Павел Степанов

Кафедра компьютерной математики и программирования СПб ГУАП

Язык программирования Java

Тема 5 I/O

- Файловый ввод-вывод
- Сетевой ввод-вывод
- Защищенные сетевые соединения
- Nio2
- Сериализация
- JDBC

5.1 Концепция потокового IO

- Бинарные потоки
  - InputStream
  - OutputStream
- Текстовые потоки
  - Reader
    - System.in
  - Writer
    - System.out
    - System.err

5.2 Бинарные потоки

- Наследники InputStream и OutputStrem
  - FileInputStream
  - FileOutputStream
- Небуферизованный обмен

5.3 Символьные потоки

- Наследники Reader и Writer
  - FileReader
  - FileWriter
- Небуферизованный обмен

5.3 Бефуризованный ввод и вывод

- BufferedInputStream
- BufferedOutputStream
- BufferedReader
- BufferedWriter
- “матрешка” из потоков
- PrintWriter – уже все умеет

5.4 Стандартные потоки

- System.in
- System.out
- System.err

5.5 Сетевое взаимодействие

- TCP
  - URL
  - URLConnection
  - Socket
  - ServerSocket
- UDP
  - DatagramPacket
  - DatagramSocket
  - MulticastSocket

5.6 URL и URLConnection

- Представляет ресурс в сети
- Особенно хорошо подходит для чтения веб-страниц
- Можно читать напрямую или с использованием URLConnection
- URLConnection myURLConnection = myURL.openConnection(); myURLConnection.connect();
  - Дальше можно получить InputStream и OutputStream

5.7 Пример чтения URL

```
public class URLSample {

    public static void readWeb(String url) throws IOException {
        URL myURL = new URL(url);
        try (
                InputStreamReader reader = new InputStreamReader(myURL.openStream());
                BufferedReader in = new BufferedReader(reader)) {
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                System.out.println(inputLine);
            }
        }
    }

    public static void main(String... args) {
        try {
            readWeb("http://www.guap.ru");
        } catch (IOException ex) {
            Logger.getLogger(URLSample.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

5.8 URLConnection

- Типовое применение - соединение по протоколу http
- Может выполнять отправку данных на сервер (“post”)

5.9 Сокеты

- Сокет – это абстракция
- Сокет – это ресурс операционной системы
- Сокет – это пара из имени машины и номера порта
- Сокет – конечная точка сетевой коммуникации

5.10 Чтение и запись в сокеты

- У сокета есть InputStream и OutputStream
- Передача данных по сети ничем не отличается от записи в файл
- Блокировка потока на обмене через сокет
  - Если надо срочно остановить поток, заблокированный на чтении из сокета, то сокет можно асинхронно закрыть из другого потока (это хак)
- Важно не забыть по окончании взаимодействия сперва закрыть все потоки, а потом и сокет

5.11 Как правильно работать с ServerSocket

```
public class SocketProcessor implements Runnable{
    final Socket mySocket;

    public SocketProcessor(Socket socket){
        mySocket=socket;
    }

    public void run(){
        // code
    }
}

public void openServerSocket(int port) throws IOException{
    ServerSocket serverSocket = new ServerSocket(port);

    while (true){
        Socket socket = serverSocket.accept();
        new Thread(new SocketProcessor(socket)).start();
    }

}
```

5.12 Датаграммы

- Идея та же, но доставка сообщений не гарантируется
- Серверный сокет – один,
  - создается с указанием порта
    - new DatagramSocket(порт);
  - В бесконечном цикле получает и отправляет датаграммные пакеты методами send и receive
- Клиентский сокет
  - Создается без указания порта
- Пакеты
  - Содержат в себе имя и порт адресата

5.13 Широковещание

- MulticastSocket – слушает широковещательные сообщения
- Понятие широковещательного адреса
  - Инверсия маски подсети
  - Подсеть 192.168.0.0 и маска 255.255.255.0 дают адрес 192.168.0.255

5.14 Параметры сети

- NetworkInterface
- NetworkInterface.getNetworkInterfaces()

5.15 SSL соединения

- SSL/TLS
- Аутентификация
- Integrity
- Шифрование
- Нужен certificate и keystore
- SSLSocket
- SSLServerSocket
- SSLServerSocketFactory
- SSLEngines
- SSLContext

5.16 Nio2 file I/O

- Path
- Paths
- Files
- Поддерживает все основные функции файловой системы

5.17 nio2 channels

- Более производительны
- Поддерживают неблокирующий IO
- Поддерживают асинхронные операции
- Многопоточность
- Channels

5.18 Сериализация

- Serializable
- ObjectInputStream & ObjectOutputStream
  - readObject & writeObject
- Конструкторы и transient
- Код класса не сериализуется!
- serialVersionUID

5.19 Нестандартная сериализация

- private void writeObject(ObjectOutputStream stream) throws IOException;
- private void readObject(ObjectInputStream stream) throws IOException, ClassNotFoundException;

5.20 JDBC

- Интерфейс
  - Connection
  - Statement
  - ResultSet
- Драйвер

5.21 Никогда так не делайте

```
public void executeUpdate(Connection con, int x, int y)
        throws SQLException {

    PreparedStatement myStatement = null;

    String updateString =
            "update IMPORTANT_TABLE "
            + "set X = " + x + " where Y = " + y;

    con.setAutoCommit(false);
    myStatement = con.prepareStatement(updateString);
    myStatement.executeQuery();
    con.commit();

}
```

5.22 Делайте так

```
public void executeUpdate(Connection con, int x, int y)
        throws SQLException {

    PreparedStatement myStatement = null;

    String updateString =
            "update IMPORTANT_TABLE "
            + "set X = ? where Y = ?";



    con.setAutoCommit(false);
    myStatement = con.prepareStatement(updateString);
    myStatement.setInt(1, x);
    myStatement.setInt(2, y);
    myStatement.executeQuery();
    con.commit();

}
```

5.23 А еще лучше – вот так

```
PreparedStatement myStatement = null;

public void executeUpdate(int x, int y)
        throws SQLException {

    if (myStatement == null) {

        String updateString =
                "update IMPORTANT_TABLE "
                + "set X = ? where Y = ?";

        myStatement = con.prepareStatement(updateString);
    }
    myStatement.setInt(1, x);
    myStatement.setInt(2, y);
    myStatement.executeQuery();
}
```

5.24 Q & A