     }  
 }
```

Обработка событий главной панели

```
□ public class InfoPanel extends JPanel
                                implements CurrentListener {
□     public void setWayCost(int cost) {
□         wayCost.setEnabled(true);
□         wayCost.setValue(Integer.valueOf(cost));
□     }
□     public void setWayTime(int time) {
□         wayTime.setEnabled(true);
□         wayTime.setValue(Integer.valueOf(time));
□     }
□ }
```

Полноэкранный режим

```
□ private void onFullScreen(boolean on) {  
□     GraphicsEnvironment environment =  
□         GraphicsEnvironment.getLocalGraphicsEnvironment();  
□     GraphicsDevice device =  
□         environment.getDefaultScreenDevice();  
□     if (on) {  
□         //setUndecorated(true);  
□         setResizable(false);  
□         device.setFullScreenWindow(this);  
□     } else {  
□         device.setFullScreenWindow(null);  
□         setResizable(true);  
□     }  
□ }
```

Таблица путей

- Список существующих путей удобно просматривать и редактировать с помощью таблицы
- Для этого существует удобный компонент – `JTable`, демонстрирующий таблицу
- Создадим диалог на его основе – `WaysInfoDialog`

Демонстрация WaysInfoDialog

- См. графический редактор

Заполнение таблицы путей

```
□ private void fillWayTable() {
□     int wayix = 0;
□     for (Way way: world.getWays()) {
□         if (wayix < startWay) {
□             wayix++; continue;
□         }
□         int row = wayix - startWay;
□         if (row >= wayTable.getRowCount()) break;
□         wayTable.setValueAt(way.getStartName(), row, 0);
□         wayTable.setValueAt(way.getFinishName(), row, 1);
□         // ...
□         wayix++;
□     }
□     jShownLabel.setText("Показаны пути с "+(startWay+1)+
□     " по "+wayix+" из существующих "+world.getWays().size());
□ }
```


Обработчики нажатия клавиш

```
□ private void onPrev(java.awt.event.ActionEvent evt) {  
□     if (startWay==0)  
□         return;  
□     startWay -= wayTable.getRowCount();  
□     if (startWay < 0)  
□         startWay = 0;  
□     fillWayTable();  
□ }  
□ private void onNext(java.awt.event.ActionEvent evt) {  
□     if (startWay+wayTable.getRowCount() >=  
□         world.getWays().size())  
□         return;  
□     startWay += wayTable.getRowCount();  
□     fillWayTable();  
□ }
```

Способы сохранения/загрузки

- ❑ Способ 1 (самый простой в реализации) – поддержка интерфейса Serializable
- ❑ Обратите внимание – "самый простой в реализации" не означает "самый простой в поддержке"!
- ❑ Достаточно добавить в класс данное-член, или добавить не закрытую функцию, как файлы, сформированные старой версией, перестанут загружаться

Почему возникают проблемы при восстановлении?

- ❑ При сохранении объекта записывается также `SerialVersionUID`
- ❑ Он формируется из всех данных (кроме тех, которые помечены как `transient`), имен функций (кроме `private`), статических членов (кроме `private`)
- ❑ Если мы меняем класс, `SerialVersionUID` меняется

Демонстрация восстановления

□ См. пример

Вывод по Serializable

- Если вы решаете, что некоторый объект реализует этот интерфейс, тем самым вы соглашаетесь более не добавлять в него членов-данных (кроме `transient`) и методов (кроме `private`)

Способы сохранения/загрузки

- ❑ Способ 2 (несколько более сложный) – поддержка интерфейса `Externalizable`
- ❑ Необходимо реализовать два метода
 - `public void writeExternal(ObjectOutput out) throws IOException {}`
 - `public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {}`
- ❑ В первом из них мы определяем, как записать объект в поток
- ❑ А во втором – как прочитать объект из потока

Пример (для класса City) writeExternal/readExternal

- ❑ `public void writeExternal(ObjectOutput out) throws IOException {`
 - ❑ `out.writeInt(1); // версия`
 - ❑ `out.writeUTF(name);`
 - ❑ `out.writeInt(x);`
 - ❑ `out.writeInt(y);`
 - ❑ `}`
 - ❑ `public void readExternal(ObjectInput in)`
`throws IOException, ClassNotFoundException {`
 - ❑ `final int version = in.readInt();`
 - ❑ `if (version > 1) throw new IOException("...");`
`name = in.readUTF();`
 - ❑ `x = in.readInt();`
 - ❑ `y = in.readInt();`
 - ❑ `}`
-

Пример (для класса World) writeExternal

```
□ public void writeExternal(ObjectOutput out) throws IOException {
□     out.writeInt(1);
□     out.writeInt(cities.size()); // Города
□     for (City city: cities) city.writeExternal(out);
□     out.writeInt(ways.size()); // Пути
□     for (Way way: ways) {
□         way.writeExternal(out);
□         final int startIndex = cities.indexOf(way.getStart());
□         out.writeInt(startIndex);
□         final int finishIndex = cities.indexOf(way.getFinish());
□         out.writeInt(finishIndex);
□     }
□ }
```


Пример (для класса World) readExternal

```
□ public void readExternal(ObjectInput in)
    throws IOException, ClassNotFoundException {
□     final int version = in.readInt();
□     if (version > 1) throw new IOException("...");
□     cities.clear();
□     final int cityNum = in.readInt();
□     for (int i=0; i<cityNum; i++) {
□         final City city = new City("", 0, 0);
□         city.readExternal(in);
□         cities.add(city);
□     }
□     // ...
□ }
```

Пример (для класса World) readExternal

```
□ public void readExternal(ObjectInput in)
    throws IOException, ClassNotFoundException {
□     // ...
□     ways.clear();
□     final int wayNum = in.readInt();
□     for (int i=0; i<wayNum; i++) {
□         final Way way = new Way(null, null, WayKind.BUS, 0, 0);
□         way.readExternal(in);
□         way.setStart(cities.get(in.readInt()));
□         way.setFinish(cities.get(in.readInt()));
□         ways.add(way);
□     }
□ }
```

Достоинства writeExternal/readExternal

- ❑ Мы сами определяем, как сохранять наши объекты
- ❑ При помощи поля "версия" мы можем следить за изменениями версии объекта (обеспечивается обратная совместимость, то есть более новые версии программы могут прочитать файлы, созданные старыми версиями)

Недостатки writeExternal/readExternal

- ❑ Содержимое сохраненного файла (см. пример) абсолютно непонятно человеку; если произойдет какая-либо ошибка, понять, что произошло, достаточно сложно

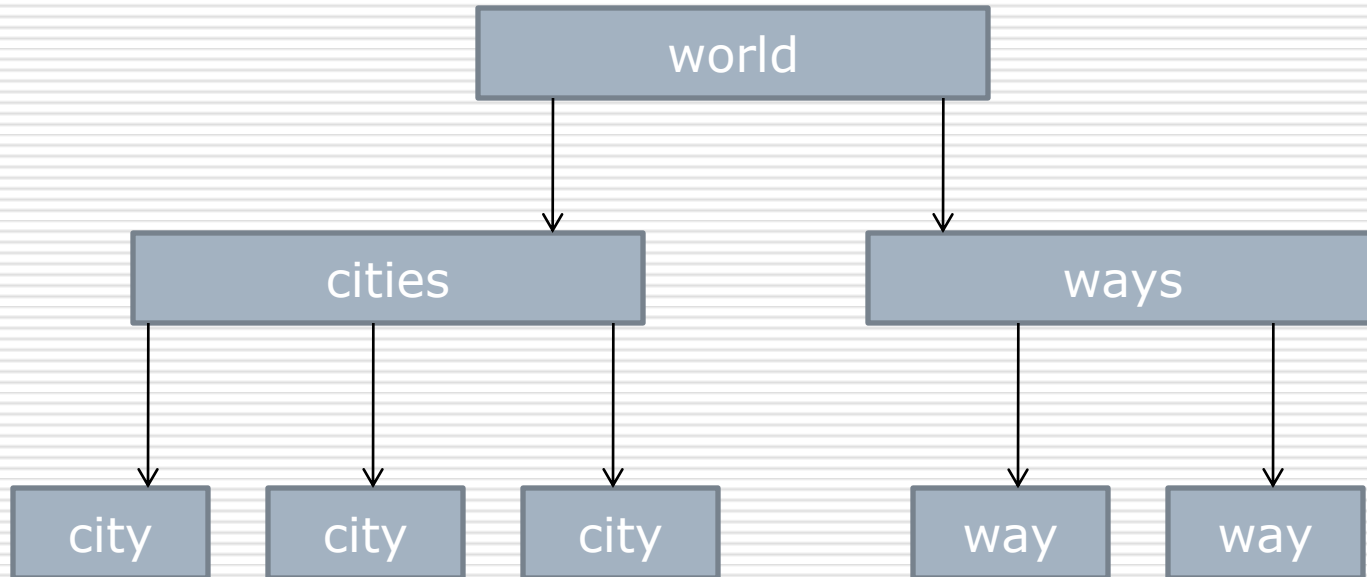
Способы сохранения/загрузки

- ❑ Способ 3 – использование универсальных форматов
- ❑ Примеры таких форматов – XML, JSON, INI
- ❑ Общее их свойство – все они текстовые, то есть читаемые

Пример файла в формате XML (eXtensible Markup Language)

- ❑ `<?xml version="1.0" encoding="UTF-8" ?>`
- ❑ `<world>`
- ❑ `<cities>`
- ❑ `<city name="Питер" x="48" y="98" />`
- ❑ `<city name="Москва" x="286" y="70" />`
- ❑ `<city name="Киев" x="196" y="267" />`
- ❑ `</cities><ways>`
- ❑ `<way kind="Автобус" time="10" cost="500" start="0" finish="1"/>`
- ❑ `<way kind="Поезд" time="5" cost="2000" start="0" finish="1"/>`
- ❑ `<way kind="Самолет" time="2" cost="4600" start="0" finish="1"/>`
- ❑ `<way kind="Поезд" time="4" cost="2100" start="1" finish="2"/>`
- ❑ `<way kind="Самолет" time="4" cost="6000" start="0" finish="2"/>`
- ❑ `</ways></world>`

Структура XML



Как работать с XML на Java?

- Есть много способов. Один из них – открытая библиотека JDOM (см. www.jdom.org)
- Основные используемые понятия
 - `org.jdom.Document` – документ XML
 - `org.jdom.Element` – один из элементов документа, например, `<city>...</city>` или `<world>...</world>`
 - `org.jdom.Attribute` – атрибут одного из элементов, например, `cost`

Создание элемента для города

```
❑ public Element getElement() {  
❑     final Element root = new Element("city");  
❑     root.setAttribute("name", name);  
❑     root.setAttribute("x", String.valueOf(x));  
❑     root.setAttribute("y", String.valueOf(y));  
❑     return root;  
❑ }
```

Сохранение в XML

```
❑ public void writeXML(final File file) throws IOException {  
❑     final Element citiesRoot = new Element("cities");  
❑     final List<Element> cityElements = new LinkedList<Element>();  
❑     for (City city : cities)  
❑         cityElements.add(city.getXMLElement());  
❑     citiesRoot.setContent(cityElements);  
❑ }
```

Сохранение в XML

```
❑ public void writeXML(final File file) throws IOException {
❑     final Element waysRoot = new Element("ways");
❑     final List<Element> wayElements = new LinkedList<Element>();
❑     for (Way way: ways) {
❑         final Element wayElement = way.getXMLElement();
❑         final int startIndex = cities.indexOf(way.getStart());
❑         wayElement.setAttribute("start", String.valueOf(startIndex));
❑         final int finishIndex = cities.indexOf(way.getFinish());
❑         wayElement.setAttribute("finish", String.valueOf(finishIndex));
❑         wayElements.add(wayElement);
❑     }
❑     waysRoot.setContent(wayElements);
❑ }
```

Сохранение в XML

```
❑ public void writeXML(final File file) throws IOException {  
❑     final Element root = new Element("world");  
❑     // ...  
❑     root.addContent(citiesRoot);  
❑     root.addContent(waysRoot);  
❑     final Document document = new Document(root);  
❑     final XMLOutputter outputter = new XMLOutputter();  
❑     outputter.output(document, new FileOutputStream(file));  
❑ }
```

Разбор элемента для города

```
□ public static City readXML(Element element) {  
□     return new City(  
        element.getAttributeValue("name"),  
        Integer.parseInt(  
            element.getAttributeValue("x")),  
        Integer.parseInt(  
            element.getAttributeValue("y")));  
□ }
```

Загрузка из XML

```
□ public void readXML(final File file)
    throws IOException, JDOMException {
□     // SAX – Simple API for XML parsing
□     final SAXBuilder builder = new SAXBuilder();
□     final Document document = builder.build(file);
□     final Element root = document.getRootElement();
□     cities.clear();
□     final Element citiesRoot = root.getChild("cities");
□     for (Object obj: citiesRoot.getChildren()) {
□         cities.add(City.readXML((Element)obj));
□     }
□     // ...
□ }
```

Загрузка из XML

```
❑ public void readXML(final File file)
    throws IOException, JDOMException {
❑     ways.clear();
❑     final Element waysRoot = root.getChild("ways");
❑     for (Object obj: waysRoot.getChildren()) {
❑         final Element wayElem = (Element)obj;
❑         final Way way = Way.readXML(wayElem);
❑         way.setStart(cities.get(
            Integer.parseInt(wayElem.getAttributeValue("start"))));
❑         way.setFinish(cities.get(
            Integer.parseInt(wayElem.getAttributeValue("finish"))));
❑         ways.add(way);
❑     }
❑ }
```
