Семинар №5

Задачи семинара

1. Вопросы и обсуждение семинара №4
2. Исключения, ключевые слова try, catch, finally, throw, throws
3. Иерархия исключений: Throwable, Exception, RuntimeException и Error
4. Понятие потока. Корневые классы иерархии потоков: InputStream, OutputStream, Reader, Writer. Построение цепочки потоков.
5. Работа с файловой системой. Классы File, FileInputStream, FileOutputStream, FileReader, FileWriter, RandomAccessFile, BufferedInputStream, BufferedOutputStream
6. Специальные потоковые классы: ByteArrayXXX, StringXXX, DataXXX
7. Классы для связи потоков в разных форматах OuputStreamWriter, InputStreamReader

Материалы

Пример корректного использования ресурсов (потоков, соединений с базой данных, сокетов, …)

```
SomeReosurce resource = null;
try {
        resource = new SomeResource();
        // custom code where resource is used
} catch (SomeException e) {
//do something
}
finally {
        if( resource!=null){
                resource.close();
        }
}
```

Быстрое копирование между потоками

```
BufferedInputStream in = ….
BufferedOutputStream out = ….
byte buf[] = new byte[4096];
int count;
while((count = in.read(buf))>=0){
        out.write(buf, 0, count);
}
```

Работа с серверным сокетом для приема соединений от клиента

```
ServerSocket s = new ServerSocket(8080);
while(true){
        Socket clientS = s.accept();
       new Thread(new Client(clientS), "client thread").start();
}
```

Java — задания

Генерация исключений
Написать класс, реализующий интерфейс:

```
public interface ExceptionGenerator {
	void generateNullPointerException();
	void generateClassCastException();
	void generateNumberFormatException();
	void generateStackOverflowError();
	void generateOutOfMemoryError();
	void generateMyException(String message) throws MyException;
}
```

Методы класса должны выбрасывать соответствующие исключения (по имени метода).

Для первых 5-и методов нельзя использовать директиву **throw**. Надо написать такой код, который будет сам генерировать исключение.

В последнем методе использовать **throw** для выбрасывания собственного исключения MyException, который содержит message, заданный при вызове метода. Класс MyException написать самим.

Добавить метод main(), в котором тестируются все методы класса. Каждый вызов оборачивается try/catch, и stack trace исключения выводится на экран с помощью функции printStackTrace().

Работа с файлами
Написать генератор index.html файла. То есть для заданной директории создать HTML файл, который содержит кликабельный список всех директорий и файлов внутри заданной директории. При этом:

- В начале идёт директория .. которая указывает на верх дерева файловой системы.
- Затем идёт список директорий отсортированный по именам
- Потом список файлов отсортированный по именам
- Кроме того показывать информацию о дате модификации и размере (для файлов)

Дополнительно - простейший HTTP сервер

Используя результат предыдущего задания написать простейший HTTP сервер, который будет принимать в качестве аргументов порт и директорию доступ до которой надо дать по HTTP. Реализовать две команды:

- GET — получить ресурс
- HEAD — получить только заголовки от GET без самого ресурса

Если запрошенная директория содержит файл index.html сервер возвращает его, иначе использует результат предыдущего задания для создания листинга директории. Реализовать обработку ошибок:

- Если ресурс отсутствует - возвращаем 404
- Если произошло исключение - 500 и текст ошибки
- Неизвестная команда - 501 Not Implemented

Пример работы с HTTP клиентом:

```
//читаем первую строку запроса, игнорируем все заголовки которые идут дальше первой строки
StringBuilder sb = new StringBuilder();
int c;
while((c =in.read())!=-1 && c!=10 && c!=13){
    sb.append((char)c);
}

//получаем команду и ее аргументы
String data = sb.toString();
String args[] = data.split(" ");
String cmd = args[0].trim().toUpperCase();

// пишем ответ  Hello world
String s = "<html><title>test</title><body>Hello <b>world</b></body></html>";
//пишем статус ответа
out.write("HTTP/1.0 200 OK\r\n".getBytes());
//минимально необходимые заголовки, тип и длина
out.write("Content-Type: text/html\r\n".getBytes());
out.write(("Content-Length: "+s.length()+"\r\n").getBytes());
//пустая строка отделяет заголовки от тела
out.write("\r\n".getBytes());
//тело
out.write(s.getBytes());
out.flush();
```

Для определения Content-Type (MIME типа) файла можно использовать следующий класс: javax.activation.MimetypesFileTypeMap. Пример:

```
File f = new File("1.gif");
System.out.println("Mime Type of " + f.getName() + " is " +
              new MimetypesFileTypeMap().getContentType(f));
```

Дополнительно можно передовать заголовок Last-Modified, где Last-Modified брать из атрибутов файла. Пример форматирования даты: Thu Jul 15 17:40:27 2010 GMT