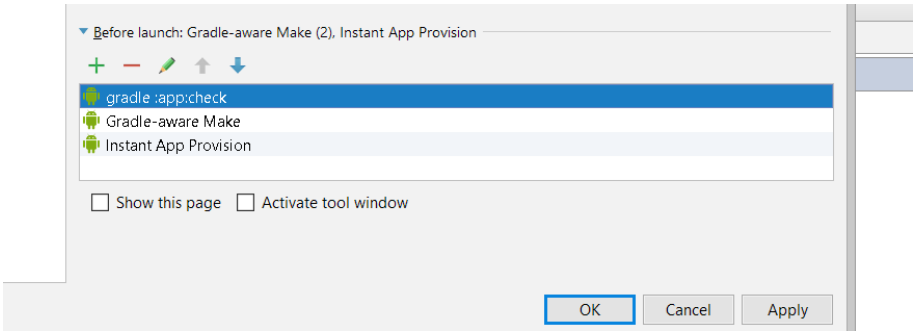
Java NIO

Java IO

Tricks

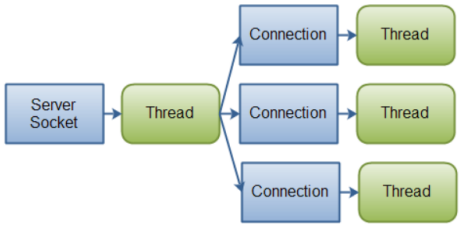
- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- ВНИМАНИЕ. Git password request постоянно запрашивает пароль
    - просто переустановить GIT отсюда или скачать, заодно удалить credentials
    - credentials автоматом создаются при вводе login pass для git push
- ВНИМАНИЕ. Run multiple compound
- ВНИМАНИЕ. Regex punctuation
- ВНИМАНИЕ. Android Lint Check СУЩЕСТВЕННО ЗАМЕДЛЯЕТ КОМПИЛЯЦИЮ
    - подключение Lint добавить в Gradle app:check в Edit Configurations
    - Lint checking изменить Mission Translation на Warning
    - изменить build.gradle или прямо в strings.xml
- 

- Пример. реализация три варианта отключения проверки на translation build.gradle проверено
- ```
  android {
      compileSdkVersion 27
      buildToolsVersion '27.0.3'
      defaultConfig {
          applicationId "br.com.marangoni.android.faced"
          minSdkVersion 23
          targetSdkVersion 27
          versionCode 20
          versionName '1.0.20'
          testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
          lintOptions {
              disable 'MissingTranslation'
          }
      }
  ```
- ```
  <?xml version="1.0" encoding="utf-8"?>
  <resources
      xmlns:tools="http://schemas.android.com/tools"
      tools:ignore="MissingTranslation"  >
          <string name="pref_user_id">user_id</string>
      <string name="pref_user_name">user_name</string>
      <string name="pref_user_cpf">cpf</string>
  </resources>
  ```
- ```
  <?xml version="1.0" encoding="utf-8"?>
  <resources>
      <string name="pref_user_id" translatable="false">user_id</string>
      <string name="pref_user_name" translatable="false">user_name</string>
      <string name="pref_user_cpf">cpf</string>
      <string name="pref_user_email">email</string>
      <string name="pref_user_phone_code">phone_code</string>
  </resources>
  ```
  ```

- ВНИМАНИЕ. ExecutorService применение методов завершения
    - shutdown() стандартное завершение
    - awaitTermination() ожидание завершения заданное время
- ВНИМАНИЕ. Imported JAR Library error message < package DOES NOT EXISTS >
    - если после импорта библиотеки не видит package значит в JAR просто НЕТ Class файлов
    - создать свой проект, скопировать туда исходники и создать свой JAR с class фалами
- Пример. реализация FaceppSDK project
- ВНИМАНИЕ. J2EE Glassfish Server выдает NullPointerException надо задать Java8 SDK
- ВНИМАНИЕ. Files.list() создает поток ЗАКРЫТЬ ОБЯЗАТЕЛЬНО блокирует каталог **java8_nio/nio2/Main03F**
    - смотреть Moving Files раздел
- ВНИМАНИЕ. Files.newDirectoryStream() поток ЗАКРЫТЬ ОБЯЗАТЕЛЬНО блокирует **java8_nio/nio2/Main03F**
    - смотреть Moving Files раздел

Java IO Systems

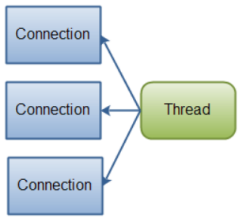
- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java IO Systems системы ввода вывода в Java java8_nio
  - Java IO стандартная система ввода вывода IO на базе потоков
  - Java NIO новая система ввода вывода на NIO на базе буферов
  - справка по заменам deprecated классов и методов
- Java IO стандартная система ввода вывода IO на базе потоков java8_nio
  - данные доступны байт за байтом, при чтении потока от начала до конца
  - перемещаться по данным нельзя, можно перечитать поток заново
  - если данных нет, поток блокируется, пока данные не будут доступны



- **Java IO: A classic IO server design - one connection handled by one thread.**
- Java IO Недостатки
  - FIle нет методов копирования или обработки каталогов
  - методы выдают boolean вместо того, чтобы выбросить Exception
  - методы работают с малым числом атрибутов
  - IOStreamнеправильно работают с символьными строками

Java NIO

- Java NIO новая система ввода вывода на NIO на базе буферов
  - данны доступны в буфере в любом порядке чтения
  - если данных нет, поток не блокируется, в буфер читается то, что есть
  - Channel канал данных, который можно пристегнуть к потоку
  - Selector селектор, позволяет выбрать канал для чтения или записи



- **Java NIO: A single thread managing multiple connections.**

Java NIO1

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- обновление информации по селекторам прерываемым и асинхронным каналам

Java NIO2

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO2 новые возможности
  - каналы добавлены каналы и селекторы, которые не блокируются
  - буферы добавлена буферизация всех классов обмена данными
  - кодировки добавлены кодеры и декодеры для обработки символов

Java IO Classes

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- IO Interfaces интерфейсы
    - DataInput, ObjectInput, FileFilter, FilenameFilter, Flushable, Closeable, Readable
- IO File работа с файлами
    - File, FileDescriptor, FilePermission, SerializablePermission, RandomAccessFile
- IO Stream потоки работы с байт данными
    - AudioInputStream, ByteArrayInputStream,FileInputStream, ObjectInputStream,
    - SequenceInputStream, PipedInputStream, FilterInputStream, PrintStream
    - ~~StringBufferInputStream~~ (StringReader), ~~LogStream~~(no replacement)
- IO Common
    - Console, StreamTokenizer, PrintStream

- IO Decorator для работы с байтами
    - BufferedInputStream, DataInputStream, CipherInputStream, CheckedInputStream,
    - InflaterInputStream, DeflaterInputStream, DigestInputStream, PushbackInputStream
    - ProgressMonitorInputStream , ~~LineNumberInputStream~~ (LineNumberReader)
- IO Reader поток работы с символьными данными
    - Reader BufferedReader, CharArrayReader, InputStreamReader, PipedReader,
    - StringReader, LineNumberReader, FilterReader, PushbackReader,
    - Writer PrintWriter

Класс Stream		Класс Reader, Writer		Примечание
InputStream		Reader адаптер : InputStreamReader		
OutputStream		Writer адаптер : OutputStreamWriter		
FileInputStream		FileReader		
FileOutputStream		FileWriter		
StringBufferinputStream		StringReader		
---		StringWriter		
ByteArrayInputStream		CharArrayReader		
ByteArrayOutputStream		CharArrayWriter		
PipedInputStream		PipedReader		
PipedOutputStream		PipedWriter		

- IO Decorator для работы с символами

Класс Stream		Класс Декораторы Reader, Writer		Примечание
FilterInputStream		FilterReader		
FilterOutputStream		FilterWriter		
BufferedInputStream		BufferedReader		
BufferedOutputStream		BuffereWriter		
DataInputStream		DataInputStream, для строк BufferedReader		
PrintStream		PrintWriter		
StreamTokenizer		StreamTokenizer		
PushBackInputStream		PushBackReader		

-

-

Java IO File Interfaces

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java IO File Interfaces интерфейсы для работы с файлами или потоками
  - DataInput чтение байт и примитивов read(), readInt(), skip()
  - ObjectInput чтение объектов read(), skip(), close()
  - FileFilter фильтрация файлов по пути accept()
  - FilenameFilter фильтрация файлов по имени файлов accept()
  - Flushable сброс буфера flush()
  - Closeable закрытие потока close()
- Java File Classes классы работы с файлами
  - File класс работы с файлами и каталогами
  - FileDescriptor дескриптор к файлу, на базе которого открыт FileInputStream
  - FilePermission доступ к файлу или каталогу, проверяемые SecurityManager

Java IO File Classes

- Java IO File interfaces интерфейсы для работы с файлами или потоками
  - FileFilter фильтрация файлов по пути accept()
  - FilenameFilter фильтрация файлов по имени файлов accept()
- JavaIO File classes работа с файлами
  - File открывает файл или каталог для чтения, записи
  - FileDescriptor открывает дескриптор файла потока FileInputStream().getFD()
  - FilePermission
  - SerializablePermission разрешения для потомков классов ObjInputStream и ObjOutputStream
  - смотреть в примере, там активно используется Custom SecurityManager
  - системный SecurityManager по умолчанию ничего не проверяет
  - RandomAccessFile произвольный доступ к файлу

FileDescriptor

- FIleDescriptor доступ к содержимому файла по дескриптору
  - открытие по дескриптору файла создается shallow клон потока, по сути просто ссылка
  -
- ВНИМАНИЕ. НЕ ПОЛЬЗОВАТЬСЯ new FileInputStream(Ffs.getFd()) так как это ведет к путанице
- Пример. реализация **java8_nio/io/Main02SD**

```
System.out.printf(format, "FileInputStream");
try {
    FileInputStream fs = new FileInputStream( "./data/result.txt");
    FileInputStream fs2 = new FileInputStream(fs.getFD());
    FileInputStream fs3 = new FileInputStream( new File( "./data/result.txt"));
    readout(fs);
    readout(fs2,new byte[10]);            // по сути это тот же самый поток и он уже закрыт
    readout(fs3, new byte[25]);
    fs = new FileInputStream( "./data/result.txt");
    fs2 = new FileInputStream(fs.getFD());
    readout(fs,new byte[50],50);                    // читаем попеременно
    readout(fs2,new byte[50],50);
    readout(fs,new byte[50]);                      // закрываем оба потока
    System.out.println("fs2:"+fs2.available());     // поток тоже закрыт
}catch (IOException e) {
    System.out.println("IOException:"+e);
}
```

  - I

FilePermission

- FilePermission разрешения SecurityManager для файла или каталога
- Пример. реализация java8_nio/io/Main01F
- ```
  try {
      String path = PATH + "result.txt";
      FilePermission fp1 = new FilePermission(PATH + "-", "read");
      PermissionCollection pc = fp1.newPermissionCollection();
      pc.add(fp1);
      FilePermission fp2 = new FilePermission(path, "write");
      pc.add(fp2);
      Enumeration<Permission> en = pc.elements();
      while(en.hasMoreElements()){
          Permission p = en.nextElement();
          System.out.println(p.getName()+" actions:"+p.getActions());
      }

      if (pc.implies(new FilePermission(path, "read,write"))) {
          System.out.println("Permission for " + path + " is read and write.");
      } else {
          System.out.println("No read, write permission for " + path);
      }
  } catch (Exception ex) {
      ex.printStackTrace();
  }
  ```

FileNameFilter

- FIleNameFilter класс фильтра по имени файла
- Пример. реализация поиска каталогов по маске с FilenameFilter java8_nio/io/Main01F

```
private static FilenameFilter filter(final String wildCard) {
    return new FilenameFilter() {
        final String regex = wildCard.replaceAll("*", ".*");
        private Pattern pattern = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);
        @Override
        public boolean accept(File dir, String name) {
            return pattern.matcher(name).matches();
        }
    };
}
private static List<String> getDir(String sPath, String sPattern) {
    File path = new File(sPath);
    if (!path.exists()) {
        System.out.println("Path not exists!");
        return null;
    }
    String[] paths = path.list(filter(sPattern));
    if (paths == null) return null;
    return Stream.of(paths)
            .sorted(String.CASE_INSENSITIVE_ORDER)
            .collect(Collectors.toList());
}
public static void main(String[] args) {
    List<String> list = getDir("D:/temp2", "*b*");
    System.out.println(list);
}
```
```

Catalog Tree Example

- Catalog Tree Example
- Пример. реализация рекурсивного поиска каталогов по regex **java8_nio/io/Main01F**

```
    private static class TreeInfo implements Iterable<File> {
        private List<File> listDirs;
        public TreeInfo() {
            this.listDirs = new ArrayList<>();
        }
        @Override
        public Iterator<File> iterator() {
            return listDirs.iterator();
        }
        public List<File> getListDirs() {
            return listDirs;
        }
        public boolean isEmpty() {
            return listDirs == null || listDirs.isEmpty();
        }
        public void addAll(TreeInfo treeInfo) {
            if (treeInfo == null || treeInfo.isEmpty()) return;
            listDirs.addAll(treeInfo.getListDirs());
        }

        public void add(File file) {
            if (file == null || !file.exists()) return;
            listDirs.add(file);
        }
        @Override
        public String toString() {
            if (listDirs == null || listDirs.isEmpty()) return "[]";
            return listDirs.toString();
        }

        private static TreeInfo recursive(File filePath, Pattern pattern) {
            TreeInfo treeInfo = new TreeInfo();
            File[] paths = filePath.listFiles();                    // любые каталоги совпадение по факту
            if (paths == null) return null;
            for (File f : paths) {
                if (f.isFile()) continue;
                treeInfo.addAll(recursive(f, pattern));
                if (pattern.matcher(f.getName()).matches()) treeInfo.add(f); // проверка тут, так как
            }                                                       // совпадение может на любом уровне
            return treeInfo;
        }
        public static TreeInfo walk(String path, String wildCard) {
            File filePath = new File(path);
            if (!filePath.exists()) return null;
            String regex = wildCard.replaceAll("\\*", "\\.\\*");
            Pattern pattern = Pattern.compile(regex,Pattern.CASE_INSENSITIVE);
            return recursive(filePath, pattern);
        }
    }
    public static void main(String[] args) {
        TreeInfo treeInfo = TreeInfo.walk("D:/temp2", "*b[al]*");
        System.out.println(treeInfo);
    }
```

SerializablePermission

- SerializablePermission — класс permissions для потомков классов Serializable потоков
  - permissions — стандартные и пользователя
- `USER_SERIALIZABLE_PERMISSION = new SerializablePermission("userSerializable");`
  - применение — создается свой SecurityManager, checkPermission()проверяет выбранные
  - SerializablePermissions и выдает нужные SecurityExceptions(message)
  - в user классе — вызывается SecurityManager и проверяет выбранное SerializablePermission
- SerializablePermission
  - штатных разрешений всего два

```
new  SerializablePermission("enableSubclassImplementation");
new SerializablePermission("enableSubstitution");
```

  - в классах ObjInputStream и ObjOutputStream проверка на эти разрешения запрещена
  - для классов потомков, встроена проверка на разрешения
  - ObjectOutputStream — проверяется `enableSubclassImplementation`
  - для методов `putFields(), writeUnshared()`
  - проверяется `enableSubstitution`
  - для метода `enableReplaceObject()`
  - ObjectInputStream — проверяется `enableSubclassImplementation`
  - для методов `readFields (),readUnshared ()`
  - проверяется `enableSubstitution`
  - для метода `enableResolveObject()`
  - User Permissions — в методе пользователя вызвать SecurityManager.checkPermission()
  - в самом SecurityManager проверить свое SerializablePermission
- Пример. реализация — java8_nio/io/Main01F
- ВНИМАНИЕ. Здесь НЕ ПОКАЗАНЫ классы пользователя и методы проверка объектов смотреть в коде.

```
final Class<?>[] TEST_CLASSES = { SubTest1Out.class, SubTest2Out.class,
                                  SubTest3Out.class,SubTest4Out.class, SubTest5Out.class   };
SecurityManager oldSm = null;
try {
    SecurityManager sm = new SecurityManager() {
        final SerializablePermission SUBCLASS_IMPLEMENTATION_PERMISSION =
                new SerializablePermission("enableSubclassImplementation");
        final SerializablePermission SUBSTITUTION_PERMISSION =
                new SerializablePermission("enableSubstitution");
        final SerializablePermission USER_SERIALIZABLE_PERMISSION =
                new SerializablePermission("userSerializable");
        @Override
        public void checkPermission(Permission perm) {
            if (perm.equals(SUBCLASS_IMPLEMENTATION_PERMISSION))
                throw new SecurityException("Enable Subclass Implementation");
            if (perm.equals(SUBSTITUTION_PERMISSION))
                throw new SecurityException("Enable Substitution");
            if (perm.equals(USER_SERIALIZABLE_PERMISSION))
                throw new SecurityException("User Serializable");
        }
    };
    oldSm = System.getSecurityManager();
    System.setSecurityManager(sm);
    for (Class c : TEST_CLASSES) {
        System.out.printf(FORMAT, c.getSimpleName() + ".class:");
        try {
             checkSerializableOut((Class<? extends ObjectOutputStream>)c, PATH + "person.dat");
        } catch (SecurityException e) {
            System.out.println("SecurityException:" + e);
        }
    }
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
} finally {
    if (oldSm != null)   System.setSecurityManager(oldSm);  // restore SecurityManager
}
```

RandomAccessFile

- RandomAccessFile класс произвольного доступа к содержимому файла
  - свойства поиск seek() по файлу
  - read(), write() операции чтения и записи
  - readInt(), readUTF() работа с примитивами
- Пример. реализация
-

```
RandomAccessFile ra = null;
BufferedReader br = null;
try {
    ra = new RandomAccessFile(PATH + "random.txt", "rw"); // чтение и запись
    br = new BufferedReader(new FileReader(PATH + "result.txt"), 100);
    String s;
    double d = 1.23;
    int counter = 1;
    while ((s = br.readLine()) != null) {
        s = String.format("%s", s);
        ra.writeInt(s.length());        // 4 bytes
        ra.writeUTF(s);                 // s.length() + 2
        ra.writeDouble(d);              // 8 bytes
        ra.writeDouble(d * counter);    // 8 bytes
        counter++;
    }
    ra.seek(0);          // set to 0 pos  rewind
    long pos2 = 0;
    long pos3 = 0;
    for (int i = 0; i < 4; i++) {
        int len = ra.readInt();  //
        s = ra.readUTF();        //
        if(i == 2) pos2 = ra.getFilePointer();
        if(i == 3) pos3 = ra.getFilePointer();
        double d1 = ra.readDouble();
        double d2 = ra.readDouble();
        System.out.printf("%02d <%-25s> %5.2f %5.2f%n", len, s, d1, d2);
    }
    ra.seek(pos2);
    ra.writeDouble(Math.random()*100);
    ra.seek(pos3);
    ra.writeDouble(Math.random()*100);
    ra.close();
    System.out.printf(FORMAT,"Random seek():");
    ra = new RandomAccessFile(PATH + "random.txt", "rw"); // чтение и запись
    ra.seek(25);          // set to 0 pos
    for (int i = 0; i < 4; i++) {
        int len = ra.readInt();  //
        s = ra.readUTF();        //
        double d1 = ra.readDouble();
        double d2 = ra.readDouble();
        System.out.printf("%02d <%-25s> %5.2f %5.2f%n", len, s, d1, d2);
    }
    ra.close();
    System.out.printf(FORMAT,"win-1251:");
    ra = new RandomAccessFile(PATH + "result_w.txt", "r"); // чтение и запись
    int len;
    byte[] bytes = new byte[50];
    while((len = ra.read(bytes))> 0) {
        System.out.printf("%s",new String(bytes,0,len,Charset.forName("WINDOWS-1251")));
    }
    ra.close();
    System.out.printf(FORMAT,"koi8r:");
    ra = new RandomAccessFile(PATH + "result_k.txt", "r"); // чтение и запись
    while((len = ra.read(bytes))> 0) {
        System.out.printf("%s",new String(bytes,0,len,Charset.forName("KOI8-R")));
    }
    ra.close();

} catch (
        IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(ra);
    IOUtils.closeStream(br);
}
```

Java IO Common Classes

- JavaIO Common classes общие классы
  - Console работа с системной консолью, ЗАПУСКАТЬ только с консоли
  - StreamTokenizer nextToken()автоматом разбивает строку на слова,распознает числа, кавычки
  - PrintStream работа на вывод на печать

Console

- Console работает с системной консолью
  - открытие в IDEA не работает System.console() возвращает null
  - запускать с консоли или через Runtime, *.bat
  - методы readline() чтение с консоли
  - printf() печать в консоль
- ВНИМАНИЕ. Альтернатива, ИСПОЛЬЗОВАТЬ потоки System.in и System.out и классы Reader, Writer
- Пример. реализация **java8_nio/io/Main02C**

```
System.out.printf(FORMAT, "Console:");
Console console = null;
console = System.console();
if (console != null) {
    String s;
    console.printf("Type any text and press <Enter>('exit' for exit):%n: ");
    while ((s = console.readLine()) != null) {
        console.printf(": %s%n: ", s);
        if (s.equals("exit")) {
            console.printf("Press any key to close...");
            console.readLine();
            System.exit(0);
        }
    }
}
```

- Пример. реализация запуск из исходников **java8_nio/io/Main02C**

```
try {
    String path = "java_nio/src/io;java_nio/src";
    String fileName = "java_nio/src/io/console/MainConsole.java";
    String fileClass = "io.console.MainConsole";

    Runtime.getRuntime().exec("javac -cp " + path + " " + fileName);
    Runtime.getRuntime().exec("cmd /c start java -cp "+path+" "+fileClass);

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

- Пример. реализация запуск из классов **java8_nio/io/Main02CC**

```
try {
    System.out.printf(FORMAT,"Run classes:");
    String path = "out/production/java_nio/io;out/production/java_nio;";
    String fileClass = "io.console.MainConsole";
    Runtime.getRuntime().exec("cmd /c start java -cp "+path+" "+fileClass);

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

-

StreamTokenizer

- StreamTokenizer позволяет разбить поток токенами, распознает текст, числа, комментарии
  - символы \u0000..\u00FF
  - методы commentChar() задает токен комментария
  - slachSlashComment() задает распознавание комментариев C++ //
  - slachStarComment() задает распознавание комментарией C /*
  - ordinaryChar() задает токен разделителя
  - quoteChar() задает токен кавычек, позволяет вытащить слов внутри ""
  - pushback pushBack() возвращает последний токен для повторного парсинга
- Пример. реализация **java8_nio/io/Main02C**
-

```
BufferedReader br = null;
try {
    br = new BufferedReader(new InputStreamReader(new FileInputStream(PATH + "token.txt")));
        StreamTokenizer st = new StreamTokenizer(br); // Reader вместо InputStream из за символов
        st.parseNumbers();
        st.slashSlashComments(false);   // не работает
        st.slashStarComments(true);     // работает и включено

        st.commentChar('#');
        st.quoteChar('"');

        IOUtils.readout(new BufferedInputStream(new FileInputStream(PATH + "token.txt")));
        int typeToken;
        int counter = 0;
        while ((typeToken = st.nextToken()) != StreamTokenizer.TT_EOF) {
            switch (typeToken) {
                case StreamTokenizer.TT_WORD:
                    System.out.println(st.lineno() + " word:" + st.sval);
                    break;
                case StreamTokenizer.TT_NUMBER:
                    System.out.println(st.lineno() + " number:" + st.nval);
                    break;
                case StreamTokenizer.TT_EOL:
                    System.out.println(st.lineno() + " eol: ");
                    break;
                case '"':
                    System.out.println(st.lineno() + " quoted: " + st.sval);
                    break;
                case '*':
                    System.out.println(st.lineno() + " commnt: " + st.sval);
                                                    // сработает если открыть комменты /*
                    break;
                default:
//                      System.out.println("unknown: "+st.sval);
                    break;
            }
            counter++;
            if(counter%5 == 0 ){
                System.out.println("push back:");
                st.pushBack();  // push back every 5th token
            }

        }

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        IOUtils.closeStream(br);
    }
```

PrintStream

- PrintStreamоткрывает поток на печать
    - печатает форматированные строки, символы в стандартный поток или System.out
- Пример. реализация **java8_nio/io/Main02C**
- ```
  PrintStream ps = null;
  BufferedOutputStream out = null;
  BufferedInputStream in = null;
  try {
      out =  new BufferedOutputStream(new FileOutputStream(PATH + "print.txt"), 100);
      ps = new PrintStream(out);
      in = new BufferedInputStream(new FileInputStream(PATH + "result.txt"), 100);  // internal buffer
      byte[] bytes =  new byte[100];
      int len;
      while((len = in.read(bytes))> 0) {
          ps.printf("%s",new String(bytes,0,len).replaceAll("* ","*_"));
      }
      ps.close();
      out.close();
      in.close();
      in = new BufferedInputStream(new FileInputStream(PATH + "print.txt"), 100);  // internal buffer
      ps = new PrintStream(System.out);
      while((len = in.read(bytes))> 0) {
          ps.printf("%s",new String(bytes,0,len));
      }

  } catch (IOException e) {
      e.printStackTrace();
  } finally {
      IOUtils.closeStream(in);
      IOUtils.closeStream(out);
      IOUtils.closeStream(ps);
  }
  ```
-
  ```

Java IO Stream

- Java IO система ввода вывода **java8_nio/io/Main02S**
  - stream это поток данных на ввод или вывод, producer или consumer откуда угодно
  - источником данных может быть файл, буфер памяти или интернет
- InputStream абстрактный класс producer stream
  - child AudioInputStream
  - ByteArrayInputStream не требует закрывать поток close()
  - FileInputStream, поток к файлу
  - ObjectInputStream поток для записи примитивов и объектов
  - SequenceInputStream объединяет два потока в один, друг за другом
  - PipedInputStream создает поток пользователя на базе Runnable
  - FilterInputStream абстрактный класс для декораторов
  - deprecated StringBufferInputStream рекомендован StringReader
  -
  - methods read(), mark(), reset()
  - read() читает данные побайтно, в массив byte[], в часть массива byte[] по offset и len
  - mark() устанавливает метку на текущей позиции потока и число байт [limit]
  - reset() сбрасывает указатель потока к метке mark() если не превышен [limit] при чтении
  - если при чтении с позиции mark() превышено чтениех[limit] может дать Exception
- ВНИМАНИЕ. СРАБАТЫВАЕТ для BufferedInputStream если размер буфера меньше размера данных
- Пример. реализация **java8_nio/io/Main02S**

```
public static void checkMark(InputStream in, boolean isExceeded) {
    if(!in.markSupported()) {
        System.out.println("mark() is not supported");
        return;
    }
    try {
        byte[] bytes = new byte[100];
        int size = in.available() - 4;
        in.read();  // read 2 bytes
        in.read();
        in.mark(size);
        if (isExceeded) size++;
        int len;
        while (size > 0 && (len = in.read(bytes, 0, size > bytes.length ? bytes.length : size)) > 0) {
            System.out.printf("%s", new String(bytes, 0, len));
            size -= bytes.length;
        }
        System.out.println();
        System.out.println("read : ok");
        in.reset();
        System.out.println("reset: ok");
    }catch (IOException e) {
        e.printStackTrace();
    } finally {
        IOUtils.closeStream(in);
    }
}
```

- Пример. реализация reset() выбрасывает Exception для BufferedInputStream **java8_nio/io/Main02S**

```
System.out.printf(FORMAT, "InputStream mark():");
InputStream in = null;
try {
    in = new BufferedInputStream(new FileInputStream(PATH+"result.txt"),100);
    IOUtils.checkMark(in, false);
    in = new BufferedInputStream(new FileInputStream(PATH+"result.txt"),100);
    IOUtils.checkMark(in, true);
} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(in);
    }
```

ByteArrayInputStream

- ByteArrayInputStream создает поток из массива данных
  - свойства не требует закрытия потока методом close()
  - reset() ByteArrayStream НЕ ВЫБРАСЫВАЕТ Exception при превышении [limit]
- Пример. реализация в размер массива или части массива **java8_nio/io/Main02SD**

```
ByteArrayInputStream bs = new ByteArrayInputStream(s.getBytes("utf-8");));
ByteArrayInputStream bs2 = new ByteArrayInputStream(s.getBytes("utf-8"), 5, 27);
System.out.println("bs:" + bs.available());
readout(bs);
readout(bs2);
System.out.println();
```

FileInputStream

- FileInputStream открывает поток к файлу
  - открытие по имени файла, по файлу
  - по дескриптору файла создается sdhallow клон потока, по сути просто ссылка
- ВНИМАНИЕ. НЕ ПОЛЬЗОВАТЬСЯ new FileInputStream(Ffs.getFd()) так как это ведет к путанице
- Пример. реализация **java8_nio/io/Main02SD**

```
System.out.printf(format, "FileInputStream");
try {
    FileInputStream fs = new FileInputStream( "./data/result.txt");
    FileInputStream fs2 = new FileInputStream(fs.getFD());
    FileInputStream fs3 = new FileInputStream( new File( "./data/result.txt"));
    readout(fs);
    readout(fs2,new byte[10]);            // по сути это тот же самый поток и он уже закрыт
    readout(fs3, new byte[25]);
    fs = new FileInputStream( "./data/result.txt");
    fs2 = new FileInputStream(fs.getFD());
    readout(fs,new byte[50],50);                    // читаем попеременно
    readout(fs2,new byte[50],50);
    readout(fs,new byte[50]);                      // закрываем оба потока
    System.out.println("fs2:"+fs2.available());    // поток тоже закрыт
}catch (IOException e) {
    System.out.println("IOException:"+e);
}
```

SequenceinputStream

- SequenceinputStream объединяет два потока в один, друг за другом
  - читает два потока, один за другим, допускает объединение большого числа потоков
  - открытие два потока, объект Enumeration<T> , аналог Iterator<T>
  - свойства available() работает только для текущего активного потока
  - read() ПЕРЕКЛЮЧАЕТ на другой поток, если текущий закончился
- Пример. реализация **java8_nio/io/Main02SD**

```
try {
    List<InputStream> list=new ArrayList<>();
    for (int i = 0; i < 5; i++) {
        list.add(new ByteArrayInputStream(s.getBytes("utf-8")));
    }
    Enumeration<InputStream> en = new Enumeration<InputStream>() {
        final Iterator<InputStream> it = list.iterator();
        public boolean hasMoreElements() {
            return it.hasNext();
        }
        public InputStream nextElement() {
            return it.next();
        }
    };
  SequenceInputStream sIn = new SequenceInputStream(en);
  SequenceInputStream sIn2 = new SequenceInputStream(sIn, new FileInputStream("./data/result.txt"));
    readout(sIn2);
}catch (IOException e) {
}
```

ObjectInputStream

- ObjectInputStream поток для сериализации объектов и примитивов
  - свойства работает только с классами реализующими Serializable
  - работает только с файлами созданными на базе objectOutputStream
  - при чтении объекта, когда поток пуст, вылетает EOFException, игнорировать
  -
- ВНИМАНИЕ. EOFException надо просто поймать и игнорировать
- Пример. реализация **java8_nio/io/Main02SD**
- ```
  // ObjectInputStream
          System.out.printf(format, "ObjectInputStream:");
          ObjectInputStream sin = null;
          ObjectOutputStream sout = null;
          in = null;
          out = null;
          try {
              sout = new ObjectOutputStream(
                      new BufferedOutputStream(
                              new FileOutputStream("./data/person.txt"), 100)
              );
              for (int i = 0; i < 10; i++) {
                  sout.write(String.format("%s%n", s).getBytes("utf-8"));
              }
              sout.flush();
              sout.close();
              in= new BufferedInputStream(new FileInputStream("./data/person.txt"), 100);//internal buffer
              sin = new ObjectInputStream(in);  // internal buffer
              int c;
              while ((c = sin.read()) != -1) {
                  System.out.printf("%c", c);
              }
              sin.close();
              in.close();  // underlying stream close

              sout = new ObjectOutputStream(
                      new BufferedOutputStream(
                              new FileOutputStream("./data/person.dat"), 100)
              );
              for (Base p : BaseFactory.newList(10)) {
                  sout.writeObject(p);
              }
              sout.flush();
              sout.close();

              in= new BufferedInputStream(new FileInputStream("./data/person.dat"), 100);//internal buffer
              sin = new ObjectInputStream(in);  // internal buffer
              List<Base> listB = new ArrayList<>();
              Base b;
              try {
                  while ((b = (Base) sin.readObject()) != null) {
                      listB.add(b);
                  }
              } catch (EOFException e) {                // eof reached
              } finally {
                  sin.close();
                  in.close();
              }
              listB.stream().sorted(
                      Comparator.comparing(v->v.getClass().getName())
                      .thenComparing(v->((Base)v).getPrice())
                      .thenComparing(v->((Base)v).getName())
                      .thenComparing(v->((Base)v).getHeight()))
                      .forEach(p -> System.out.printf("%s%n", p));

          } catch (IOException | ClassNotFoundException e) {
              e.printStackTrace();
          } finally {
              IOUtils.closeStream(in);
          }
  ```
  ```

AudioInputStream

- AudioInputStream работа с аудио файлами WAV
  - свойства открывает поток для работы с файлами wav
  - mp3 НЕ ПОДДЕРЖИВАЕТСЯ нужны библиотеки
  - mp3 spi подключается библиотека динамически с помощью JDK13services.getProviders()
  - метод AudioSystem.getAudioInputStream () использует JDK13services.getProviders()
  -
- ВНИМАНИЕ. НЕЯСНО подключение mp3spi библиотек, почему тот же код с ними работает, без них нет
- Пример. реализация аудио потока **java8_nio/io/Main02SD**

```
System.out.printf(format, "AudioInputStream Wav:");
in = null;
AudioInputStream as = null;
SourceDataLine line = null;
try {
    as = AudioSystem.getAudioInputStream(new File("./data/short.wav"));
    AudioFormat fmt = as.getFormat();

    long frames = as.getFrameLength();
    double duration = (double) frames / fmt.getFrameRate();
    System.out.printf("audio: %.2f sec%n", duration);

    DataLine.Info info = new DataLine.Info(SourceDataLine.class, fmt);
    line = (SourceDataLine) AudioSystem.getLine(info);

    line.open(fmt);
    line.start();
    byte[] bytes = new byte[65536];
    int n;
    while ((n = as.read(bytes)) != -1) {
        line.write(bytes, 0, n);
        System.out.print(".");
    }
    line.drain();
    line.stop();
    System.out.printf(format, "AudioInputStream MP3:");
    AudioMP3.testPlay("./data/short.mp3");

} catch (IOException | UnsupportedAudioFileException | LineUnavailableException e) {
    e.printStackTrace();

} finally {
    IOUtils.closeStream(in);
    if (line != null) line.close();
}
```

- Пример. реализация динамического подключения библиотек

```
System.out.printf(format, "Audio Providers:");
List<Class> listC = new ArrayList<>();
listC.add(MixerProvider.class);
listC.add(FormatConversionProvider.class);
listC.add(AudioFileReader.class);
listC.add(AudioFileWriter.class);
listC.add(MidiDeviceProvider.class);
listC.add(SoundbankReader.class);
listC.add(MidiFileWriter.class);
listC.add(MidiFileReader.class);

for (Class c : listC) {
    System.out.printf(format,"Providers for "+c.getName()+":");
    List<?> listP = JDK13Services.getProviders(c);
    listP.forEach(v -> System.out.printf("%s%n", v.getClass().getName()));

}
```

PipedInputStream

- PipedInputStream специальный класс для создания потоков внутри JVM
    - работает только с потоком PipedOutputStream
    - создается на базе Runnable для передачи данных пользователя
- Пример. реализация **java8_nio/io/Main02SD**

```
final PipedOutputStream po = new PipedOutputStream();
final PipedInputStream pi = new PipedInputStream();
final ReentrantLock sLock = new ReentrantLock();
Runnable rOut = () -> {
    try {
        po.connect(pi);
        po.write(s.getBytes("utf-8"));
        po.flush();
        po.close();
        synchronized (sLock) {
            sLock.wait();
        }
    } catch (InterruptedException | IOException e) {
        e.printStackTrace();
    }finally {
        closeStream(po);
    }
};
Runnable rIn = () -> {
    try {
        while (pi.available() == 0) {};
        byte[] bytes = new byte[10];
        int len;
        while ((len = pi.read(bytes)) > 0) {
            System.out.printf("%s", new String(bytes,0,len));
        }
        System.out.println();
        synchronized (sLock) {
            sLock.notifyAll();
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        closeStream(pi);
    }
};
ExecutorService exec = Executors.newFixedThreadPool(2);
exec.execute(rOut);
exec.execute(rIn);
exec.shutdown();
```

StringBufferInputStream

- StringBufferInputStream замена StringReader **java8_nio/io/Main02SD**
- StringReader создает поток из строки возможно чтение строки из файла
- Пример. реализация

```
try {
    String s = new String(Files.readAllBytes(Paths.get("./data/result.txt")),Charset.forName("utf-8"));
    StringReader sr = new StringReader(s);
    char[] chars = new char[10];
    int len;
    while ((len = sr.read(chars)) > 0) {
        System.out.print(new String(chars, 0, len));
    }
} catch (IOException e) {
}
```

FilterInputStream

- FilteredInputStream абстрактный класс для создания декораторов
  - буферы BufferedInputStream обертка потока буферизацией
  - данные DataInputStream работает с примитивами кроме readLine()
  - readLine) deprecated замена BufferedReader.readLine
  - шифры CipherInputStream
  - CheckedInputStream
  - InflaterInputStream, DeflaterInputStream
  - DigestInputStream
  - PushbackInputStream
  - ProgressMonitorInputStream
  - deprecated LineNumberInputStream замена LineNumberReader
  -

BufferedInputStream

- BufferedInputStream буферизация потока
  - свойства оптимизирует работу любого потока
  - позволяет обернуть любой поток, который наследует InputStream
  - поток клиента теперь оптимизирован использованием буферизации
  - поток клиента теперь поддерживает методы mark() и reset()
- Пример. реализация **java8_nio/io/Main03D**
-

```
        BufferedInputStream bs = null;
        BufferedInputStream bs2 = null;
        SequenceInputStream ss = null;
        List<InputStream> list=new ArrayList<>();
        try {
            for (int i = 0; i < 2; i++) {
                list.add(new ByteArrayInputStream(s.getBytes("utf-8")));
            }
            list.add(new FileInputStream("./data/result.txt")); // does not support mark()
//            Enumeration<InputStream> en = new IOUtils.Enumerator<>(list.iterator());
            Enumeration<InputStream> en = new Enumeration<InputStream>() {
                final Iterator<InputStream> it = list.iterator();
                @Override
                public boolean hasMoreElements() {
                    return it.hasNext();
                }
                @Override
                public InputStream nextElement() {
                    return it.next();
                }
            };
            ss = new SequenceInputStream(en);           // does not support mark()
            bs = new BufferedInputStream(ss);
            bs2= new BufferedInputStream(new FileInputStream("./data/result.txt"),100);//internal buffer
            IOUtils.checkMarkSequence(bs,75);       // supports mark()
            IOUtils.readout(bs);                        // up to the end
            IOUtils.readout(bs2);                        // up to the end

        }catch (IOException e) {
            e.printStackTrace();
        }finally {
            IOUtils.closeStream(list);
            IOUtils.closeStream(ss);
            IOUtils.closeStream(bs);
            IOUtils.closeStream(bs2);
        }
```

DataInputStream

- DataInputStream декоратор потока
  - свойства позволяет записывать и читать примитивные типы данных в поток
  - deprecated readline() не распознает правильно символы \r\n
  - замена BufferedReader.readLine()
- Пример. реализация **java8_nio/io/Main03D**

```
        System.out.printf(format, "DataInputStream:");
        DataInputStream dIn = null;
        DataOutputStream dOut = null;
        BufferedInputStream in = null;
        BufferedOutputStream out = null;
        BufferedReader br = null;               // readline support
        try {
            out = new BufferedOutputStream(new FileOutputStream("./data/data.txt"));
            dOut = new DataOutputStream(out);

            dOut.writeBoolean(true);
            dOut.writeChar('$');
            dOut.writeByte(115);
            dOut.writeInt(12003);
            dOut.writeLong(100254L);
            dOut.writeFloat(2.75F);
            dOut.writeDouble(1.235);
            dOut.writeUTF("Проверка кодировки UTF8");
            dOut.flush();
            dOut.close();

            in = new BufferedInputStream(new FileInputStream("./data/data.txt"));
            in.mark(0);
            dIn = new DataInputStream(in);                  // support mark for DataInputStream()
            System.out.println("bytes:" + dIn.available());
            int c;
            while ((c = dIn.read()) != -1) {
                System.out.printf("%02X ", c);
            }
            System.out.println();
            System.out.println("dIn:" + dIn.available());
            in.reset();
            System.out.println("dIn:" + dIn.available());

            System.out.printf("bool:%5b char:%c byte:%3d int:%8d long:%8d float:%8.3f double:%8.3f%n",
                    dIn.readBoolean(), dIn.readChar(), dIn.readByte(), dIn.readInt(), dIn.readLong(),
                    dIn.readFloat(), dIn.readDouble());
            System.out.printf("%s",dIn.readUTF());
// readline
            System.out.printf(format,"BufferedReader:");
            in.close();
            in = new BufferedInputStream(new FileInputStream("./data/result.txt"));
//            DataInputStream dIn = new DataInputStream(in);    //              dIn.readline() deprecated
            br = new BufferedReader(new InputStreamReader(in)); // closes in    br.readline() recommended
            in.mark(in.available());
            String line;
            while ((line = br.readLine()) != null) {  // readline
                System.out.printf("%s%n", line);
            }
            in.reset();
            System.out.println("mark:");
            while ((line = br.readLine()) != null) {  // readline
                System.out.printf("%s%n", line);
            }
            br.close();

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeStream(dIn);
            IOUtils.closeStream(dOut);
            IOUtils.closeStream(br);
            IOUtils.closeStream(in);
            IOUtils.closeStream(out);
        }
```

CipherInputStream

- CipherInputStream используется в потоках с шифрованием данных
    - свойства
    - позволяет
- Пример. реализация **java8_nio/io/Main03D**

```
// CipherInputStream
        System.out.printf(format, "CipherInputStream:");
        CypherUtils.main(args);
```

-

CheckedInputStream

- CheckedInputStream декоратор потока
    - свойства включает в поток checksum и проверку целостности данных
    - позволяет проверять целостность данных потока
- Пример. реализация **java8_nio/io/Main03D**

```
System.out.printf(format, "CheckedInputStream:");
in = null;
CheckedInputStream cin = null;
try {
    in= new BufferedInputStream(new FileInputStream("./data/result.txt"), 100);//internal buffer
    Checksum cIF = new Checksum() {
        private long checksum = 0;
        @Override
        public void update(int b) {
            checksum += b;
        }
        @Override
        public void update(byte[] b, int off, int len) {
            for (int i = off; i < off + len; i++) {
                checksum += b[i]; // XOR
            }
        }
        @Override
        public long getValue() {
            return checksum;
        }
        @Override
        public void reset() {
            checksum = 0;
        }
    };
    cin = new CheckedInputStream(in, cIF);
    cin.mark(cin.available());
    Checksum checksum = cin.getChecksum();
    System.out.println("checksum:" + checksum.getValue());
    int c;
    while ((c = cin.read()) != -1) {
        System.out.printf("%c", c);
        if (cin.available() < 10) break;
    }
    System.out.println();

    System.out.println("checksum:" + checksum.getValue());
    cin.reset();
    checksum.reset();
    System.out.println("checksum:" + checksum.getValue());
    while ((c = cin.read()) != -1) {
        System.out.printf("%c", c);
        if (cin.available() < 10) break;
    }
    System.out.println();
    System.out.println("checksum:" + cin.getChecksum().getValue());
} catch (
        IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(in);
    IOUtils.closeStream(cin);
}
```

DeflaterInputStream

- DeflaterInputStream декоратор потока сжимает данные используя ZLIB библиотеку
  - свойства реализует сжатие данных в формате "deflate"
  - иерархиясжатия от предка к потомку deflate >> zip >> gzip
- Пример. реализация сжатия и распаковки данных

```
        DeflaterInputStream din = null;
        InflaterInputStream iin = null;
        in = null;
        try {
            in= new BufferedInputStream(new FileInputStream(PATH + "result.txt"), 100);//internal buffer
            out = new BufferedOutputStream(new FileOutputStream(PATH + "result.dft")); // compressed
            Deflater def = new Deflater(BEST_COMPRESSION, false);  // true no header for GZIP and PKZIP
            din = new DeflaterInputStream(in, def);
//            byte[] bytes = din.readAllBytes();  //JDK9
            byte[] bytes = new byte[100];
            int len;
            int source = in.available();
            int compressed = 0;
            while ((len = din.read(bytes)) > 0) {
                out.write(bytes, 0, len);  // compressed data
                compressed += len;
            }
            System.out.println("source: "+source+" compressed: "+compressed);

                din.close();            // close underlying in
                out.close();            // close compressed file
                System.out.println("Inflater:");
                in= new BufferedInputStream(new FileInputStream(PATH + "result.dft"), 100);//internal buffer
                Inflater inf = new Inflater(false); // true  no header for GZIP PKZIP
                iin = new InflaterInputStream(in,inf);
                source = in.available();
                compressed = 0;
                while ((len = iin.read(bytes)) > 0) {
                    compressed += len;
                    System.out.printf("%s",new String(bytes,0,len));
                }
                iin.close();            // close underlying in
                System.out.println("source: "+source+" uncompressed: "+compressed);
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.closeStream(din); // closes underlying in
                IOUtils.closeStream(iin); // closes underlying in
                IOUtils.closeStream(in);
                IOUtils.closeStream(out);
            }
```

InflaterInputStream

- InflaterInputStream декоратор потока распаковывает данные используя ZLIB библиотеку
  - свойства реализует распаковку данных в формате "deflate"
- Пример. реализация методов Deflate и Inflate без потоков

```
try {
    String inputString = "blahblahblah$\u20BD\u20AC";    // Encode a String into bytes
    byte[] input = inputString.getBytes("UTF-8");
    byte[] output = new byte[100];
    Deflater compressor = new Deflater();
    compressor.setInput(input);
    compressor.finish();
    int compressedDataLength = compressor.deflate(output);    // Compress the bytes
    // Decompress the bytes
    Inflater decompressor = new Inflater();
    decompressor.setInput(output, 0, compressedDataLength);
    byte[] result = new byte[100];
    int resultLength = decompressor.inflate(result);
    decompressor.end();
    String outputString = new String(result, 0, resultLength, "UTF-8");// Decode the bytes into a String
    System.out.println(outputString);
} catch (IOException | DataFormatException e) {
    e.printStackTrace();
}
```

LineNumberInputStream

- LineNumberInputStream deprecated замена LineReader
    - свойства разбивает поток на строки и заменяет \r\n на \n
- Пример. реализация LineReader java8_nio/io/Main03D
- ```
  System.out.printf(format, "LineNumberInputStream >> LineNumberReader:");
  in = null;
  LineNumberReader lr = null;
  try {
      in = new BufferedInputStream(new FileInputStream("./data/result.txt"), 100);  // internal buffer
      lr = new LineNumberReader(new InputStreamReader(in));  // translate InputStream >> Reader >>
  LinNumberReader
      while((s=lr.readLine())!=null) {
          System.out.println(lr.getLineNumber()+":"+s);
      }
  } catch (IOException e) {
      e.printStackTrace();
  } finally {
      IOUtils.closeStream(in);
      IOUtils.closeStream(lr);
  }
  ```

DigestInputStream

- DigestInputStream создает поток, который обновляет объект MessageDigest данными потока
    - свойства использует MessageDIgest объект для подсчета хэш кода
    - позволяет вычислить хэш код массива данных
    - дополнение показана работа с URL потоком на базе файла
- Пример. реализация java8_nio/io/Main03D
- ```
          DigestInputStream ds = null;
          in = null;
          try {
              in= new BufferedInputStream(new FileInputStream(PATH + "result.txt"), 100);//internal buffer

              MessageDigest md5 = MessageDigest.getInstance("MD5");
              MessageDigest mds = MessageDigest.getInstance("SHA");

              ds = new DigestInputStream(in, md5);
              ds.mark(ds.available() + 1);          // full stream
              byte[] bytes = new byte[100];
              int len;
              while ((len = ds.read(bytes)) > 0) {
                  System.out.printf("%s", new String(bytes, 0, len));
              }
              byte[] hash = md5.digest();
              System.out.println("hash MD5:" + toHex(hash));
              ds.reset();
              System.out.println("hash MD5:" + toHex(ds.getMessageDigest().digest()));

              ds.setMessageDigest(mds);
              while ((len = ds.read(bytes)) > 0) {
                  System.out.printf("%s", new String(bytes, 0, len));
              }
              hash = ds.getMessageDigest().digest();
              System.out.println("hash SHA:" + toHex(hash));
  // url
              System.out.printf(FORMAT, "URL.stream():");
              URL url = new File(PATH + "result.txt").toURI().toURL();
              ds = new DigestInputStream(url.openStream(), md5);
              System.out.println("hash MD5:" + toHex(ds.getMessageDigest().digest()));
              while ((len = ds.read(bytes)) > 0) { // просто читает поток
              }
              System.out.println("hash MD5:" + toHex(ds.getMessageDigest().digest()));
              } catch (IOException | NoSuchAlgorithmException e) {
                  e.printStackTrace();
              } finally {
                  IOUtils.closeStream(in);
              }
  ```

PushbackInputStream

- PushbackInputStream — поток с возможностью возврата символов
  - свойства — поток имеет метод unread() который допускает возврат замену символов в потоке
  - конструктор задает размер буфера возврата, если возврат больше, то Exception
  - позволяет — производить анализ и переработку потока на ходу
- Пример. реализация замены последовательностей символов потока **java8_nio/io/Main03D**
- ```
  PushbackInputStream pin = null;
  in = null;
  try {
      in = new BufferedInputStream(new FileInputStream(PATH + "result.txt"), 100);  // internal buffer
      pin = new PushbackInputStream(in, 9); // all bytes returned
      final StringBuilder sb = new StringBuilder();
      final byte[] bytes = new byte[100];

      BiPredicate<String, Integer> p = (s1, v1) -> String.CASE_INSENSITIVE_ORDER
              .compare(new String(bytes, 0, v1),s1.substring(1)) == 0;
      int len;
      int c;
      while ((c = pin.read()) > 0) {
          switch (c) {
              default:
                  sb.append((char) c);
                  break;
              case '/':
                  int c2 = pin.read();
                  int c3 = pin.read();
                  if (c2 == '*' && c3 == '*') {
                      sb.append("<!>");
                      break;
                  } else {
                      sb.append('/');
                      pin.unread(c2);
                      pin.unread(c3);
                  }
                  break;
              case '*':
                  s = "* created:"; // case insensitive
                  len = pin.read(bytes, 0, s.length() - 1);
                  if (p.test(s, len)) {
                      sb.append(">> Developed by");
                      break;
                  } else {
                      sb.append(s.charAt(0));
                      pin.unread(bytes, 0, len);
                  }
                  break;
          }
      }
      System.out.printf("%s", sb);
  } catch (
          IOException e) {
      e.printStackTrace();
  } finally {
      IOUtils.closeStream(in);
      IOUtils.closeStream(pin);
  }
  ```
  ```

ProgressMonitorInputStream

- ProgressMonitorInputStream поток для показа монитора при чтения потока
  - свойства создает окно Progress Monitor и показывает прогресс передачи данных
  - позволяет отслеживать прогресс передачи данных в потоке
- ВНИМАНИЕ. Работает СРАЗУ с основным экраном, не требует JFrame
- Пример. реализация **java8_nio/io/Main03D**

```
System.out.printf(FORMAT, "ProgressMonitorInputStream:");
ProgressMonitorInputStream pmin = null;
DigestInputStream dis = null;
in = null;
try {
    in = new BufferedInputStream(new FileInputStream(PATH + "result.txt"), 100);  // internal buffer
    pmin = new ProgressMonitorInputStream(null, "Verifying Key", in);
    ProgressMonitor pm = pmin.getProgressMonitor();
    pm.setMaximum(in.available());
    pm.setMillisToPopup(10);

    MessageDigest ms = MessageDigest.getInstance("SHA-256");
    System.out.println("hash SHA256: " + toHex(ms.digest()));
    dis = new DigestInputStream(pmin, ms);
    try {
        byte[] bytes = new byte[5];
        int len;
        while ((len = dis.read(bytes)) > 0) {
            Thread.sleep(100,10);
        }
    } catch (InterruptedIOException e) {
        System.out.printf("%nCancel pressed...%n");
    }
    byte[] hash = ms.digest();
    System.out.println("hash SHA256: " + toHex(hash));
} catch (IOException | NoSuchAlgorithmException |InterruptedException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(in);
}
```

Java IO Reader Classes

- IO Reader потоки работы с символьными данными **java8_nio/io/Main04R**
  - Reader FileReader
  - BufferedReader
  - CharArrayReader
  - InputStreamReader
  - PipedReader,
  - StringReader
  - LineNumberReader
  - FilterReader
  - PushbackReader
  - Writer PrintWriter
- Пример. реализация

Таблица соответствия классов IO Stream и IO Reader

- Таблица соответствия классов IO Stream и IO Reader

Класс Stream		Класс Reader, Writer		Примечание
InputStream		Reader адаптер : InputStreamReader		
OutputStream		Writer адаптер : OutputStreamWriter		
FileInputStream		FileReader		
FileOutputStream		FileWriter		
StringBufferinputStream		StringReader		
---		StringWriter		
ByteArrayInputStream		CharArrayReader		
ByteArrayOutputStream		CharArrayWriter		
PipedInputStream		PipedReader		
PipedOutputStream		PipedWriter		

Таблица соответствия декораторов IO Stream и IO Reader

- IO Decorator для работы с символами

Класс Stream		Класс Декораторы Reader, Writer		Примечание
FilterInputStream		FilterReader		
FilterOutputStream		FilterWriter		
BufferedInputStream		BufferedReader		
BufferedOutputStream		BuffereWriter		
DataInputStream		DataInputStream, для строк BufferedReader		
PrintStream		PrintWriter		
StreamTokenizer		StreamTokenizer		
PushBackInputStream		PushBackReader		

-

InputStreamReader

- InputStreamReader класс который связывает IOStream и IOReader классы
    - свойства позволяет задать Charset или CharsetDecoder для потока InputStream
    - применение ОБЯЗАТЕЛЬНО использовать BufferedReader для буферизации потока
- Пример. реализация Charset, UserCharsetDecoder UTF-8,WINDOIWS-1251,KOI8-R **java8_nio/io/Main04R**

```
        FileInputStream in = null;
        BufferedReader br = null;
        try {
            in = new FileInputStream(PATH + "result.txt");
            Charset cUTF = Charset.forName("UTF-8");
            Charset cWIN = Charset.forName("WINDOWS-1251");
            Charset cKOI8 = Charset.forName("KOI8-R");
            CharsetDecoder dUTF = new UserDecoder(Charset.forName("UTF-8"), 2, 2);
            CharsetDecoder dWIN = new UserDecoder(Charset.forName("WINDOWS-1251"), 2, 2);
            CharsetDecoder dKOI8 = new UserDecoder(Charset.forName("KOI8-R"), 2, 2);
            in.close();
            System.out.printf(FORMAT, "Charset "+cUTF+":");
            in = new FileInputStream(PATH + "result_u.txt");
            br = new BufferedReader(new InputStreamReader(in, cUTF));
            IOUtils.readout(br, in); // closes br >> InputStreamReader, closes in
            System.out.printf(FORMAT, "User CharsetDecoder "+dUTF+":");
            in = new FileInputStream(PATH + "result_u.txt");
            br = new BufferedReader(new InputStreamReader(in, dUTF));// closes InputStreamReader
            IOUtils.readout(br, in);                                  //closes br >> InputStreamReader,in

            in = new FileInputStream(PATH + "result_w.txt");
            br = new BufferedReader(new InputStreamReader(in, cWIN));
            IOUtils.readout(br, in);

            in = new FileInputStream(PATH + "result_w.txt");
            br = new BufferedReader(new InputStreamReader(in, dWIN)); // closes InputStreamReader
            IOUtils.readout(br, in);
            System.out.printf(FORMAT, "Charset "+cKOI8+":");
            in = new FileInputStream(PATH + "result_k.txt");
            br = new BufferedReader(new InputStreamReader(in, cKOI8));
            IOUtils.readout(br, in);

            System.out.printf(FORMAT, "User CharsetDecoder "+dKOI8+":");
            in = new FileInputStream(PATH + "result_k.txt");
            br = new BufferedReader(new InputStreamReader(in, dKOI8));    // closes InputStreamReader
            IOUtils.readout(br, in);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeStream(in);
        }

    private static class UserDecoder extends CharsetDecoder {
        protected UserDecoder(Charset cs, float averageCharsPerByte, float maxCharsPerByte) {
            super(cs, averageCharsPerByte, maxCharsPerByte);
        }
        @Override
        protected CoderResult decodeLoop(ByteBuffer in, CharBuffer out) {
            final CharsetDecoder decoder = charset().newDecoder();
            CoderResult cr = null;
            try {
                if (!in.hasRemaining()) return CoderResult.UNDERFLOW; // no input data

                final CharBuffer cout = decoder.decode(in);
                cr = decoder.flush(cout);                             // check for flush
                while (cout.hasRemaining()) {
                    out.put(cout.get());
                }
            } catch (CharacterCodingException e) {
                e.printStackTrace();
            }
            return cr;
        }
        @Override
        public String toString() {
            return charset().toString();
        }
    }
```

FileReader

- FileReader — класс чтения символьных данных с файла расширяет InputStreamReader
  - свойства — использует кодировку и размер буфера по умолчанию
  - задать — кодировку можно через конструктор InputStreamReader(in,Charset)
- Пример. реализация — java8_nio/io/Main04R

```
System.out.printf(FORMAT, "FileReader:");
br = null;
try {
    br = new BufferedReader(new FileReader(PATH + "result_u.txt"));  // internal buffer
    String s;
    while((s = br.readLine())!= null) {
        System.out.printf("%s%n",s);
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(br);
}
```

BufferedReader

- BufferedReader — класс буферизации для всех классов Reader
  - свойства — позволяет задать размер буфера
  - mark() — срабатывает reset() Exception если размер чтения больше limit и размера буфера
  - mark(limit) — задает максимальный размер чтения от данной точки
  - BufferedReader(in,size) — задает размер буфера чтения
  - reset() — выдает Exception если прочитано больше чем limit, независимо от размера буфера если limit не превышен, то Exception не будет
- Пример. реализация — java8_nio/io/Main04R

```
fs = null;
br = null;
try {
    fs = new FileInputStream(PATH + "result_u.txt");
    br = new BufferedReader(new InputStreamReader(fs, Charset.defaultCharset()),50);
    IOUtils.checkMark(br, false, fs.available(), buff);
    fs = new FileInputStream(PATH + "result_u.txt");
    br = new BufferedReader(new InputStreamReader(fs, Charset.defaultCharset()),50);
    IOUtils.checkMark(br, true, fs.available(), buff);
 } catch (IOException e) {
 } finally {
    IOUtils.closeStream(br);
    IOUtils.closeStream(fs);
}
```

```
public static void checkMark(BufferedReader br, boolean isExceeded, int size, int bSize) {
    if (!br.markSupported()) {
         return;
    }
    try {
        int limit = size;
        if (isExceeded) limit -= 30;  // делаем меньше
        br.read();  // read 2 bytes
        br.read();
        br.mark(limit);
        String s;
        size = 0;
        while((s = br.readLine())!= null) {  // через построчное чтение
            System.out.printf("%s%n",s);
            size += s.length()+2;
        }
        System.out.println();
        System.out.println("read : ok");
        br.reset();
        System.out.println("reset: ok");
    } catch (IOException e) {
    } finally {
        IOUtils.closeStream(br);
        }
    }
```

CharArrayReader

- CharArrayReader класс работает с массивом символов
  - свойства символы уже являются конечным объектом их нельзя перекодировать
  - можно только закодировать их в байты, а байты закодировать в другие символы
- Пример. реализация CharArrayReader и Charset encoder, decoder **java8_nio/io/Main04R**

```
CharArrayReader cr = null;
 try {
    cr = new CharArrayReader(IOUtils.STRING_ENC.toCharArray());  // utf-8
    char[] chars = new char[50];
    int len;
    while ((len = cr.read(chars)) > 0) {
        System.out.printf("%s", new String(chars, 0, len));
    }
    cr.close();
// encoded
    final CharsetDecoder decoder = Charset.forName("WINDOWS-1251").newDecoder();
    final CharsetEncoder encoder = Charset.forName("WINDOWS-1251").newEncoder();

    char[] chs = IOUtils.STRING_ENC.toCharArray();              // characters
    byte[] bbs = encoder.encode(CharBuffer.wrap(chs)).array();  // encoded WINDOWS-1251 bytes
    chars = decoder.decode(ByteBuffer.wrap(bbs)).array();       // decoded WINDOWS-1251

    System.out.printf("orig:%s", new String(chs));
    System.out.printf("utf8:%s", new String(bbs, Charset.forName("UTF-8")));
    System.out.printf("byte:%s", new String(bbs, Charset.forName("WINDOWS-1251")));
    System.out.printf("char:%s", new String(chars));

    cr = new CharArrayReader(chars);
    cr.mark(chs.length);
    chars = new char[50];
    while ((len = cr.read(chars)) > 0) {
        System.out.printf("%s", new String(chars, 0, len));
    }
    cr.reset();
    while ((len = cr.read(chars)) > 0) {
        System.out.printf("%s", new String(chars, 0, len));
    }
} catch (UnsupportedEncodingException e) {
    System.out.println("Encoding Exception:" + e);
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (cr != null) cr.close();

}
```

PipedReader

- PipedReader — класс работы с потоками символов пользователя
  - свойства — позволяет создать поток пользователя и работает с PipedWriter
  - блокирует Thread при чтении PipedReader.read() использовать PipedReader.ready()
- Пример. реализация — java8_nio/io/Main04R
- ```
  final PipedWriter pw = new PipedWriter();
  final PipedReader pr = new PipedReader();
  final ReentrantLock sLock = new ReentrantLock();
  Runnable rOut = () -> {
      try {
          pw.connect(pr);
          Thread.sleep(10);
          String s = IOUtils.STRING_ENC; // UTF-8
          byte[] bbs = s.getBytes(Charset.forName("WINDOWS-1251"));
          String s2 = new String(bbs,Charset.forName("WINDOWS-1251"));            // bytes >> win-1251
          String s3 = new String(s.getBytes(Charset.forName("WINDOWS-1251")));    // UTF-8 >> win-1251
          pw.write(String.format("UTF-8:%n"));
          pw.write(s);
          pw.write(String.format("WINDOWS-1251:%n"));
          pw.write(s2);
          pw.write(String.format("UTF-8 >> WINDOWS-1251:%n"));
          pw.write(s3);
          pw.flush();
          synchronized (sLock) {
              sLock.wait();
          }
      } catch (InterruptedException | IOException e) {
          e.printStackTrace();
      }finally {
          try {

              pw.close();
          }catch (IOException e) {
              e.printStackTrace();
          }
      }
      System.out.println("pipedOut: closed");

  };
  Runnable rIn = () -> {
      try {
          char[] chars = new char[50];
          int len;
          while (!pr.ready()) {     // awaiting
              System.out.print(".");
              Thread.sleep(1);
          }
          System.out.println();
          while (pr.ready() && (len = pr.read(chars)) > 0) {  // блокирующее чтение Reader
              String s = new String(chars,0,len);
              System.out.printf("%s",s);
          }
          synchronized (sLock) {
              sLock.notifyAll();
          }
      } catch (InterruptedException | IOException e) {
          e.printStackTrace();
      } finally {
          try {
              pr.close();
          } catch (IOException e) {
              e.printStackTrace();
          }
      }
      System.out.println("pipedIn: closed");
  };
  ExecutorService exec = Executors.newFixedThreadPool(2);
  exec.execute(rOut);
  exec.execute(rIn);
  exec.shutdown();
  ```
  ```

StringReader

- StringReader класс работы со строками
  - свойства стандартный поток работы со строками
- Пример. реализация java8_nio/io/Main04R

```
            StringReader sr = null;
            br = null;
            try {
                String s = IOUtils.STRING_ENC;
                sr = new StringReader(s);
                IOUtils.readout(sr);
    // string reader
                int bSize = 200;
                System.out.printf(FORMAT,"CheckMark StringReader:");
                sr = new StringReader(s);
                IOUtils.checkMark((Reader)sr, false, s.length(), bSize);
                sr = new StringReader(s);
                IOUtils.checkMark((Reader)new BufferedReader(sr,bSize), true, s.length(), bSize);
    // buffered reader
                System.out.printf(FORMAT,"CheckMark BufferedReader(StringReader):");
                sr = new StringReader(s);
                IOUtils.checkMark((Reader)new BufferedReader(sr,bSize), false, s.length(), bSize);
                sr = new StringReader(s);
                IOUtils.checkMark((Reader)new BufferedReader(sr,bSize), true, s.length(), bSize);
            } finally {
                IOUtils.closeStream(br);
            }
    public static void checkMark(Reader r, boolean isExceeded, int size, int bSize) {
        if (!r.markSupported()) {
            System.out.println("mark() is not supported");
            return;
        }
        try {
            int limit = size;
            if (isExceeded) limit -= 10;  // делаем меньше
            r.read();  // read 2 bytes
            r.read();
            r.mark(limit);
            int len;
            char[] chars = new char[50];
            while (r.ready() && (len = r.read(chars)) > 0) {
                System.out.printf("%s", new String(chars, 0, len));
            }
            System.out.println("read:" + size + " limit:" + limit + " buffer:" + bSize);

            System.out.println();
            System.out.println("read : ok");
            r.reset();
            System.out.println("reset: ok");
        } catch (IOException e) {
            System.out.println("<< reset() Exception"+e+" >>");
        } finally {
            IOUtils.closeStream(r);
        }
    }
```

LineNumberReader

- LineNumberReader класс оболочка для потока Reader, позволяет получить номер строки
  - свойства есть метод номера строки LineNumberReader.getLineNUmber()
- Пример. реализация java8_nio/io/Main04R

```
            LineNumberReader lr = null;
            br = null;
            try {
                br = new BufferedReader(new FileReader(PATH + "result.txt"), 100);  // internal buffer
                lr = new LineNumberReader(br);
                IOUtils.readout(lr); // closes br
            } catch (
                    IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.closeStream(lr); // closes br
            }
```

FilterReader

- FilterReader абстрактный класс для потомков, пробрасывает все запросы встроенному потоку
  - свойства при создании потомка задать конструктор, переопределить методы фильтрации
  - известный потомок PushbackReader
- Пример. реализация анонимного класса на базе FilterReader **java8_nio/io/Main04R**

```
FilterReader fr = null;
br = null;
try {
    fr = new FilterReader(new FileReader(PATH + "result.txt")) {
        @Override
        public int read(char[] cbuf, int off, int len) throws IOException {
            len = super.read(cbuf, off, len);
            if (len > 0) {
                for (int i = 0; i < len ; i++) {
                 if(cbuf[i]=='*') cbuf[i] = '+';
                    if(cbuf[i]=='-') cbuf[i] = '/';
                }
            }
            return len;
        }
    };
    br = new BufferedReader(fr, 100);  // internal buffer
    IOUtils.readout(br);
} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(br);
}
```

PrintWriter

- PrintWriter класс печати символов, аналог PrintStream
  - свойства вывод форматированных строк на печать
- ВНИМАНИЕ. При работе с системными потоками НЕ ЗАКРЫВАТЬ PrintWriter, использовать flush()
- Пример. реализация **java8_nio/io/Main04R**

```
PrintWriter pw = null;
PrintWriter pwb = null;
FileReader fr = null;
br = null;
BufferedWriter bw = null;
try {
    fr = new FileReader(PATH + "result.txt");
    pw = new PrintWriter(System.out);
    char[] chars = new char[100];
    int len;
    while (fr.ready()) {
        len = fr.read(chars);
        if (len > 0) {
            pw.printf("%s", new String(chars, 0, len));
        }
    }
    pw.flush();  // DO NOT CLOSE
    br = new BufferedReader(new FileReader(PATH + "result.txt"), 100);  // internal buffer
    bw = new BufferedWriter(new FileWriter(PATH + "printw.txt"), 100);
    pwb = new PrintWriter(bw);
    String s;
    while ((s = br.readLine()) != null) {
        pwb.printf("%s%n", s);
    }
    pwb.close(); // flushes pwb, closes bw
    br.close();
    br = new BufferedReader(new FileReader(PATH + "printw.txt"), 100);  // internal buffer
    while ((s = br.readLine()) != null) {
        pw.printf("%s%n", s);
    }
    pw.flush();
} catch (IOException e) {
} finally {
    IOUtils.closeStream(br);
    IOUtils.closeStream(fr);
    }
```

PushbackReader

- PushbackReader класс который позволяет вернуть символ в поток
    - свойства используется для синтаксического анализа потока и подмены данных
- Пример. реализация **java8_nio/io/Main04R**
- ```
  PushbackReader pbr = null;
  br = null;
  try {
      br = new BufferedReader(new FileReader(PATH + "result.txt"), 100);  // internal buffer
      pbr = new PushbackReader(br, 10); // all bytes returned
      final StringBuilder sb = new StringBuilder();
      final char[] chars = new char[100];
      BiPredicate<String, Integer> p = (s1, v1) -> String.CASE_INSENSITIVE_ORDER
              .compare(new String(chars, 0, v1), s1.substring(1)) == 0;
      int len;
      int c;
      while ((c = pbr.read()) > 0) {
          switch (c) {
              default:
                  sb.append((char) c);
                  break;
              case '/':
                  int c2 = pbr.read();
                  int c3 = pbr.read();
                  if (c2 == '*' && c3 == '*') {
                      sb.append("<!>");
                      break;
                  } else {
                      sb.append('/');
                      pbr.unread(c2);
                      pbr.unread(c3);
                  }
                  break;
              case '*':
                  String s = "* created:"; // case insensitive
                  len = pbr.read(chars, 0, s.length() - 1);
                  if (p.test(s, len)) {
                      sb.append(">> Developed by");
                      break;
                  } else {
                      sb.append(s.charAt(0));
                      pbr.unread(chars, 0, len);
                  }
                  break;
          }
      }
      System.out.printf("%s", sb);

  } catch (
          IOException e) {
      e.printStackTrace();
  } finally {
      IOUtils.closeStream(pbr);
  }
  ```
-
  ```

Process Demo

- Process Demo — использование BufferedReader для перехвата системных потоков
  - потоки — перехватываются у процесса потоки System.out, System.err и System.in
  - работа — запускается процесс, в данном случае исполнение java io.IOUtils.class main()
  - системный ввод System.in читается через BufferedReader.in в поток BufferWriter.bw
  - так как BufferedReader блокирует ввод, используется ready() для прохода в цикле
- Пример. реализация **java8_nio/io/Main04R**

```
// BufferedReader Process
        System.out.printf(FORMAT, "BufferedReader Process:");
        IOUtils.process("java -cp out/production/java_nio io.IOUtils");
    public static void process(String cmd) { // processes
        boolean err = false;
        BufferedReader br = null;
        BufferedReader be = null;
        BufferedWriter bw = null;
        BufferedReader in = null;
        Scanner sc = null;
        Process p = null; // split string on words
        try {
            p = new ProcessBuilder(cmd.split(" ")).start(); // split string on words
            br = new BufferedReader(new InputStreamReader(p.getInputStream()));
            be = new BufferedReader(new InputStreamReader(p.getErrorStream()));
            bw = new BufferedWriter(new OutputStreamWriter(p.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(System.in));
//            sc = new Scanner(System.in);
            String s;
            boolean isExit = false;
            while (true) {
                while (br.ready() && (s = br.readLine()) != null) {
                    System.out.println(s);
                }
                while (be.ready() && (s = be.readLine()) != null) {
                    System.err.println(s);
                }
                if (isExit) {
                    break;
                }
                while (in.ready() && (s = in.readLine()) != null) {
                    bw.write(String.format("%s%n", s));
                    bw.flush();
                    if (s.equals("exit")) {
                        Thread.sleep(10);
                        isExit = true;
                    }
                }
//                if ((s = sc.nextLine()) != null) {  // waiting input
//                    if (s.equals("exit")) break;
//                    bw.write(String.format("%s%n", s));
//                    bw.flush();
//                }
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeStream(br);
            IOUtils.closeStream(be);
            IOUtils.closeStream(bw);
            IOUtils.closeStream(in);
        }
    }
    public static void main(String[] args) {
        System.out.println("JAVA8 IO Support");
        System.out.println("Java8 NIO Support");
        System.out.println("Type something('exit' to cancel):");
        Scanner in = new Scanner(System.in);
        String s;
        while ((s = in.nextLine()) != null) {
            if (s.contains("exit")) break;
            System.out.printf("process: %s%n", s);
        }
        System.out.println("process finished...");
    }
```

Java NIO

Java NIO differences

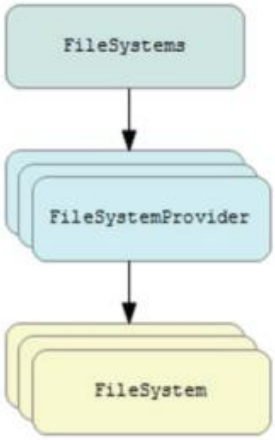
- Java NIO
- Java NIO differences ключевые отличия от IO
    - использует буферы в качестве источника или приемника данных
    - не блокирует обмен данными, если данные недоступны
    - обеспечивает произвольный доступ к полученным данным
    - предназначена для большого количества соединений с малым объемом данных
- Java NIO2 новые возможности
- обновление информации по селекторам прерываемым и асинхронным каналам

Java NIO Components

- Buffer обекты хранения данных JavaNIO
- Channel канал работает как проводник между буфером и IO services
- Sockets каналы для Socket
- Pipes объединение каналов приема и передачи в Pipe
- Selectors объекты выбора каналов
- Regex были введены как часть NIO, хотя используются для текстового поиска
- Charset были введены как механизм трансляции байт в символы и наоборот
- Formatter класс форматирования символьных данных

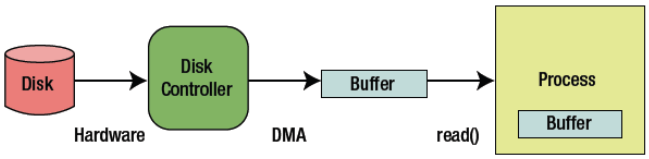
Java NIO Packages

- Java NIO Packages
    - java.nio классы Buffer, ByteBuffer, CharBuffer , ByteOrder
    - java.nio.channels классы Channel, Channels,
    - FileChannel, FileLock, MappedByteBuffer
    - SelectableChannel, Selector, SelectionKey, Pipe,
    - java.nio.charset классы Charset , CharsetDecoder, CharsetEncoder, CoderResult,
    - CodingErrorAction
    - java.nio.file классы Files, FileStore, FileSystem, FileSystems, LinkPermission
    - Paths static methods return Path by path string or URI
    - SimpleFileVisitor<T> simple visitor of files with default behavior
    - StandardWatchEventKinds defines the *standard* event kinds
    - FileSystemProvider service-provider class for file systems
    - FileTypeDetector file type detector for probing a file to guess its file type.
    - java.nio.file.attribute классы AttributeView, AclEntry, FileTime,PosixFilePermissions,
    - UserPrincipalLookupService


FileSystems
FileSystemProvider
FileSystem

Java NIO Components

- IO System система ввода вывода Java
  - Java IO является фундаментально byte oriented, поэтому работают только ByteBuffer
  - несовместима с DMA контроллером, который является block oriented
  - не имеет прямого обмена с DMA контроллером
  -
- Buffer ch6
  - процесс обмена данными между OS и JVM Buffer
  - передача данных от OS к системному буферу по DMA
  - передача данных от системного буфера в JVM User Process буфер операция read()



  - Buffer класс который OS заполнять JVM User Process буферы по DMA
  - ByteBuffer работает с Channel так как Java IO byte oriented
- Channel ch7
  - канал работает как проводник между буфером и OS DMA
  - FileChannel поддерживает FileLocking основа для работы с базами данных
  - MappedByteBuffer доступ к файлу как области памяти, исключая read(), write()
- Selector ch8
  - обмен данными может быть block oriented или stream oriented
  - block oriented чтение данных блоками, например из файла
  - stream oriented чтение с клавиатуры, мало данных, возможны паузы
  - селектор поддерживает концепцию сканирования и готовности в non-blocking mode
  - один Thread вместе с селектором работает с несколькими каналами по готовности
- Regex ch9
  - regex были введены как часть NIO, хотя используются для текстового поиска
- Charset были введены как механизм трансляции байт в символы и наоборот
- Formatter добавлены механизм форматирования

Buffer

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.nio классы Buffer, ByteBuffer, CharBuffer , ByteOrder
- Buffer объект хранения данных Java NIO два типа «nondirect» в JVM Heap Memory и «direct» в OS Memory
    - capacity емкость
    - limit позиция завершения валидных данных
    - position текущая позиция данных
    - mark позиция, к которой будет возвращен указатель position по reset()
    - 0 <= mark <= position <= limit <= capacity
    - nondirect буфер создается allocate(), имеет фоновый массив byte[] доступный через array()
- Методы
    - flip() limit = position, a position = 0, mark = discarded
    - ТОЛЬКО сбрасывает позицию и limit, НЕТ НИКАКОЙ инверсии указателей данных
    - clear() position = 0, данные объявляются невалидными, как бы очистка буфера
    - rewind() position = 0, данные остаются валидными, как бы перемотка в начало буфера
    - возврат многих методов Buffer, что позволяет создать цепочки
    - `buf.mark().position(2).reset()`
    - allocate() создает nondirect буфер HeapByteBuffer заданного размера в JVM Heap
    - минимум накладных расходов при создании или освобождении
    - более медленная работа при обмене данными
    - allocateDirect() создает буфер DirectByteBuffer заданного размера напрямую в памяти OS
    - серьезные накладные расходы при создании буфера за пределами JVM
    - обязанность освобождать буфер лежит на программисте, GC на работает вне JVM
    - максимально эффективная работа при обмене данными
    - wrap() обертка массива byte[] в ByteBuffer
    - order() задает order bytes в ByteBuffer
    - compact() копирует байты [position ...limit] в начало буфера и устанавливает position = 0
    - asReadOnlyBuffer() позволяет получить копию read only буфера
    - asLongBuffer() позволяет сразу вытащить Buffer нужного типа из ByteBuffer

Buffer. asReadOnlyBuffer ()

- asReadOnlyBuffer() получить shallow readonly copy Buffer
    - можно задать position(), flip(), limit(), read() как угодно
    - запись не разрешена, выдает Exception
- Пример. реализация **java8_nio/nio/Main01**

```
LongBuffer bLongReadOnly = bLong.asReadOnlyBuffer();
bLongReadOnly.flip();
NIOUtils.readout(bLongReadOnly);
try {
    bLongReadOnly.put(1, 1200);
} catch (ReadOnlyBufferException e) {
    System.out.printf("Buffer Exception:%s%n", e);
}
```

Buffer. allocateDirect ()

- allocateDirect() создает буфер DirectByteBuffer заданного размера напрямую в памяти OS
  - работа создать работоспособный код на базе allocate() nonblocking buffer в Heap JVM
  - заменить вызов allocate() на allocateDirect()
  - провести замеры производительности и расхода памяти

Buffer.order()

- order() задает order bytes в ByteBuffer
  - значения ByteOrder.BIG_ENDIAN, ByteOrder.LITTLE_ENDIAN
  - задается только в ByteBuffer метод ByteBuffer.order(ByteOrder)
  - позволяет изменить порядок байт и сконвертировать в методами getInt(), putInt()
- Пример. реализация

```
ByteBuffer bByte = ByteBuffer.allocate(size*8);
System.out.printf("orders  byte:%s int:%s long:%s float:%s double:%s%n",
        ByteBuffer.allocate(1).order(), IntBuffer.allocate(1).order(),
        LongBuffer.allocate(1).order(), FloatBuffer.allocate(1).order(),
        DoubleBuffer.allocate(1).order());
NIOUtils.longToByte(bByte,bLong);
bByte.order(ByteOrder.LITTLE_ENDIAN);            // read byte BE     reversed
NIOUtils.byteToLong(bByte,bLong);
NIOUtils.readout(bLong);                         // asLongBuffer
bByte.order(ByteOrder.BIG_ENDIAN);               // read byte LE      normal
LongBuffer byteLong = bByte.flip().asLongBuffer();
bLong.clear();
while(byteLong.hasRemaining())
    bLong.put(byteLong.get());
NIOUtils.readout(bLong);
```

Buffer.compact()

- compact() копирует байты [position ...limit] в начало буфера и устанавливает position = 0
  - ВНИМАНИЕ. Метод нужен когда запись в канал из буфера ПРЕРВАНА. Оставшиеся данные
  - копируются в начало буфера и добавляются к данным новой сессии чтения.
- Пример. реализация **java8_nio/nio/Main01B**

```
FileChannel in;
FileChannel out;
try {
    ByteBuffer bb = ByteBuffer.allocate(100);
    in = new FileInputStream(PATH + "result.txt").getChannel();
    out = new FileOutputStream(PATH + "result_out.txt").getChannel();
    while ((in.read(bb)) != -1) {
        bb.flip();
        System.out.println("flip    bb:" + bb.position() + " " + bb.limit());
        out.write(bb);
        bb.compact();
    }
    IOUtils.closeStream(in,out);  //  in.close() >> closes fIn  out.close() >> closes fout
    NIOUtils.readout(PATH + "result_out.txt");
    in = new FileInputStream(PATH + "result.txt").getChannel();
    out = new FileOutputStream(PATH + "result_out.txt").getChannel();
    while ((in.read(bb)) != -1) {
        bb.flip();
        System.out.println("flip    bb:" + bb.position() + " " + bb.limit());
        out.write(bb);
        bb.position(60); //  [position .. limit] > 0 write is not finished and there are bytes to copy
        bb.compact();
    }
    IOUtils.closeStream(in,out);  //  in.close() >> closes fIn  out.close() >> closes fout
    NIOUtils.readout(PATH + "result_out.txt");
} catch (IOException e) {
}
```

Buffer Multithreading

- MultiThread Buffer НЕБЕЗОПАСНЫ для многопоточного программирования
  - synchronized ОБЯЗАТЕЛЬНО использовать

Channel

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.nio.channels классы Channel, Channels,
    - FileChannel, FileLock, MappedByteBuffer
    - SelectableChannel, Selector, SelectionKey, Pipe,
- Channel это шлюз через который IO services получают доступ к буферам
    - буферы это оконечные устройства канала
    - устройства назначения это файлы, сокеты, программные модули которые передают данные с
    - помощью IO services в канал и через канал в оконечные устройства или буферы.
- Свойства
    - каналы эффективно передают данные между ByteBuffer и IO sercices
    - FileChannel являются THREAD SAFE в отличие от Buffers
- Методы
    - close() закрывает канал
    - isOpen() проверяет открыт канал или нет
    - write() метод канала с интерфейсом WritableChannel
    - read() метод канала с интерфейсом ReadableChannel
    - force(meta) форсирует запись изменений в файл, meta=true update атрибутов
- ВНИМАНИЕ. Дозапись изменений НЕ ГАРАНТИРУЕТСЯ для удаленных файлов

Channel Interfaces

- Channel Interfaces
    - интерфейс
    - WritableChannel.newInstance() создает канал типа writable byte
    - ReadableChannel.newInstance() создает канал типа readable byte
    - InterruptableChannel допускает прерывание обмена данными
    - проверка типа интерфейса реализуемого каналом через instanceof InterruptableChannel

Channel Classes

- Channel Classes классы которые работают с каналами
    - FileChannel
    - DatagramChannel,
    - FileChannel,
    - Pipe.SourceChannel,
    - SocketChannel
    -

RandomAccessFile Channel

- RandomAccessFile Channel Example
  - канал на базе RandomAccessFile двунаправленный
  - force() ОБЯЗАТЕЛЬНО после записи вызывать force() в примере РЕАЛЬНО работает
  - Charset запись строки в CharsetBuffer кодировки НЕ ТРЕБУЕТСЯ, символы ОРИГИНАЛЫ
  -
- ВНИМАНИЕ. При работе с CharsetBuffer символы пишутся ОРИГИНАЛЫ кодировки НЕ ТРЕБУЕТСЯ
- Пример. реализация обмена данными по FileChannel на базе RandomAccessFile **java8_nio/nio/Main02C**
- ```
  FileChannel fra = null;
  try {
      fra = new RandomAccessFile(PATH + "result_ra_test.txt", "rw").getChannel();
      long pos;
      System.out.printf("position: %d%n",(pos = fra.position()));
      System.out.printf("size    : %d%n",fra.size());
      ByteBuffer b = ByteBuffer.allocate(IOUtils.STRING_ENC.length()*2);
      b.asCharBuffer().put(IOUtils.STRING_ENC);                  // работает четко по символам Unicode
      fra.write(b);                                              // записать
      fra.force(true);                                           // точно записать
      System.out.printf("position: %d%n",fra.position());
      System.out.printf("size    : %d%n",fra.size());
      b.clear();
      fra.position(pos);              // start position
      fra.read(b);
      b.flip();
      while(b.hasRemaining()) {
          System.out.printf(Locale.ENGLISH,"%c",b.getChar());
      }
  } catch (IOException e) {
      e.printStackTrace();
  } finally {
      IOUtils.closeStream(fra); // closes FileInputStream, FileOutputStream, RandomAccessFile
  }
  ```
  ```

Channel IO Example

- Channel IO Example **java8_nio/nio/Main02C**
- Пример. реализация обмена данными по каналам
-
```
    private static void copy(ReadableByteChannel in, WritableByteChannel out) throws IOException {
        ByteBuffer b = ByteBuffer.allocateDirect(2048);
        ByteBuffer b2 = b.asReadOnlyBuffer();
        byte[] bytes = new byte[4];
        while (in.read(b) != -1) {
            b.flip();               // отсекаем
            out.write(b);           // пишем в выходной поток что есть
// check data
            if (b2.hasRemaining()) {
                b2.get(bytes);
                if (new String(bytes).equals("exit"))  return;
                b2.flip();
            }
            b.compact();            // если прервано сохраняем
        }
        b.flip();                   // дозапись если есть
        while (b.hasRemaining()) {  // выкачиваем остатки полностью
            out.write(b);
        }
    }
    private static void copyAlt(ReadableByteChannel in, WritableByteChannel out) throws IOException {
        ByteBuffer b = ByteBuffer.allocateDirect(2048);
        byte[] bytes = new byte[4];
        while (in.read(b) != -1) {          // получили данные в буфер
            b.flip();                       // отсекаем
            b.get(bytes);
            if (new String(bytes).equals("exit")) return;
            b.position(0);
            while (b.hasRemaining()) {      // пишем в выходной поток
                out.write(b);
            }
            b.clear();
        }
    }
    public static void main(String[] args) {
        System.out.printf(FORMAT, "Channel IO:");
        ReadableByteChannel in = null;
        WritableByteChannel out = null;
        try {
            in = Channels.newChannel(System.in);
            out = Channels.newChannel(System.out);
            System.out.println("copy: type than <Enter>('exit' to exit:");
            copy(in, out);
            System.out.println("copyAlt: type than <Enter>('exit' to exit:");
            copyAlt(in, out);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeStream(in, out);
        }
    }
```

Channel Scattering

- Channel Scattering работа одного канала с несколькими буферами
  - ScatteringByteChannel канал работы с несколькими буферами на чтение
  - read(ByteBuffer[]), read(ByteBuffer[], offset,len)
  - GatheringByteChannel канал работы с несколькими буферами на запись
  - write(ByteBuffer[]), write(ByteBuffer[], offset,len)
- Пример. реализация **java8_nio/nio/Main02C**
-
```
        ScatteringByteChannel sin = null;
        GatheringByteChannel gout = null;
        try {
            sin =  new FileInputStream(PATH + "result_i.txt").getChannel();
            gout =  new FileOutputStream(PATH + "result_o.txt").getChannel();
            ByteBuffer b5 = ByteBuffer.allocate(5);
            ByteBuffer b3 = ByteBuffer.allocate(3);
            ByteBuffer[] bbs = {b5, b3};
            sin.read(bbs);
            NIOUtils.readout(b5);
            NIOUtils.readout(b3);
            b5.rewind();
            b3.rewind();
            bbs[0] = b3;
            bbs[1] = b5;
            gout.write(bbs);
//              ((FileChannel) gout).force(true);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeStream(sin, gout);
        }
```

FIleChannel

- FileChannel — канал работы с файлами, в отличие от Buffer является Thread Safe
  - FileInputStream.getChannel() readOnly FileChannel
  - FileOutputStream.getChannel() writeOnly FileChannel
  - RandomAccessFile.getChannel read and write FileChannel
- FileChannel
  - position(int) задать текущую позицию, используется при чтении после записи
  - read(Buffer) читает данные в буфер, возвращает число байт, -1 если пусто
  - write(Buffer) записывает данные из буфера, возвращает число байт, -1 если пусто
  - size() размер подключенного к каналу файла
  - truncate(size) сократить файл до размера size
  - size() возвращает размер lock
- ВНИМАНИЕ. FileChannel является THREAD SAFE в многопоточном программировании
- Пример. реализация RandomAccessFile **java8_nio/nio/Main02C**

```
FileChannel fr = null, fw = null, frw = null;
try {
    fr = new FileInputStream(PATH + "result.txt").getChannel();
    fw = new FileOutputStream(PATH + "result_w.txt").getChannel();
    frw = new RandomAccessFile(PATH + "result_ra.txt", "rw").getChannel();

    ByteBuffer b = ByteBuffer.allocate(200);
    while (fr.read(b) != -1) {          // получили данные в буфер
        b.flip();                       // отсекаем
        while (b.hasRemaining()) {      // пишем в выходной поток
            fw.write(b);
            b.flip();
            frw.write(b);
        }
        b.compact();
        b.clear();
    }
    frw.force(true);  // force store changes to file
    frw.position(0);
    while (frw.read(b) != -1) {          // получили данные в буфер
        b.flip();                       // отсекаем
        System.out.printf("%s", new String(b.array(), Charset.defaultCharset()));
        b.clear();
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(fr, fw, frw); // closes FileInputStream
}
```

FileLocking

- FileLocking блокировка файлов, крайне важна при работе с транзакциями
  - типы exclusive lock single lock на запись, больше никто не может блокировать
  - shared lock multi lock на чтение, shared read access не разрешен exclusive lock
  - свойства блокировки работают только на базе файлов
  - блокировки работают на уровне процессов (application), а не thread или channel
  - если shared lock не поддерживается OS, она становится exclusive lock
  -
  - применение exclusive lock используется для update файла,shared lock blocked и ждут завершения
  - shared lock для чтения несколькими участниками одновременно
- FileLock методы
  - метод lock() получить exclusive lock на конкретный файл
  - с параметрами получить exclusive/shared lock на регион (pos,len)
  - если файл или регион уже заблокированы, блокирует приложение
  - trylock() попытаться получить exclusice lock на файл без блокирования
  - с параметрами попытаться получить exclusive/shared на регион
  - если файл или регион уже заблокированы, просто возвращает null
  - channel вернуть канал, связанный с заблокированным файлом
  - close() снять lock, вызывает метод release()
  - isShared() возвращает тип lock
  - isValid() возвращает валидность, lock valid пока не закрыт сам lock или файл
  - overlaps() показывает есть ли наложение регионов shared lock
  - position() старт региона lock
  - release() освобождение lock
- Пример. реализация стандартная конструкция работы с Lock **java8_nio/nio/Main02C**

```
FileLock lock = null;
try {
    in = new FileInputStream(PATH+"result.txt").getChannel();
    out = new FileOutputStream(PATH+"result_o.txt").getChannel();

} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(in, out);
    try {
        if (lock != null)
            lock.release();
    }catch (IOException e) {
        e.printStackTrace();
    }
}
```

File Lock Example

- File Lock Example — пример параллельной работы двух приложений над одним файлом
  - свойства — запуск через Runtime.getRuntime().exec()
  - — используется вызов call чтобы увидеть результаты завершения
  - write — версия пишет в файл данные, использует exclusive lock
  - read — версия читает его, использует shared lock
  - — каждая из версий, захватив файл не пускает другое приложение до завершения
  - mulitapp — для чтения в 20 приложений, задержка в 20ns после чтения, позволяет писать
  - — без задержки идет постоянная блокировка shared и write не может писать вообще
- Пример. реализация **java8_nio/nio/Main02C**

```
try {
    Runtime.getRuntime()
                .exec("cmd /c start call java -cp ./out/production/java_nio nio.fileLock.MainLock w");
    Runtime.getRuntime()
                .exec("cmd /c start call java -cp ./out/production/java_nio nio.fileLock.MainLock" );
}catch (IOException e) {
    e.printStackTrace();
}
public class MainLock {
    private static final int MAX_QUERIES = 10000;
    private static final int MAX_UPDATES = 10000;
    private static final int RECORD_LEN = 16;

    private FileChannel fc;
    private RandomAccessFile raf;
    private ByteBuffer bByte;
    private IntBuffer bInt;
    private int counter;

    public MainLock() {
        this.bByte = ByteBuffer.allocate(RECORD_LEN);
        this.bInt = bByte.asIntBuffer();
        this.counter = 1;
    }

    public static void main(String[] args) {
        new MainLock().run(args.length > 0);
        System.exit(0);
    }

    private void query(FileChannel fc) throws IOException ,InterruptedException{
        for (int i = 0; i < MAX_QUERIES; i++) {
            FileLock lock = fc.lock(0, RECORD_LEN, true);
            try {
                bByte.clear();
                fc.read(bByte, 0);
                int a = bInt.get(0);
                int b = bInt.get(1);
                int c = bInt.get(2);
                int d = bInt.get(3);

                System.out.printf("Reading: %d %d %d %d%n", a, b, c, d);
                if (a * 2 != b || a * 3 != c || a * 4 != d) {
                    System.out.printf("Read Error%n");
                    return;
                }
            } finally {
                lock.release();
//                Thread.sleep(0,20); // for 10 apps
            }
        }
    }
```

- Пример. реализация продолжение **java8_nio/nio/Main02C**

```
    private void update(FileChannel fc) throws IOException {
        for (int i = 0; i < MAX_UPDATES; i++) {

            FileLock lock = fc.lock(0, RECORD_LEN, false);
            try {
                bInt.clear();
                int a = counter;
                int b = counter * 2;
                int c = counter * 3;
                int d = counter * 4;
                System.out.printf("writing: %d %d %d %d%n", a, b, c, d);
                bInt.put(a);
                bInt.put(b);
                bInt.put(c);
                bInt.put(d);
                counter++;
                bByte.clear();              // actually position=0 limit=capacitance()
                fc.write(bByte, 0); // write all the buffer
            } finally {
                lock.release();
            }
        }
    }

    private void run(boolean isWrite) {
        raf = null;
        fc = null;
        try {
            raf = new RandomAccessFile(PATH + "result_lock.txt", isWrite ? "rw" : "r");
            fc = raf.getChannel();

            if (isWrite) update(fc);
            else query(fc);

        } catch (IOException |InterruptedException e) {
            e.printStackTrace();
        } finally {
            try {
                if (fc != null) fc.close();
                if (raf != null) raf.close();

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

Channel Bulk Transfer

- Channel Bulk Transfer пересылка байт между каналами bulk transfer для файлов
  - методы transferFrom() пересылка из канала ReadableFileChannel
  - transferTo() пересылка в канал WritableFileChannel
- Пример. реализация

```
in = null;
ReadableByteChannel rin = null;
WritableByteChannel wout = null;
try {
    in = new FileInputStream(PATH + "result.txt").getChannel();
    wout = NIOUtils.newInstance(System.out);  // Channels.newChannel() that does not close System.out
    in.transferTo(0, in.size(), wout); // из файла в канал WritableByteChannel
    wout.close();

    rin = new FileInputStream(PATH+"audio.wav").getChannel();
    wout = new FileOutputStream(PATH+"audio_bulk.wav").getChannel();
    ((FileChannel) wout).transferFrom(rin,0,((FileChannel) rin).size());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.closeStream(in,rin,wout);
}
```

File Map to Memory

- File Map to Memory — MappedByteBuffer отображение региона файла на память
  - свойства — эффективный способ обмена данными с файлом
  - — исключает IO service вызовы, обмен идет напрямую
  - позволяет — резко ускорить обмен за счет работы с памятью
  - режимы — открытия MappedByteBuffer
  - — READ_ONLY — чтение в память, запись в буфер вызывает Exception
  - — READ_WRITE — чтение и запись из памяти в файл
  - — PRIVATE — чтение в память, запись в буфер, но в файл запись не передается
  - unMap() — запись в файл который был Mapped даже после закрытия потоков НЕВОЗМОЖНА
  - — нужно обязательно чтобы System.gc() собрал буфер MappedByteBuffer и его клоны
  - CharBuffer — работает с UTF-16 именно поэтому ВОЗМОЖЕН reverse
  -
- ВНИМАНИЕ. Нельзя просто сделать unMap() пока Garbage Collector сам не соберет данные
- ВНИМАНИЕ. Невозможно String reverse() без CharBuffer, обычный текст содержит символы РАЗНОГО размера
- File Map to Memory Methods
  - метод — map() — отображает регион файла на системную память
  - — isReadOnly() — проверяет можно ли изменять файл для map()
  - — load() — загружает содержимое файла в память
  - — isLoaded() — показывает, все содержимое файла было загружено в память
  - — force() — форсирует запись изменений,
- ВНИМАНИЕ. ИСПОЛЬЗОВАТЬ MappedByteBuffer.load(), а не FileChannel.load(),channel может не знать о map
- Пример. реализация реверса содержимого файла через MappedByteBuffer с кодировкой и asCharBuffer
-

```
        FileChannel.MapMode mapMode = FileChannel.MapMode.READ_WRITE;
        FileChannel fc = null;
        RandomAccessFile raf = null;
        try {
            raf = new RandomAccessFile(PATH + "result_random.txt", "rw"); // make UTF-16
            raf.setLength(0);                           // clear file
            fc = raf.getChannel();
            String s = Arrays.stream(NIOUtils.STRINGS_ENC).collect(Collectors.joining(String.format("%n")));
            ByteBuffer b = ByteBuffer.allocate(s.length() * 2);
            b.asCharBuffer().put(s);
            fc.write(b);
            fc.force(true);
            fc.close();
            fc = new RandomAccessFile(PATH + "result_random.txt", "rw").getChannel();// read  UTF-16
            long size = fc.size();
            System.out.printf("Size: %d%n", size);
            MappedByteBuffer mb = fc.map(mapMode, 0, size);
            CharBuffer cb = NIOUtils.readMappedBuffer(mb);
            for (int i = 0; i < cb.limit() / 2; i++) {// reverse
                char c = cb.get(i);
                char c2 = cb.get(cb.limit() - i - 1);
                cb.put(i, c2);
                cb.put(cb.limit() - i - 1, c);
            }
            cb.flip();
            System.out.printf(FORMAT, "Check Character Buffer:");
            while (cb.remaining() > 0) {
                System.out.printf("%c", cb.get());
            }
            System.out.printf("%n");
            mb = null;                                  // garbage collect to UNMAP
            cb = null;
            System.gc();
        } catch (IOException e) {
        } finally {
            IOUtils.closeStream(fc, raf);
        }
    }
```

Sockets

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.nio.channels SocketChannel, ServerSocketChannel, DatagramChannel
    - SocketChannel selectable канал для stream oriented connecting socket
    - ServerSocketChannel selectable канал для stream oriented listening socket
    - DatagramChannel selectable канал для datagram socket
    - свойства являются Thread Safe в отличие от Buffers
    - работают в blocking и nonblocknig режимах
    -
- Методы
    - socket() получить peer socket нужного типа
    - getChannel() получить объект связанного с socket Channel
    - можно получить только от объекта socket().getChannel()
    - нельзя от конструктора, new SocketChannel().getChannel() даст null
    - configureBlocking() задает режим работы SocketChannel, blocking или non-blocking mode
    - isBlocking() текущий режим работы
    - blockingLock() получить synchronized объект на базе которого осуществляется блокировка
    - только Thread владелец blockingLock() может менять режим SocketChannel
    -

SocketChannel

- SocketChannel selectable канал для stream oriented connecting socket
    - свойства ведет себя как клиент в TCP/IP, посылает запросы серверу
    - может работать в non-blocking и blocking режимах
- Методы
    - open() открыть Socket Channel, то есть клиентскую часть
    - с параметрами открывает сразу с привязкой к адресу порту
    - socket() получить Socket связанный с Socket Channel
    - connect() создать запрос соединения к серверу ServerSocketChannel
    - для локального ServerSocket выдает true сразу,
    - для удаленного ServreSocket постоянно вызывать finishConnect() пока не даст true
    - finishConnect() метод для установления соединения с удаленным ServerSocket
    - вызывается пока не выдаст true
    - isConnectionPending() выдает результат установления соединения с удаленным сервером
    - isConnected() выдает результат если Socket открыт и соединение установлено
- Процедура установления соединения
    - open() создать Socket, этот же объект возвращает SocketChannel.socket()
    - Socket.getChannel() возвращает SocketChannel
    - connect() подключить Socket к SocketChannelозволяет
- Charset ByteBuffer to CharBuffer можно сделать двумя способами
    - string String = new String(byte[],Charset,forName())
    - decode CharBuffer = Charset.decode(ByteBuffer)
- ВНИМАНИЕ. Перекодировку можно сделать либо методом new String() либо Charset.decode()
- Напрямую использовать ByteBuffer можно ТОЛЬКО если нет Unicode символов
- Пример. реализация смотреть ниже в коде SocketChannel **java8_nio/nio/Main03S**

SocketChannel

- Процедура
  - открытие open() >> connect() >> finishConnect()
  - чтение read(b) >> b.flip() >> b.get() >> b.clear >> close()
- Пример. реализация **java8_nio/nio/Main03S**
- ```
  public class MainSocket {
      public static void main(String[] args) {
          int port = 9998;
          if (args.length > 0) {
              try {
                  port = Integer.parseInt(args[0]);
              } catch (NumberFormatException e) {
                  e.printStackTrace();
                  return;
              }
          }

          System.out.printf(FORMAT, "SocketChannel( assert: java -ea):");
          SocketChannel sc = null;
          try {
              sc = SocketChannel.open();
              sc.configureBlocking(false);                              // nonblocking mode
              InetSocketAddress inetAddr = new InetSocketAddress("localhost", port);
              sc.connect(inetAddr);
  //            SocketChannel sc2=SocketChannel.open(new InetSocketAddress("localhost",9999)).
              String s = "Remote address: " + sc.getRemoteAddress();
              System.out.printf("Waiting finish connection at %s%n", s);
              while (!sc.finishConnect()) {
                  System.out.print(".");
                  try {
                      Thread.sleep(100);
                  } catch (InterruptedException e) {
                      e.printStackTrace();
                  }
              }

              ByteBuffer b = ByteBuffer.allocate(200);

              int len;
              while ((len = sc.read(b)) >= 0) {  // data read from socket
  //                if (len == 0) continue;
                  b.flip();
  // string
  //                byte[] bytes =  new byte[len];
  //                b.get(bytes);
  //                System.out.printf("%s",new String(bytes,Charset.forName("KOI8-R")));
  // char buffer
  //                CharBuffer c = Charset.forName("KOI8-R").decode(b);
  //                while (c.hasRemaining()) {
  //                    System.out.printf("%c", c.get());
  //                }
  // no encoding
                  while (b.hasRemaining()) {
                      System.out.printf("%c", (char)b.get());
                  }
                  b.clear();
              }
              sc.close();

          } catch (IOException e) {
              e.printStackTrace();
          } finally {
              IOUtils.closeStream(sc);
          }

          System.out.println("Socket finished...");
      }
  }
  ```
-
  ```

ServerSocketChannel

- ServerSocketChannel selectable канал для stream oriented listening socket
  - свойства ведет себя как сервер в TCP/IP, слушает клиентов и отвечает на вызовы
  - может работать в non-blocking и blocking режимах
- ВНИМАНИЕ. Run COMPOUND ServerSocket andSocket запуск одновременно нескольких приложений
- Методы
  - open() открыть server socket channel без привязки адрес порт, то есть серверную часть
  - socket() получить ServerSocket объект связанный с ServreSocketChannel
  - accept() принять соединение,вернуть Socket Channel,работающий в режиме blocking
  - если соединения нет non-blocking возвращает null
  - если соединения нет, blocking блокирует thread до соединения or Exception
- Процедура установления соединения
  - open() создает ServerSocketChannel instance неподключенным ServerScoket
  - socket() вернуть ServerSocket object
  - bind() подключает ServerSocket к адресу или порту
  - accept() принимает входящее соединение, либо возвращает ServerSocketChannel
  - либо возвращает null или блокирует Thread до соединения
  - альтернатива
  - accept() вызвать метод у ServerSocket >> socket().accept()
  - но при этом метод будет всегоа в blocking mode
- Пример. реализация ServerSocketChannel **java8_nio/nio/Main03S**
- ```
  public class MainServerSocket {
      public static void main(String[] args) {
          int port = 9999;
          if (args.length > 0) {
              try {
                  port = Integer.parseInt(args[0]);
              } catch (NumberFormatException e) {
                  return;
              }
          }
          System.out.printf(FORMAT, "SocketChannel( assert: java -ea):");
          ServerSocketChannel ssc = null;
          SocketChannel sc = null;
          try {
              ssc = ServerSocketChannel.open();
              ssc.socket().bind(new InetSocketAddress(port));     // port:9999
              ssc.configureBlocking(false);                        // nonblocking mode
              String s = String.format("Local address: %s%n", ssc.socket().getLocalSocketAddress());
              System.out.printf("Server started at %s", s);
              ByteBuffer b = ByteBuffer.wrap(s.getBytes(Charset.forName("UTF-8")));
              while (true) {
                  sc = ssc.accept();
                  if (sc != null) {
                      System.out.printf("%nReceived connection from: %s%n",
                              sc.socket().getRemoteSocketAddress());
                      b.rewind();
                      sc.write(b);            // запись из буфера в SocketChannel
                      sc.close();             // закрыть сразу
                  }else {
                      System.out.print(".");
                      Thread.sleep(500);
                  }
              }
          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
          } finally {
              IOUtils.closeStream(ssc, sc);
          }
          System.exit(0);
      }
  }
  ```
  ```

DatagramChannel

- DatagramChannel канал работает с данными по протоклу UDP/IP
    - свойства канал НЕ ГАРАНТИРУЕТ доставку Datagram даже если подтверждена отправка
    - канал не может работать с разными адресами после установления соединения
    - канал может быть сервером и клиентом, двухсторонним, то есть listening и request
- ВНИМАНИЕ. Datagram разбивается на пакеты, и если пакет потерян и на приеме не могут собрать
    - datagram и она ОТБРАСЫВАЕТСЯ без уведомления
- Методы
    - open() открывает Datagram Channel
    - socket() возвращает Datagramm Scoket object
    - connect(addr) подключиться к серверу Datagram по адресу
    - disconnect() отключить channel socket, не влияет на запущенные операции чтения и записи
    - receive() принять datagram по каналу, НЕ ТРЕБУЕТ connection to Datagram Channel
    - send() отправить datagram по каналу, НЕ ТРЕБУЕТ connection to Datagram Channel
    - read() читает datagram по каналу, требует Datagram Channel
    - write() записывает datagram по каналу, требует Datagram Channel
- Процедура
    - open() получить DatagramChannel instance
    - socket() получить DatagramSocketInstance
    - bind(addr) на стороне сервера подключить к адресу listener
    - connect(addr) на стороне клиента подключиться к серверу по адресу
    -
- Пример. реализация **java8_nio/nio/Main03S**
    - запуск можно запустить двумя способами combined, realtime
    - combined Edit Configurations >> добавить >> DatagramServer, DatagramClient
    - runtime Runtime.getRuntime().exec() по умолчанию
    - cmd /c start java ... запуск с закрытием cmd
    - cmd /c start call java ... запуск выход на комадную строку
    - DatagramServer запускается с параметрами 9996 [/b,/n]
    - 9996 номер порта
    - /b blocking mode /n nonblocking mode
    - DatagramClient запускается с параметрами 9996 [MSFT,MSBT] [/s,/w] [/b,/n]
    - 9996 номер порта
    - MSFT символ ключ
    - /s send() без connect() /w write() с connect()
    - /b blocking mode /n nonblocking mode

```
// DatagramChannel
        System.out.printf(FORMAT, "Datagram Channel:");
        try {
            Runtime.getRuntime().exec("cmd /c start call java -ea -cp " +
                    "out/production/java_nio nio.socket.DatagramServer 9996 /b");

            Runtime.getRuntime().exec("cmd /c start call java -ea -cp " +
                    "out/production/java_nio nio.socket.DatagramClient 9996 MSFT /s /b");

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

-

Pipes

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.nio.channels Pipe, Pipe.SinkChannel, Pipe.SourceChannel
    - Pipe пара каналов, которые образуют однонаправленный Pipe
    - Pipe.SinkChannel канал, представляет часть Pipe куда идет запись
    - Pipe.SourceChannel канал, представляет часть Pipe откуда идет чтение
    - свойства работают только внутри JVM, не могут передавать данные вне JVM
    - идеальные для создания producer/consumer заданного типа канала
- Методы
    - open() открыть новый Pipe
    - source() получить Pipe source объект
    - sink() получить Pipe sink объект
- Пример. реализация Pipe на базе каналов ReadableChannel и WritableChannel
- ```
  final int PIPE_SIZE = 10;
  final int PIPE_LIMIT = 3;
  final Pipe pipe;
  final Random rnd = new Random();
  try {
      pipe = Pipe.open();
      Runnable senderPipe = () -> {
          try {
              WritableByteChannel pSinkChannel = pipe.sink();
              ByteBuffer b = ByteBuffer.allocate(PIPE_SIZE);
              for (int i = 0; i < PIPE_LIMIT; i++) {      // 3 раза по 10 значений
                  b.clear();
                  for (int j = 0; j < PIPE_SIZE; j++) {
                      b.put((byte) rnd.nextInt(256));
                  }
                  b.flip();
                  while (pSinkChannel.write(b) > 0) ; // пока буфер не будет записан
              }
              pSinkChannel.close();
          } catch (IOException e) {
              e.printStackTrace();
          }
          System.out.printf("Send pipe closed%n");
      };
      Runnable receiverPipe = () -> {
          try {
              ReadableByteChannel pSourceChannel = pipe.source();
              ByteBuffer b = ByteBuffer.allocate(PIPE_SIZE);
              Thread.sleep(100);
              while (pSourceChannel.read(b) >= 0) {
                  b.flip();
                  while (b.hasRemaining()) {
                      System.out.printf("%d ", (byte) (b.get() & 255));
                  }
                  b.clear();  //clear buffer
              }
              pSourceChannel.close();
          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
          }
          System.out.printf("%nReceiver pipe closed%n");
      };
      ExecutorService exec = Executors.newFixedThreadPool(2);
      exec.execute(senderPipe);
      exec.execute(receiverPipe);
      exec.shutdown();                                        // normal completion
      if (exec.awaitTermination(1000, TimeUnit.MILLISECONDS)) {
          System.out.printf("Termination: true%n");
      } else {
          System.out.printf("Termination: false%n");
      }
  } catch (IOException | InterruptedException e) {
      e.printStackTrace();
  }
  ```
  ```

Selectors

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.nio.channels SelectableChannel, Selector, SelectionKey,
    - SelectableChannel абстрактный класс позволяет каналу multiplexed by Selector
    - потомки AbstractSelectableChannel базовый класс selectable каналов
    - потомки DatagramChannel, Pipe.SinkChannel, Pipe.SourceChannel,
    - ServerSocketChannel, SocketChannel
    - Selector абстрактный класс используется для multiplex SelectableChannel каналов
    - потомки AbstractSelector базовый класс для селекторов
    - SelectionKey абстрактный класс представляет токен для регистрации канала в селекторе
- Selectors
    - созданы для работы с nonblocking каналами с использованием POSIX механизмов OS
- ВНИМАНИЕ. Основные достоинства селекторов это НЕ НАДО ЖДАТЬ данных, НЕ НАДО ПРОВЕРЯТЬ данные
    - используют встроенные OS POSIX процедуры проверки канала на наличие и готовность данных
    - наличие данных говорит о том, что канал что то имеет
    - готовность данных говорит о том, что данные проверены и это то, что нужно
    - свойства работают только с nonblocking каналами котрые наследуют SelectableChannel
    - высокая производительность и масштабирование приложения с каналами
- Методы
    - open() создать Selector
    - register() зарегистрировать канал в объекте Selector,
    - параметр ops это bit set поддерживаемых interest и ready операций
    - OP_ACCEPT server socket channel ready to accept connection
    - OP_CONNECT channel ready to complete connection
    - OP_READ channel ready for reading
    - OP_WRITE channel ready for writing
    - unregister interestops(int) задать новые операции или 0 для полного unregister
    - register() с другими ops
    - select() получить Set<SelectionKeys> каналов, который сработали по interest, ready
    - select() blocking метод, блокирует Thread пока не будет результата
    - select(long) метод ждет timeout и выходит либо по результату, либо по timeout
    - selectNow() nonblocking метод, выход сразу с результатом или без него
- SelectionKeys методы
    - selector() возвращает Selector в котором зарегистрирован данный токен
    - channel() возвращает Channel канал, к которому привязан данный SelectionKey
    - attachment() возвращает объект, идентификатор канала, задается как параметр в register()
- Процедура
    - создать Selector.open() создать Selector
    - создать канал open(), newInstance() или getChannel() создать или получить Channel
    - задать режим configureBlocking(false) задать режим nonblocking иначе выдаст Exceptption
    - регистрировать register(Selector, devOps) зарегистрировать Channel в Selector
    - использовать select() получить набор SelectionKeys, перебрать итератором и отработать каналы
- ВНИМАНИЕ. Когда канал пуст но данные могут быть он возвращает 0, когда канал пуст и данных не будет -1
- если канал пуст, нужно либо закрыть канал, либо разрегистрировать ключ
- Пример. реализация Selector с Pipe.SinkChannel, Pipe.SourceChannel **java8_nio/nio/Main04S**

Selector with Pipe Channels

- Selector with Pipe Channels — реализация обмена
  - UserPipe — класс пользователя имеет поля
    - mPW — Pipe.SinkChannel — канал для записи в него сгенерированных данных
    - mPR — Pipe.SourceChannel — канал из которого данные читаются и выводятся на экран
    - exec — ExeciterService — fixedThreadPool в один поток
    - senderPipe — Runnable — объект реализует запись данных в mPW
    - receiverPipe — Runnable — объект реализует чтение и вывод на экран из канал mPR
  - Selector — selector — основной Selector который регистрирует 4 объекта Pipe
  - Pipe[] — pipes — массив Pipe который обеспечивают перекачку данных по 4 Pipe
    - p[0,1] — Pipe — запускает senderPipe и в Thread пишет данные в mPW, регистрирует mPR
    - p[2,3] — Pipe — запускает receiverPipe и в Thread читает данные из mPR, регистрирует mPW
- Процедура
  - read — запущенный в потоке Runnable senderPipe пишет данные в канал UserPipe.mPW
    - зарегистрированный в Selector канал UserPipe.mPR работает с методом select()
    - данные из User.mPR, читае статический метод UserPipeUtils.readChannel(mPR)
  - eof — select() постоянно срабатывает на чтение канала, и только при получении read(b)=-1
    - пользователь обязан САМОСТОЯТЕЛЬНО закрыть канал mPR или разрегистрировать ключ
  - write — запущенный в потоке Runnable receiverPipe читает данные из канала UserPipe.mPR
    - зарегистрированный в Selector канал UserPipe.mPW работает с методом select()
    - статический метод UserPipeUtils.writeChannel(mPW) пишет сам данные в канал
  - eof — select() постоянно срабатывает на запись в канал, пользователь определяет конец записи,
    - пользователь обязан САМОСТОЯТЕЛЬНО закрыть канал mPW или разрегистрировать ключ
- Пример. реализация класс UserPipe **java8_nio/nio/Main04S**
- ```
  public class UserPipe implements Closeable {
      static final int PIPE_LIMIT = 2;
      static final int PIPE_SIZE = 3;
      static final int RANGE_ID = 128;
      private static final int PIPE_DELAY = 500;
      private static final Random rnd = new Random();

      private Pipe mPipe;
      private int mId;
      private int mDelay;
      private int mLimit;
      private Pipe.SinkChannel mPW;
      private Pipe.SourceChannel mPR;
      private ExecutorService exec;

      public UserPipe(int id, int delay) {
          if (id < 0 || id > RANGE_ID) throw new IllegalArgumentException();
          try {
              this.mPipe = Pipe.open();
              this.mId = id;
              this.mPW = mPipe.sink();
              this.mPR = mPipe.source();
              this.mDelay = delay;
              this.mLimit = PIPE_LIMIT;
              exec = Executors.newFixedThreadPool(1);

          } catch (IOException e) {
              e.printStackTrace();
              this.mPipe = null;
              this.mPW = null;
              this.mPR = null;
              this.mId = -1;
              this.mDelay = 0;
          }
      }
  ```
  ```

- Пример. реализация класс UserPipe (продолжение) java8_nio/nio/Main04S
-
```
public int getId() {
        return mId;
    }
    public int getDelay() {
        return mDelay;
    }
    public Pipe.SinkChannel getPW() {
        return mPW;
    }
    public Pipe.SourceChannel getPR() {
        return mPR;
    }
    public int decLimit() {
        if (mLimit == 0) return -1;
        mLimit--;
        return mLimit;
    }
    public void runW() {
        if (((ThreadPoolExecutor) exec).getActiveCount() > 0) return null;
        exec.execute(senderPipe);
        exec.shutdown();
    }
    public void runR() {
        if (((ThreadPoolExecutor) exec).getActiveCount() > 0) return null;
        exec.execute(receiverPipe);
        exec.shutdown();
    }
    private final Runnable senderPipe = () -> {
        try {
            ByteBuffer b = ByteBuffer.allocate(PIPE_SIZE);
            for (int i = 0; i < PIPE_LIMIT; i++) {      // 10 раз по 3 значения
                b.clear();
                for (int j = 0; j < PIPE_SIZE; j++) {
                    byte c = (byte) (rnd.nextInt(RANGE_ID) + mId);
                    b.put(c);
                    System.out.printf("wt%03d:%d ", mId, c);
                }
                System.out.printf("%n");
                b.flip();
                while (mPW.write(b) > 0) ;               // пока буфер не будет записан
                Thread.sleep(mDelay);                  // 500ms
            }
            mPW.close();
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
        System.out.printf("wt%03d:send pipe closed%n", mId);
    };
    private final Runnable receiverPipe = () -> {
        int len = -1;
        try {
            ByteBuffer b = ByteBuffer.allocate(UserPipe.PIPE_SIZE);
            while ((len = mPR.read(b)) > 0) {
                b.flip();
                while (b.hasRemaining()) {
                    System.out.printf("rt%03d:%d ", mId, (byte) (b.get() & 255));
                }
                b.clear();
                System.out.printf("%n");
            }
        } catch (IOException e) {
            e.printStackTrace();
            len = -1;
        }
        System.out.printf("rt%03d:receiver pipe closed%n", mId);
    };
    @Override
    public String toString() {
        return "[" + mId + "]";
    }
    @Override
    public void close() throws IOException {
        if (mPW != null && mPW.isOpen()) mPW.close();
        if (mPR != null && mPW.isOpen()) mPR.close();
    }
}
```

- Пример. реализация класс UserUtils статические методы работающие с Selector **java8_nio/nio/Main04S**
- ```
  public class UserPipeUtils {
      public static void registerReadChannel(Selector selector, UserPipe p) throws IOException {
          p.runW();                           // write in Thread, read in Selector
          p.getPR().configureBlocking(false);
          p.getPR().register(selector, SelectionKey.OP_READ, p);  // OP_ACCEPT, OP_WRITE not allowed
      }
      public static void registerReadChannels(Selector selector, UserPipe[] pipes,
                                              int start, int end) throws IOException {
          if (selector == null || pipes == null ||
                  pipes.length < end ) throw new IllegalArgumentException();
          for (int i = start; i < end; i++) {
              registerReadChannel(selector, pipes[i]);
          }
      }
      public static void registerWriteChannel(Selector selector, UserPipe p) throws IOException {
          p.runR();                           // write in Thread, read in Selector
          p.getPW().configureBlocking(false);
          p.getPW().register(selector, SelectionKey.OP_WRITE, p);  // OP_ACCEPT, OP_READ not allowed
      }
      public static void registerWriteChannels(Selector selector, UserPipe[] pipes,
                                               int start, int end) throws IOException {
          if (selector == null || pipes == null ||
                  pipes.length < end) throw new IllegalArgumentException();
          for (int i = start; i < end; i++) {
              registerWriteChannel(selector, pipes[i]);
          }
      }
      public static int writeChannel(Pipe.SinkChannel pW, UserPipe p) {
          final Random rnd = new Random();
          int len = -1;
          try {
              ByteBuffer b = ByteBuffer.allocate(UserPipe.PIPE_SIZE);
              if ((len = p.decLimit()) < 0) return len;
              b.clear();
              for (int j = 0; j < UserPipe.PIPE_SIZE; j++) {
                  byte c = (byte) (rnd.nextInt(UserPipe.RANGE_ID) + p.getId());
                  b.put(c);
                  System.out.printf("ws%03d:%d ", p.getId(), c);
              }
              System.out.printf("%n");
              b.flip();
              while (pW.write(b) > 0) ;               // пока буфер не будет записан
              Thread.sleep(p.getDelay());                 // 500ms
              return len;

          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
              len = -1;
              return len;
          }
      }
      public static int readChannel(Pipe.SourceChannel pR, UserPipe p) {
          int len = -1;
          try {
              ByteBuffer b = ByteBuffer.allocate(UserPipe.PIPE_SIZE);
              Thread.sleep(100);
              while ((len = pR.read(b)) > 0) {
                  b.flip();
                  while (b.hasRemaining()) {
                      System.out.printf("rs%03d:%d ", p.getId(), (byte) (b.get() & 255));
                  }
                  b.clear();  //clear buffer
                  System.out.printf("%n");
              }
              return len;
          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
              len = -1;
              return len;
          }
      }
  }
  ```
-
  ```

- Пример. реализация Selector **java8_nio/nio/Main04S**
-
```
// Selector
        System.out.printf(FORMAT, "Selectors:");
        UserPipe[] pipes = {
                new UserPipe(100, 750),
                new UserPipe(110, 250),
                new UserPipe(60, 400),
                new UserPipe(80, 650)};

        Selector selector = null;
        try {
            selector = Selector.open();
            UserPipeUtils.registerReadChannels(selector, pipes, 0, 2);
            UserPipeUtils.registerWriteChannels(selector, pipes, 2, 4);

            int counter = 5;
            while (true) {
                int numReadyChannels = selector.select(250);
                if (numReadyChannels == 0) {
                    System.out.print(".");
                    if (counter-- > 0) {
                        continue;
                    } else {
                        break;
                    }
                }
                Set<SelectionKey> set = selector.selectedKeys();
                Iterator<SelectionKey> it = set.iterator();
                while (it.hasNext()) {
                    SelectionKey key = it.next();
                    if (key.isAcceptable()) {
                        System.out.printf("accepted:%s%n", key.attachment());
                    } else if (key.isReadable()) {
                        int len;
//                          System.out.printf("read:%s%n", key.attachment());
                        Pipe.SourceChannel pR = (Pipe.SourceChannel) key.channel();
                        UserPipe pAttach = (UserPipe) key.attachment();
                        len = UserPipeUtils.readChannel(pR, pAttach);
                        if (len < 0 && pR.isOpen()) {       // len = -1 when Pipe is empty and Send is
closed
//                              pR.register(selector,0);      // unregister all for channel ok
                            key.interestOps(0);             // unregister all for key ok
                            pR.close();                     // close channel ok
                        }
                    } else if (key.isWritable()) {
//                          System.out.printf("write:%s%n", key.attachment());
                        Pipe.SinkChannel pW = (Pipe.SinkChannel) key.channel();
                        UserPipe pAttach = (UserPipe) key.attachment();
                        int len = UserPipeUtils.writeChannel(pW, pAttach);
                        if (len < 0) {
//                            pR.register(selector,0);
                            key.interestOps(0);
                            pW.close();
                        }
                    }
                    it.remove(); // remove key
                }
            }
            System.out.printf("%nSelector finished%n");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeStream(selector);
            IOUtils.closeStream(pipes);
        }
```
-

Selector ServerSocket Example

- Selector ServerSocket Example пример работы ServerSocket
  - свойства ServerSocketChannel работает в nonblocking mode с Selector
  - SocketChannel работает к ак обычный клиент
- Пример. реализация Selector SocketChannel **java8_nio/nio/Main04S**

```
System.out.printf(FORMAT, "Selector ServerSocket:");
try {
    Runtime.getRuntime().exec("cmd /c start call java -ea -cp " +
            "out/production/java_nio nio.selector.SelectorServerSocket 9994");

//            Runtime.getRuntime().exec("cmd /c start call java -ea -cp " +
//                    "out/production/java_nio nio.selector.SelectorSocket 9994");
    for (int i = 0; i < 5; i++) {
        Thread.sleep(1000);
        SelectorSocket.main(new String[]{"9994"});
    }
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
```

```
public class SelectorSocket {
    public static void main(String[] args) {
        int port = 9995;
        if (args.length > 0) {
            try {
                port = Integer.parseInt(args[0]);
            } catch (NumberFormatException e) {
                e.printStackTrace();
                return;
            }
        }
        System.out.printf(FORMAT, "Client starting... connecting port: " + port);

        SocketChannel sc = null;
        ByteBuffer bb;
        try {
            sc = SocketChannel.open();
            bb = ByteBuffer.allocateDirect(8);

            sc = SocketChannel.open();
            InetSocketAddress address = new InetSocketAddress("localhost",port);
            sc.connect(address);
            long time = 0;
            while(sc.read(bb) != -1) {
                bb.flip();
                while(bb.hasRemaining()) {
//                    time <<= 8;
//                    time |= bb.get()&255;
                    time = bb.getLong();
                }
                bb.clear();
            }
            Instant instant = Instant.ofEpochMilli(time);
            LocalDateTime localTime = LocalDateTime.ofInstant(instant,ZoneId.systemDefault());
            DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd/MM/YYYY HH:mm:ss");
            System.out.printf("date: %s",localTime.format(dateTimeFormatter));

            sc.close();
        } catch (IOException e) {
            e.printStackTrace();
        }finally {
            IOUtils.closeStream(sc);

        }

    }
}
```

- Пример. реализация Selector ServerSocket **java8_nio/nio/Main04S**
- ```
  public class SelectorServerSocket {
      public static void main(String[] args) {
          int port = 9995;
          if (args.length > 0) {
              try {
                  port = Integer.parseInt(args[0]);
              } catch (NumberFormatException e) {
                  e.printStackTrace();
                  return;
              }
          }

          System.out.printf(FORMAT, "Server starting... listening port: " + port);

          ServerSocketChannel ssc = null;
          ByteBuffer bb;
          try {
              ssc = ServerSocketChannel.open();
              bb = ByteBuffer.allocateDirect(8);

              ServerSocket ss = ssc.socket();
              ss.bind(new InetSocketAddress(port));
              ssc.configureBlocking(false);
              Selector selector = Selector.open();
              ssc.register(selector, SelectionKey.OP_ACCEPT); // ожидает соединения

              while (true) {
                  int n = selector.select(500); // blocking method
                  if (n == 0) {
                      System.out.print(".");
                      Thread.sleep(500);
                      continue;
                  }

                  Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                  while(it.hasNext()) {
                      SelectionKey key = it.next();
                      if(key.isAcceptable()) {
                          SocketChannel sc = ((ServerSocketChannel)key.channel()).accept();
                          if(sc == null) continue;
                          System.out.printf("Receiving connection%n");
                          bb.clear();
                          bb.putLong(System.currentTimeMillis());
                          bb.flip();
                          while(bb.hasRemaining()){ //write ByteBuffer to SocketChannel of ServerSocketChannel
                              sc.write(bb);
                          }
                          sc.close();
                      }
                      it.remove();
                  }
              }


          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
          }finally {
              IOUtils.closeStream(ssc);
          }

      }

  }
  ```
-
  ```

Regex

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.util.regex Pattern, Matcher
    - Pattern скомпилированное представление регулярного выражения
    - позволяет назначить дополнительные функции сравнения lowcase, uppercase, ignorecase
    - Matcher механизм, предоставляет сравнение строки с Pattern
- Методы
    - compile() создание Pattern из выражения String regex и флагов
    - CANON_EQ Enables canonical equivalence.
    - CASE_INSENSITIVE Enables case-insensitive matching.
    - COMMENTS Permits whitespace and comments in pattern.
    - DOTALL Enables dotall mode, когда regex = .* берет в том числе и \r\n
    - LITERAL Enables literal parsing of the pattern.
    - MULTILINE Enables multiline mode.
    - UNICODE_CASE Enables Unicode-aware case folding.
    - UNICODE_CHARACTER_CLASS Enables the Unicode version of *Predefined character classes* and *POSIX character classes*.
    - UNIX_LINES Enables Unix lines mode.
    - flags() возвращает флаги, которые использовались при создании Pattern
    - split() разбивает строку на строки по паттерну в качестве разделителя
    - matches() собственно проверка соответствует ли строка паттерну
- PatternSyntaxException
    - это Exception которое выбрасывается если regex не соответствует синтаксису, например одна)
- Методы
    - у PatternSyntaxException есть свои методы которые вызываются прямо у объекта e
    - getDescription() получить описание ошибки
    - getIndex() индекс символа с ошибкой в строке regex
    - getMessage() сообщение об ошибке
    - getPattern() собственно исходный pattern
    - \\Q<spec>\\E использование spec символа как обычного, то есть \\s не <space>, а как \s
- Пример. реализация **java8_nio/nio/Main05R**
-

```
        String regex = "abdc";// regex
        String s = "abdc efgh1 abcd true come before efgh come before2 abdc efgh3";
        String regex2 = "abdc.*efgh3$";
        Pattern p = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);
        String[] ss =  p.split(s,8);
        for (String str : ss) {
            System.out.printf("%s%n",str);
        }
        System.out.printf("matches :%b  flags:%d%n",Pattern.matches(regex,s),p.flags());
        System.out.printf("matches :%b%n",Pattern.matches(regex2,s));
        Pattern p2=Pattern.compile(regex);
        Matcher m2 = p2.matcher(s);
        System.out.printf("matches2:%b%n",m2.matches());
        System.out.printf("matches2:%b%n",Pattern.compile(regex2).matcher(s).matches());
        System.out.printf(FORMAT,"Pattern Syntax:");
        try {
            p = Pattern.compile("abdc { cdba)");
        }catch (PatternSyntaxException e) {
            System.out.printf("getDescription:%s%n",e.getDescription());
            System.out.printf("getIndex      :%s%n",e.getIndex());
            System.out.printf("getMessage    :%s%n",e.getMessage());
            System.out.printf("getPattern    :%s%n",e.getPattern());
        }
    }
```

Matcher

- Matcher класс для работы с Regex
    - свойства
- Методы
    - matches() выдает true если есть полное совпадение шаблона regex с заданной строкой
    - lookingAt() выдает true если есть часть строки совпадает с шаблоном regex
    - find() ищет один за другим регионы строки, которые совпадают с шаблоном regex
    - используется в качестве итератора, работает с методами start(), end() и group()
    - позволяет
- Пример. реализация **java8_nio/nio/Main05R**
-

```
public class MainRegex {
//    private static final int[] PATTERN_FLAGS = {
//            Pattern.CASE_INSENSITIVE,
//            Pattern.DOTALL,
//            Pattern.CANON_EQ,
//            Pattern.MULTILINE,
//            Pattern.COMMENTS,
//            Pattern.LITERAL,
//            Pattern.UNICODE_CASE,
//            Pattern.UNICODE_CHARACTER_CLASS,
//            Pattern.UNIX_LINES
//
//    };

    public static void main(String regex, String s) {
        System.out.printf(FORMAT, "Regex group:");
        main(new String[]{regex, s});
    }

    public static void main(String regex, String s, int flags) {
        System.out.printf(FORMAT, "Regex group flags:");
        main(new String[]{regex, s, String.valueOf(flags)});
    }
    public static void main(String[] args) {

        int flags = 0;
        if (args.length < 2) {
            System.out.printf(FORMAT, "Usage: MainRegex.main(new String[]{regex, input}");
            return;
        }
        if (args.length == 3) {
            try {
                flags = Integer.parseInt(args[2]);

            } catch (NumberFormatException e) {
                System.out.printf("parse flags: failed%n");
            }
        }
        try {
            System.out.printf("regex: %s%n", args[0]);
            System.out.printf("input: %s%n", args[1]);
            Pattern p = Pattern.compile(args[0]);
            if (flags != 0) {
                p = Pattern.compile(args[0], flags);
            }
            Matcher m = p.matcher(args[1]);
            while (m.find()) {
                System.out.printf("group[%s] start:%d end:%d%n", m.group(), m.start(), m.end());
            }

        } catch (PatternSyntaxException e) {
            System.out.printf(FORMAT, "PatternSyntaxException:");
            System.out.printf("regex message:%s%n" + e.getMessage());
            System.out.printf("description  :%s%n" + e.getDescription());
            System.out.printf("index        :%s%n" + e.getIndex());
            System.out.printf("pattern      :%s%n" + e.getPattern());

        }
    }
}
```

Pattern Flags

- Pattern Flags флаги при работе с Pattern.compile()
  - CASE_INSENSITIVE сравнивает regex со строкой без учета регистра
  - DOTALL режим когда regex= «.*» берет также и символы перехода строки
  - в обычном режиме «.*» не берет «\r\n», в режиме DOTALL берет
- Пример. реализация
- ```
  // pattern DOTALL
  System.out.printf(FORMAT,"Regex Pattern.DOTALL:");
  regex = "abdc.*";
  s = "abdc efgh1 abcd true. come before\r efgh come before2 abdc efgh3";
  p = Pattern.compile(regex,Pattern.DOTALL);
  String [] strings = p.split(s);
  for (String string : strings) {
      System.out.printf("%s%n",string);
  }
  MainRegex.main(regex,s);
  MainRegex.main(regex,s,Pattern.DOTALL);
  ```

Regex Spec Characters

- Regex Spec Characters
  - `\\d` цифры 0-9 `\\D` не цифры
  - `\\s` пробелы [\t\n\0x0B\\f\r] `\\S` не пробелы `[^\\s]`
  - `\\w` символ слова [a-zA-Z0-9] `\\W` символ не слова `[^\\w]`
- Группы
  - `(A)` группа номера A=1
  - `((A)(B(C))` группы вложенные номера `((A)B(C))) = 1, (A)=2, (B(C))=3, C = 4`
  - группы можно использовать по номерам в pattern
  - `(abc) (bdf) \\2\\1 >> abc bdf bdf abc`
- Границы
  - `^` начало строки `$` конец строки
  - `\b` граница слова `\\B` граница не слова
  - `\\A` начало текста
  - `\\G?` конец предыдущего совпадения РАБОТАЕТ только со знаком вопроса
  - `\\z` конец текста `\\Z` конец текста без символа завершения строки
- Количественные квалификаторы
  - greedy least
  - `X?` ноль или один символ `X??` ноль или один символ
  - `X*` ноль или много символов `X*?` ноль или много символов
  - `X+` один или много символов `X+?` один или много символов
  - `<a.*b>` abc cdb cdb ccdc >> abc cdb cdb `<a.*?b>` abc cdb cdb ccdc >> ab
  - `X{n}` ровно n символов
  - `X{n,}` n или больше символов
  - `X{n,m}` n или не более m символов
  - greedy super
  - `X?+` ноль или один символ
  - `X*+` ноль или много символов выбирает все максимум, если .*+ то до конца строки
  - `X++` один или много символов
  - `<a.*+b>` abc cdb cdb ccdc >> «»
- ВНИМАНИЕ. Regex `<a.*+b>` выбирает по максимуму `<.*+>` до конца строки, поэтому НЕ НАХОДИТ <b>
  - т.к. выбраны все символы конструкцией `<.*+>` и символ <b> за пределами строки
  ```

- Пример. реализация **java8_nio/nio/Main05R**

```
// spaces
        System.out.printf(FORMAT,"Regex spaces:");
        MainRegex.main("\\sox"," b ox ");                    // spec character standard usage
        MainRegex.main("\\sox"," b ox \nox b\rox box");      // spec character standard usage
//group numbers
        System.out.printf(FORMAT,"Regex group numbers:");
        MainRegex.main("abc (bdf) (cfd) \\2 \\1","abc bdf cfd cfd bdf abc cfd abc bdf cfd cfd bdfa");
        MainRegex.main("(abc) (bdf) (cfd) \\3 \\2","abc bdf cfd cfd bdf abc cfd abc bdf cfd cfd bdfa");
//boundary numbers
        System.out.printf(FORMAT,"Regex word boundary greedy and not:");
        MainRegex.main("\\bb.*d\\b","abc bdf cfd cfd bdf abc");
        MainRegex.main("\\bb.*?d\\b","abc bdf cfd cfd bdf abc");
// greedy
        System.out.printf(FORMAT,"Regex word boundary greedy and not:");
        MainRegex.main("a.*b","abc cdb cdb ccdc ");
        MainRegex.main("a.*?b","abc cdb cdb ccdc ");
// greedy super
        MainRegex.main("a.*+b","abc cdb cdb ccdc ");         // empty
        MainRegex.main("a.*+","abc cdb cdb ccdc ");          // all string up to the end
        MainRegex.main("abc cdb c*+","abc cdb cccdefgt t"); // up to cccd
// \\G
        System.out.printf(FORMAT,"Regex boundary \\G:");
        MainRegex.main("abc.*\\G?d","abc cdb cdb ccdc ");
        MainRegex.main("abc.*?\\G?d","abc cdb cdb ccdc ");
// \\G and zero-length
        MainRegex.main(".?\\G","a");
        MainRegex.main("d\\G*.?g","dog dag"); // * \\G* относится к самому \\G ноль или повторы [dog][dag]
        MainRegex.main("\\G*.?g","dog dag");  // [og] [ag]
```

Practical Regex

- Practical Regex
  - `"(\\(\\d{3}\\))?\\s*\\d{3}-\\d{4}"` группа с кодом 0 или одна, затем 3 цифры, дефис, 4 цифры
  -
- Пример. реализация **java8_nio/nio/Main05R**

```
System.out.printf(FORMAT,"Regex Practical demo:");
regex = "(\\(\\d{3}\\))?\\s*\\d{3}-\\d{4}";
MainRegex.main(regex,"(800) 555-1212");
MainRegex.main(regex,"(800)555-1212");
MainRegex.main(regex,"555-1212");
```

Regex Question

- Regex Question
  - задача создать Matcher который уберет множественные пробелы в строке
- Пример. реализация **java8_nio/nio/Main05R**

```
// regex question
        System.out.printf(FORMAT, "Regex Question:");
        String sText = "Regex   word    boundary  greedy   and   not:   spaces removal string";
        Pattern pattern = Pattern.compile("\\s+");
        Matcher matcher = pattern.matcher(sText);

        String sResult = matcher.replaceAll(" ");
        System.out.printf("source:%s%n",sText);
        System.out.printf("result:%s%n",sResult);
```

Charset

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.nio.charset Charset, CharsetDecoder, CharsetEncoder, CoderResult, CodingErrorAction
    - Charset предоставляет методы отображения Unicode 16-bit символов в байты
    - стандартные Charset US-ASCII, ISO-8859-1, UTF-8, UTF-16
    - CharsetDecoder класс механизма, который декодирует byte[] array в Unicode 16-bit chars
    - CharsetEncoder класс механизма, который кодирует Unicode 16-bit chars в byte[] array
    - CoderResult класс содержит результат состояние работы кодера
    - CodingErrorAction enum класс ошибок при кодировании
- Java Charsets
    - Java Internal Charset UTF-16 проверяется печатью символа '\uXXXX', byte order mark задает
    - порядок байт, если нет или \uFFFF то UTF-16BE, если есть \uFFFE UTF-16LE
    - Java Platform Charset UTF-8 проверяется вызовом Charset.defaultCharset()
- Термины
    - character символ, который отображается на экране или печати
    - Character Set набор символов, то есть графических элементов
    - Coded Character Set набор символов, где каждому символу назначено уникальное число
    - стандартные наборы US-ASCII,ISO-8859-1 дают каждому символу свое число
    - Character Encoding Scheme кодирование чисел символов данного Coded Character Set в байты
    - каждый символ может быть закодирован в байты от 1 до 6.
    - Charset это Coded Character Set вместе с Character Encoding Scheme
    - Java использует Unicode Character Set для отображения символов
    - Charset.defaultCharset() получить Charset по умолчанию
- Методы
    - Character.isWhiteSpace() символ считается пробелом если его код лежит в диапазоне

```
* It is a Unicode space character ({@link #SPACE_SEPARATOR},
* {@link #LINE_SEPARATOR}, or {@link #PARAGRAPH_SEPARATOR})
* but is not also a non-breaking space ({@code '\u005Cu00A0'},
* {@code '\u005Cu2007'},                                    {@code '\u005Cu202F'}).
* {@code '\u005Ct'}, U+0009 HORIZONTAL TABULATION.    {@code '\u005Cn'}, U+000A LINE FEED.
* {@code '\u005Cu000B'}, U+000B VERTICAL TABULATION.  {@code '\u005Cf'}, U+000C FORM FEED.
* {@code '\u005Cr'}, U+000D CARRIAGE RETURN.          {@code '\u005Cu001C'},U+001C FILE SEPARATOR.
* {@code '\u005Cu001D'}, U+001D GROUP SEPARATOR.      {@code '\u005Cu001E'},U+001E RECORD SEPARATOR
* {@code '\u005Cu001F'}, U+001F UNIT SEPARATOR.
```

    - Character.isISOControl() символ считается ISO Control если его код лежит в диапазоне

```
* code is in the range {@code '\u005Cu0000'} through {@code '\u005Cu001F'}
* or in the range  {@code '\u005Cu007F'} through {@code '\u005Cu009F'}.
```

    - Character.getType() возвращает тип символа

```
* @see     Character#COMBINING_SPACING_MARK          * @see     Character#CONNECTOR_PUNCTUATION
* @see     Character#CONTROL                         * @see     Character#CURRENCY_SYMBOL
* @see     Character#DASH_PUNCTUATION                * @see     Character#DECIMAL_DIGIT_NUMBER
* @see     Character#ENCLOSING_MARK                  * @see     Character#END_PUNCTUATION
* @see     Character#FINAL_QUOTE_PUNCTUATION         * @see     Character#FORMAT
* @see     Character#INITIAL_QUOTE_PUNCTUATION       * @see     Character#LETTER_NUMBER
* @see     Character#LINE_SEPARATOR                  * @see     Character#LOWERCASE_LETTER
* @see     Character#MATH_SYMBOL                     * @see     Character#MODIFIER_LETTER
* @see     Character#MODIFIER_SYMBOL                 * @see     Character#NON_SPACING_MARK
* @see     Character#OTHER_LETTER                    * @see     Character#OTHER_NUMBER
* @see     Character#OTHER_PUNCTUATION               * @see     Character#OTHER_SYMBOL
* @see     Character#PARAGRAPH_SEPARATOR             * @see     Character#PRIVATE_USE
* @see     Character#SPACE_SEPARATOR                 * @see     Character#START_PUNCTUATION
* @see     Character#SURROGATE                       * @see     Character#TITLECASE_LETTER
* @see     Character#UNASSIGNED                      * @see     Character#UPPERCASE_LETTER
```

- getDirectionality() определяет Unicode directionality характеризует упорядоченность текста
- isMirrored() определяет может ли символ зеркально отображаться по горизонтали
- isIdeographic() определяет является ли символ иероглифом

- Пример. реализация проверки Charset по умолчанию **java8_nio/nio/Main06C**

```
System.out.printf("C3A7:%c C3A9:%c %n", '\uC3A7','\uC3A9');     // вывод по умолчанию в формате '\u0000'
charset = Charset.forName("UTF-16");                             // вывод кодирование в Charset.UTF-16
System.out.printf("C3A7:%s C3A9:%s %n", new String(new byte[]{(byte)0xC3,(byte)0xA7},charset),
        new String(new byte[]{(byte)0xC3,(byte)0xA9},charset));
```

Charset Examples

- Charset encoding example
  - показывает кодировку строки в байты разными Charsets
  - позволяет
- Пример. реализация **java8_nio/nio/Main06C**

```
System.out.printf(FORMAT, "Charsets:");
String[] charsets = {"US-ASCII", "ISO-8859-1", "UTF-8",
        "UTF-16BE", "UTF-16LE", "UTF-16"
};
String s = "façade touché";
for (String charset : charsets) {
    encode(s, Charset.forName(charset));
}

System.out.printf(FORMAT, "Checking UTF-8:");
Charset charset = Charset.forName("UTF-8");

System.out.printf("C3A7:%c C3A9:%c %n", '\uC3A7', '\uC3A9');
System.out.printf("C3A7:%s C3A9:%s %n", new String(new byte[]{(byte) 0xC3, (byte) 0xA7}, charset),
        new String(new byte[]{(byte) 0xC3, (byte) 0xA9}, charset));

System.out.printf("E7:%c E9:%c %n", (char) 0xE7, (char) 0xE9);

charset = Charset.forName("UTF-16");
System.out.printf("C3A7:%s C3A9:%s %n", new String(new byte[]{(byte) 0xC3, (byte) 0xA7}, charset),
        new String(new byte[]{(byte) 0xC3, (byte) 0xA9}, charset));
```

- Charset to ByteBuffer example
  - показывает кодирование и переход между ByteBuffer и CharBuffer
- ВНИМАНИЕ. CharBuffer toString() ВЫВОДИТ буфер напрямую на печать ИСПОЛЬЗУЕТ UTF16
- Пример. реализация **java8_nio/nio/Main06C**

```
System.out.printf(FORMAT, "Charset and ByteBuffer, CharBuffer:");
charset = Charset.forName("UTF-8");
ByteBuffer bb = charset.encode(s);
System.out.printf("array: %s%n", output(bb));
CharBuffer cb = CharBuffer.wrap(s);
ByteBuffer bbc = charset.encode(cb);
System.out.printf("array: %s%n", output(bbc));

System.out.printf(FORMAT, "CharrBuffer to String:");
bb.rewind();
CharBuffer cbb = charset.decode(bb);
System.out.printf("array: %s%n", output(cbb));
cbb.rewind();
s = cbb.toString();

System.out.printf("array: %s%n", cbb);
System.out.printf(FORMAT, "ByteBuffer asCharBuffer to String:");
bb = ByteBuffer.allocate(s.length() * 2);
bb.asCharBuffer().put(s);
cbb = Charset.forName("UTF-8").decode(bb);
System.out.printf("UTF8 : %s%n", cbb);
bb.rewind();
cbb = Charset.forName("UTF-16").decode(bb);
System.out.printf("UTF16: %s%n", cbb);
bb.rewind();
System.out.printf("UTF16 asCharBuffer: %s%n", bb.asCharBuffer().toString());
```

Formatter

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO Components java.util Formatter
  - Formatter класс для форматирования строк выводимых на печать
  - Appendable интерфейс, который обязан реализовать любой класс принимающий Formatter
  - возвращает экземпляр того же класса по интерфейсу Appendable
  - позволяет создать цепочку добавлений
  - Formattable интерфейс для реализации своего класса поддерживающего форматирование
  - FormattableFlags класс констант для интерфейса Formattable
- Методы
  - format() добавить к объекту форматированные данные
  - flush() обязательно после заполнения Formatter чтобы полностью вывести все данные
  - close() обязательно закрывать после использования
  -
- Пример. реализация **java8_nio/nio/Main07F**

Format

- Format строк
  - %[index$][flags][width][.precision]conversion
  - index номер аргумента в строке, позволяет один аргумент поместить несколько раз
  - 1$ первый аргумент, 2$ второй аргумент
  - flags флаги модификаторы, например %-10s, %+d
  - – LEFT JUSTIFY + обязательно знак числа
  - ' ' заполнить пробелом 0 заполнить нулями
  - финансовые
  - (писать отрицательные в () , разделять группы чисел запятой
  - width ширина целой части поля for String минимальный размер поля
  - precision ширина поля после запятой for String максимальный размер поля
- ВНИМАНИЕ. Если width > precision для String это значит строка усекается до precision и расширяется до width
- "abcdefg" %8,4s даст " abcd"
- Conversion
  - %b, %c, %d, %f, %n, %s, %x
  - %t дата %T время
  - %e день месяца
  -
- Пример. реализация **java8_nio/nio/Main07F**

Java NIO2

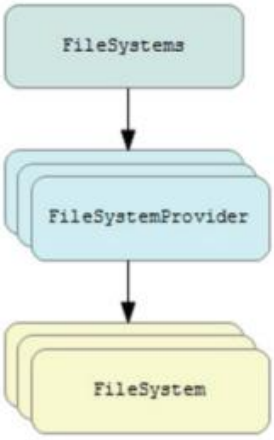
- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Java NIO java.nio.file Files, FileStore, FileSystem, FileSystems, LinkPermission, Paths,
    - SimpleFileVisitor, StandardWatchEventKinds
    - Files статические методы для работы с файлами и каталогами
    - FileStore класс хранения файлов
    - FileSystem интерфейс к фабрике файловых систем
    - FileSystems класс статических методов фабрики файловых систем
    - LinkPermission класс разрешений для создания ссылок к файлам
    - Paths класс статических методов работы с Path и конвертации в URI
    - SimpleFileVisitor класс работы с каталогами и рекурсивным поиском
    - StandardWatchEventKinds класс типов WatchEvent
- Java NIO2 Components улучшенная система ввода вывода
    - свойства методы выдают Exception
    - поддерживаются символьные ссылки на файлы
    - мощная поддержка работы с атрибутами
    - поддержка работы потоков для каталогов см AsyncronousFileChannel
    - поддержка копирования и переноса файлов, рекурсивного обхода каталогов
- Packages
    - java.nio.file классы и интерфейсы для работы с файлами и каталогами
    - java.nio.file.attribute классы и интерфейсы для работы с атрибутами файлов и каталогов
    - java.nio.file.spi класс предоставляет службы для работы с файловыми системами
- Пример. реализация **java8_nio/nio2/Main01**
- Java NIO1 Update
- обновление информации по селекторам прерываемым и асинхронным каналам

Java NIO2 Components

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- FileSystems
- Path Classes
- Files Class
- File Attribute Classes
- Files Static Methods
- Paths, Create File, Create Temporary Files, Read Files, Read Large Files, Write Small Files,Write Large Files
- Random Access File NIO2
- Create Directory_Create Temp Directory_Listing Directory
- PathMatcher
- Glob Syntax
- Copy Files
- Moving Files
- Delete Files
- Symbolic and Hard Links
- File Tree Walk
- Files Streams Methods
- Watching Directories

FileSystems

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- FileSystems работа с файловыми системами
- Методы
    - getDefault() получить объект с файловой системой по умолчанию
    - с параметром URI получить доступ к файловой системе с адресом ресурса
    - newFileSystem() создать объект new FileSystems
- Пример. реализация вывод списка FileSystemProvider **java8_nio/nio2/Main01**
- ```
  List<FileSystemProvider> list = FileSystemProvider.installedProviders();
  for (FileSystemProvider provider : list) {
      System.out.printf("%s scheme:%s%n", provider, provider.getScheme());
  }
  ```



FileSystem

- FileSystem работа с файловой системой
    - FileSystems.getDefault() получить собственно FileSystem по умолчанию
    - fs.getSeparator() получить разделитель пути в File System
    - fs.provider().getScheme() возвращает URI scheme для данного fs.provider()
    - fs.getPath() возвращает объект Path для данной FileSystem, работает как Paths.get()
    - можно собрать прямо из названий каталогов предоставив кучу String
      ```
      Path pathY = fs.getPath(".","data","nio","result.txt").toAbsolutePath();
      ```
    - fs.supportedFileAttributeViews() возвращает поддерживаемые FileAttribute класс File System
    - поддерживаемые: owner, dos, acl, basic, user
- ВНИМАНИЕ. Paths.get() это СОКРАЩЕННАЯ форма метода FileSystems.getDefault().getPath()
      ```

Path Classes

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Path работа с путями файлов и каталогов
    - свойства
- Методы
    - toAbsolutePath() возвращает абсолютный путь Path объект
    - toFile() возвращает File объект для данного Path объекта
    - getFileSystem() возвращает объект FileSystem для данного Path объекта
    - getFileName() возвращает имя файла для данного Path объекта
    - getName() возвращает имя сегмента Path объекта по индексу
    - getNameCount() возвращает число сегментов Path объекта
    - getRoot() возвращает root Path для данного Path объекта
    - toUri() вернуть URI объект Path
    - File.toPath() возвращает объект Path для данного File объекта
    - URI.create() создает URI объект по строке и Path объекту, как показано в примере
    - URI.create("file:///"+path.toString().replaceAll("\\\\","/"))
- Paths статические метолды работы с путями файлов и каталогов
- Методы
    - Paths.get() получить объект Path по параметру String или URI
- ВНИМАНИЕ. Paths.get() это СОКРАЩЕННАЯ форма метода FileSystems.getDefault().getPath()
- Path transforms
    - normalize() убрать "./" обозначение текущего каталога из пути
    - realitivize() создать Path объект относительный к данному Path объекту
    - resolve() добавляет relativized Path в конец к данному Path объекту
    - по сути восстанавливает оригинальный Path объекта который был relativized()
    - ./tmp/a ./tmp/b >> pathRel = ../b >> pathRes=./tmp/a/../b [>> ./tmp/b]
- ВНИМАНИЕ. relativized() НЕ РАБОТАЕТ для siblings Path, использовать АБСОЛЮТНЫЕ пути или resolveSiblings()
- Пример. реализация **java8_nio/nio2/Main01**

```
System.out.printf(FORMAT, "Path Normalize Relative Resolution");
path = fs.getPath(".", "data", "nio", "result.txt");      // relative path
pathX = fs.getPath(".", "data", "stream","buff-base64.txt");              // relative path
path = path.toAbsolutePath();
pathX = pathX.toAbsolutePath();
pathR = path.relativize(pathX);
pathE = Paths.get(path.toString(),pathR.toString());
pathF = path.resolve(pathR);
System.out.printf("abs :%s%n", path);
System.out.printf("abs2:%s%n", pathX);
System.out.printf("path:%-15s exists:%b%n",pathR,pathR.toFile().exists());
System.out.printf("rest:%-25s exists:%b abs:%b%n",pathE,pathE.toFile().exists(),pathE.toAbsolutePath());
System.out.printf("norm:%-25s exists:%b abs:%b%n",pathE.normalize(),pathE.normalize().toFile().exists(),
        pathE.normalize().isAbsolute());
System.out.printf("rsvd:%-25s exists:%b abs:%b%n",pathF,pathF.toFile().exists(),pathF.toAbsolutePath());
System.out.printf("norm:%-25s exists:%b abs:%b%n",pathF.normalize(),pathF.normalize().toFile().exists(),
        pathF.normalize().isAbsolute());
```

Path siblings

- Path siblings()
  - resolveSiblings() работает с именами каталогов только на текущем уровне
  - в отличие от resolve() которое работает с точными путями
- Пример. реализация **java8_nio/nio2/Main01**

```
System.out.printf("%nstring: String[%s]%n",pathR);
pathE = path.resolve(pathR.toString());
pathF = path.resolveSibling(pathR.toString());
System.out.printf("resolve()  rest:%-25s exists:%b%n",pathE,pathE.toFile().exists());
System.out.printf("resolve()  norm:%-25s
exists:%b%n",pathE.normalize(),pathE.normalize().toFile().exists());
System.out.printf("siblings() rsvd:%-25s exists:%b%n",pathF,pathF.toFile().exists());
System.out.printf("siblings() norm:%-25s
exists:%b%n",pathF.normalize(),pathF.normalize().toFile().exists());
```

Path check

- Path check
  - equals() сравнивает пути
  - compareTo() сравнивает пути
  - startWith() сравнивает
  - endsWith() сравнивает
  - realPath() возвращает реальный путь до файла по пути связанному с ссылкой.
  - toUri() возвращает URI объект версию Path объекта
- Пример. реализация **java8_nio/nio2/Main01**

```
System.out.printf(FORMAT, "Path comparisons:");
path = fs.getPath(".", "data", "nio");
pathX = fs.getPath(".", "data", "stream").normalize();
pathY = Paths.get("./data/nio");

System.out.printf("path : %s%n", path);
System.out.printf("pathX: %s%n", pathX);
System.out.printf("pathY: %s%n", pathY);


System.out.printf("path.equals(pathX): %b%n", path.equals(pathX));
System.out.printf("path.equals(pathY): %b%n", path.equals(pathY));
System.out.printf("%n");
System.out.printf("path.compareTo(pathX): %b%n", path.compareTo(pathX));
System.out.printf("path.compareTo(pathY): %b%n", path.compareTo(pathY));
System.out.printf("%n");
System.out.printf("path.startsWith(\".\\data\"): %b%n", path.startsWith(".\\data"));
System.out.printf("pathX.startsWith(\"data\") : %b%n", pathX.startsWith("data"));

System.out.printf("%n");
System.out.printf("path.endsWith(\"nio\"): %b%n", path.endsWith("nio"));
System.out.printf("pathX.endsWith(\"stream\"): %b%n", pathX.endsWith("stream"));
try {
    System.out.printf("%n");
    System.out.printf("path.toUri() : %s%n", path.toUri());
    System.out.printf("pathX.toUri(): %s%n", pathX.toUri());
    System.out.printf("path.realPath() : %s%n", path.toRealPath(LinkOption.NOFOLLOW_LINKS));
    System.out.printf("pathX.realPath(): %s%n", pathX.toRealPath(LinkOption.NOFOLLOW_LINKS));
} catch (IOException e) {
    e.printStackTrace();
}
```

Files Class

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Files Class статические методы для работы с файлами
    - getFileStore() получить объект носителя по данному Path объекту
    - readAttributes() возвращает
    - getAttributes() возвращает
    - setAttributes() возвращает

FileStore Class

- FileStore Class
    - класс работы с носителем данных, HDD, USB Drive, DVD
    - методы позволяют вывести всю информацию о носителе
- Методы
    - getTotalSpace() возвращает размер носителя
    - getUnallocatedSpace() возвращает размер свободного пространства
    - getUsableSpace() возвращает размер занятого пространства
    - isReadOnly() возвращает флаг readOnly()
    - name() возвращает имя FileStore
    - type() возвращает тип FileStore
- Пример. реализация **java8_nio/nio2/Main01**
- ```
  try {
      for (Path pathRoot : fs.getRootDirectories()) {
          if(!pathRoot.toFile().exists()) {
              System.out.printf("Path: %s not ready%n", pathRoot);
              continue;
          }
          FileStore fileStore = Files.getFileStore(pathRoot);
          System.out.printf("Path: %s%n", pathRoot);
          System.out.printf("Name: %s%n", fileStore.name());
          System.out.printf("Type: %s%n", fileStore.type());
          System.out.printf("Total space: %d%n", fileStore.getTotalSpace());
          System.out.printf("Unallocated space: %d%n",
                  fileStore.getUnallocatedSpace());
          System.out.printf("Usable space: %d%n",
                  fileStore.getUsableSpace());
          System.out.printf("Read only: %b%n", fileStore.isReadOnly());
          System.out.printf("%n");
      }

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

File Attribute Classes

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- File Attribute Classes BasicFileAttributeView, FileOwnerAttributeView, UserDefinedFileAttributeView
    - BasicFileAttributeView childs DosFileAttributeView, PosixFileAttributeView

BasicFileAttributeView Class

- BasicFileAttributeView Class
    - класс поддерживает атрибуты для большинства файловых систем
    - DosFileAttributeView поддерживает legacy MS_DOS/PC-DOS
    - PosixFileAttributeView поддерживает атрибуты OS, которые поддерживают POSIX стандарт
- Структура классов атрибутов
    - FileAttributeView >> BasicFileAttributeView, FileOwnerAttributeView, UserDefinedFileAttributeView

- Методы
    - creationTime() время создания set поддерживается
    - fileKey() Object
    - isDirectory() каталог
    - isOther() есть ли другое имя
    - isRegular() обычный файл
    - isSymbolinkLink() ссылка
    - lastAccessTime() время последнего доступа set поддерживается
    - lastModifiedTime() время последнего изменения set поддерживается
    - size() размер файла
- ВНИМАНИЕ. Практически можно задать только время, остальные атрибуты только для чтения
- Пример. реализация **java8_nio/nio2/Main01**

FileOwnerAttributeView Class

- FileOwnerAttributeView
    - класс поддерживает чтение или изменение атрибутов владельцем файла
    - AclFileAttributeView поддерживает чтение и update ACL атрибутов или owner атрибутов
    - PosixFileAttributeView поддерживает атрибуты OS, которые поддерживают POSIX стандарт
- Методы
- Пример. реализация **java8_nio/nio2/Main01**

UserDefinedFileAttributeView Class

- UserDefinedFileAttributeView
    - класс поддерживает user-defined атрибуты
- Методы
    - delete() удалить атрибут
    - list() вернуть список атрибутов
    - read() прочитать значение атрибута
    - size() получить размер user-defined атрибута
    - write() записать значение атрибута
- Пример. реализация **java8_nio/nio2/Main01**

Files Static Methods

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Paths, Create File, Create Temporary Files, Read Files, Read Large Files, Write Small Files,Write Large Files
- Random Access File NIO2

Paths

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Checking Paths методы
    - Files.exists(path) true если существует файл или каталог
    - Files.notExists(path) true если НЕ существует файл или каталог
    - Files.isDirectory(path) true если по заданному пути каталог
    - Files.isExecutable(path) true если по заданному пути каталог или исполняемый файл ???
    - Files.isRegularFile(path) true если по заданному пути файл
    - Files.isReadable(path) true если по заданному пути файл или каталог можно прочитать
    - Files.isWritable(path) true если по заданному пути в файл или каталог можно записать
    - Files.isSymbolicLink(path) true если по заданному пути файл symbolic link
- ВНИМАНИЕ. В Windows файлы Symbolic link толком не работают, это просто нечто с нулевой длиной и все
- ВНИМАНИЕ. Методы Paths которые ТРЕБУЮТ осторожности и могут создать Race Condition TOCTTOU
- 40. True or false: !exists(path) is equivalent to notExists(path).
    - false exists() вернет false если невозможно определить есть файл или нет
    - notExists() вернет false если невозможно определить есть файл или нет
    - exists() NOT Atomic то есть небезопасный метод для thread
    - notExists Atomic безопасный для thread safe
- 41. What is the isDirectory() method's default policy on symbolic links?
    - по умолчанию метод следует по всем symbolicLink и определяет реальный target объект
- 42. Why should you be careful when using the return values from exists(), notExists(),isExecutable(), isReadable(), and isWritable()?
    - значение которое возвращают эти методы НЕМЕДЛЕННО устаревает
- Пример. реализация **java8_nio/nio2/Main02D**
- ```
  private static void pathInfo(Path path) throws IOException {
      System.out.printf("%n");
      System.out.printf("path            : %s%n", path);
      System.out.printf("exists          : %b%n", Files.exists(path));
      System.out.printf("not exists      : %b%n", Files.notExists(path));
      System.out.printf("is directory    : %b%n", Files.isDirectory(path));
      System.out.printf("is executable   : %b%n", Files.isExecutable(path));
      System.out.printf("is regular      : %b%n", Files.isRegularFile(path));
      System.out.printf("is readable     : %b%n", Files.isReadable(path));
      System.out.printf("is writable     : %b%n", Files.isWritable(path));
      System.out.printf("is symbolic link: %b%n", Files.isSymbolicLink(path));
      if (Files.exists(path)){
           System.out.printf("is hidden       : %b%n", Files.isHidden(path));
      }
  }
  ```
  ```

Create File

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Создание файлов методы
    - createFile(path,attr) создание файла, параметры Path и объект FileAttrinbute
    - FileAttribute объект на базе интерфейса , методы: name(), value()
- Пример. реализация создания в Windows
- ```
  public static FileAttribute<List<AclEntry>> attributes(Path path) throws IOException {
      final List<AclEntry> list = new ArrayList<>();
      List rawList = (List) Files.getAttribute(path, "acl:acl");
      for (Object o : rawList) {
          AclEntry aclEntry = (AclEntry) o;
          list.add(aclEntry);
      }
      return new FileAttribute<>() {
          @Override
          public String name() {
              return "acl:acl";
          }
          @Override
          public List<AclEntry> value() {
              return list;
          }
      };
  }
  ```
- ```
          path = Paths.get(".", "data", "nio");
               Path pathD = Paths.get(path.toString(), "check.ini");
               Path pathE = Paths.get(path.toString(), "check.txt");
               FileAttribute<List<AclEntry>> fileAttr = MainAttrUtils.attributes(path);
               pathR = Files.createFile(pathD, fileAttr);
               pathR = Files.createFile(pathE);
  ```
- ВНИМАНИЕ. В Windows разрешения можно задать через acl:acl но это много и непрактично
- В Unix задать разрешения реально через POSIX permissions
- Пример. реализация создания в Unix POSIX **java8_nio/nio2/Main02D**

```
// Posix
                Set<PosixFilePermission> set = Files.getPosixFilePermissions(path);
                FileAttribute<Set<PosixFilePermission>> fpAttr = PosixFilePermissions.asFileAttribute(set);

                Set<PosixFilePermission> set = PosixFilePermissions.fromString("rw-r-xr-x");
                FileAttribute<Set<PosixFilePermission>> fpAttr = PosixFilePermissions.asFileAttribute(set);
                System.out.printf("name:%s%n",fpAttr.name());
                for (PosixFilePermission perm : fpAttr.value()) {
                    System.out.printf("permission:%s%n",perm.toString());
                }
                if(!Files.isDirectory(path)) {
                    path = path.subpath(0, path.getNameCount() - 1);
                }
                path = Paths.get(path.toString(),"resultPosix.txt");
                pathR = Files.createFile(path,fpAttr);
                pathInfo(pathR);
```

Create Temporary Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Create Temporary Files
    - createTempFile() создание временных файлов группой и с возможным автоудалением
- Автоматическое удаление по завершение программы
    - deleteOnExit() Path.toFile(path) >> File.deleteOnExit(file) удалить файл на выходе
    - shutdownHook() hook = new Thread() >> Runtime.getRuntime().addShutdownHook(hook)
    - newOutpuStream() Files.newOutputStream(Path, StandardOpenOption.DELETE_ON_CLOSE)
- ВНИМАНИЕ. Метод newOutputStream() СОЗДАЕТ Channels.OutputStream на базе AsynchronousChannel
- Пример. реализация Create Temp File **java8_nio/nio2/Main02D**

```
        List<OutputStream> listStreams = new ArrayList<>();
        try {
            path = Paths.get(".", "data", "temp");
            String prefix = "temp";
            String suffix = "file";
            if (!Files.exists(path))                path = Files.createDirectory(path);
            List<Path> listTempFiles = new ArrayList<>();
            final List<Path> listHookFiles = new ArrayList<>();
            listTempFiles.add(Files.createTempFile(path, prefix, suffix));
            listTempFiles.add(Files.createTempFile(path, prefix, null));
// program to delete on exit
            for (Path tempPath : listTempFiles) {
                tempPath.toFile().deleteOnExit();
            }
// runtime hook
            Thread hook = new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        for (Path filePath : listHookFiles) {
                            Files.deleteIfExists(filePath);
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
            listHookFiles.add(Files.createTempFile(path, null, suffix));
            listHookFiles.add(Files.createTempFile(path, null, suffix));
            Runtime.getRuntime().addShutdownHook(hook);
// output stream
            pathC = Paths.get(path.toString(), prefix + "12332.tmp");
            Files.deleteIfExists(pathC);
            listStreams.add (Files.newOutputStream(pathC, StandardOpenOption.DELETE_ON_CLOSE,
                                        StandardOpenOption.CREATE_NEW,StandardOpenOption.WRITE);
            listStreams.add(Files.newOutputStream(Paths.get(path.toString(), prefix + "12333.tmp"),
                    StandardOpenOption.DELETE_ON_CLOSE, StandardOpenOption.CREATE,StandardOpenOption.WRITE);
            listStreams.add(out);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(listStreams.toArray(new OutputStream[0]));
        }
```

Read Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Read Files Small
    - readAllBytes() прочитать массив байт, ограничение сверху 2Gb
    - readAllLines(path, Charset) прочитать строк с заданным Charset, максимум 2Gb
- Пример. реализация **java8_nio/nio2/Main02D**

```
        path = Paths.get(".", "data", "nio");
        Path pathD = Paths.get(path.toString(), "result.txt");
        Path pathE = Paths.get(path.toString(), "result_k.txt");
        BufferedReader br = null;
        InputStream in = null;
        try {
            byte[] bytes = Files.readAllBytes(pathD);
            in = new ByteArrayInputStream(bytes);
            br = new BufferedReader(new InputStreamReader(in, StandardCharsets.UTF_8));

            String s;
            while ((s = br.readLine()) != null) {
                System.out.printf("%s%n", s);
            }
            br.close();
            in.close();  // не требуется для ByteArray
// conversion one
            ByteBuffer b = ByteBuffer.wrap(Files.readAllBytes(Paths.get(path.toString(),"result_w.txt")));
            CharBuffer cb = CharBuffer.wrap(new String(b.array(),Charset.forName("WINDOWS-1251")));
            System.out.printf("%s%n",cb);
// conversion two
            ByteBuffer b2 = ByteBuffer.wrap(Files.readAllBytes(pathE));
            CharBuffer cb2 = Charset.forName("KOI8-R").decode(b2);
            System.out.printf("%s%n",cb2);
            List<String> list = Files.readAllLines(pathD);
            for (String string : list) {
                System.out.printf("%s%n",string);
            }
            List<String> listR = Files.readAllLines(pathE, Charset.forName("KOI8-R"));
            for (String string : listR) {
                System.out.printf("%s%n",string);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

OpenOptions

- StandardOpenOption Class
    - APPEND добавить новые данные в конец файла
    - CREATE создать файл, текущий файл НЕ ОБНУЛЯЕТ только с WRITE
    - CREATE_NEW создать новый файл или Exception если существует, только с WRITE
    - DELETE_ON_CLOSE удалить файл при закрытии приложения
    - DSYNC синхронизировать буферы файла и носитель
    - READ открыть файл для чтения
    - SPARSE создает sparse файл, по сути сжатая версия обычного файла
    - SYNC синхронизировать буферы файла и носитель
    - TRUNCATE_EXISTING если открывается с WRITE то обрезать длину до 0
    - WRITE открыть файл для записи

CopyOptions

- StandardCopyOptions Class
    - REPLACE_EXISTING заместить текущий файл
    - COPY_ATTRIBUTES скопировать атрибуты файла,
    - ATOMIC_MOVE переместить файл как Atomic то есть в Thread safe манере

Read Large Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Read Large Files прочитать на базе BufferedReader
    - newBufferedReader() прочитать в поток BufferedReader , использует StandardCharsets.UTF8
    - newBufferedReader(Path,Charset) прочитать в поток BufferedReader с Charset
    - new InputStream(Path,OpenOptions) создать поток на базе файла и опций
- ВНИМАНИЕ. OpenOptions описаны далее в разделе Write Files
- Пример. реализация **java8_nio/nio2/Main02D**
-

```
        System.out.printf(FORMAT, "Read Large Files :");
        pathD = Paths.get(".", "data", "nio2", "result.txt");
        pathE = Paths.get(".", "data", "nio2", "result_k.txt");

        BufferedReader br = null;
        BufferedWriter bw = null;
        InputStream in = null;
        OutputStream out = null;
        try {
// conversion one
            br = Files.newBufferedReader(pathD);
            br.lines().forEach(s -> System.out.printf("%s%n", s));
            br.close();
// conversion two
            br = Files.newBufferedReader(pathE, Charset.forName("KOI8-R"));
            br.lines().forEach(s -> System.out.printf("%s%n", s));
            br.close();
// console stream
            pathD = Paths.get(".", "data", "nio2", "result_k.txt");
            pathE = Paths.get(".", "data", "nio2", "result_in.txt");
            Files.copy(pathD, pathE, StandardCopyOption.REPLACE_EXISTING);

            in = Files.newInputStream(pathE);
            String s = new String(in.readAllBytes(), Charset.forName("KOI8-R"));
            System.out.printf("%s%n", s);
            in.close();

            System.out.println("Before channels:");
            ReadableByteChannel rc = Channels.newChannel(Files.newInputStream(pathE));
            WritableByteChannel wc =
Channels.newChannel(Files.newOutputStream(pathE.getParent().resolve("result_out.txt")));
            ByteBuffer b = ByteBuffer.allocate(50);
            while(rc.read(b) > 0) {
                b.flip();
                wc.write(b);
                b.compact();
            }
            System.out.println("After channels:");
            in = Files.newInputStream(pathE.getParent().resolve("result_out.txt"));
            s = new String(in.readAllBytes(), Charset.forName("KOI8-R"));
            System.out.printf("%s%n", s);
            in.close();


// temp
//            Files.copy(pathD,pathE,StandardCopyOption.REPLACE_EXISTING);
//            System.out.printf("path:%s  size:%d%n",pathE.getFileName(),Files.size(pathE));
//            OutputStream out = Files.newOutputStream(pathE, StandardOpenOption.CREATE,
//                   StandardOpenOption.TRUNCATE_EXISTING);
//            System.out.printf("path:%s  size:%d%n",pathE.getFileName(),Files.size(pathE));
//            out.write(121);
//            out.close();
//            in = Files.newInputStream(pathE);
//            s = new String(in.readAllBytes(),StandardCharsets.UTF_8);
//            System.out.printf("%s%n",s);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(br, bw, in, out);
        }
```

Write Small Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Write Small Files прочитать на базе BufferedReader
    - Files.write(Path, byte[],Options) запись массива байт byte[]
    - Files.write(PathE,List<String>,Charset, Options) запись списка строк List<String> с перекодировкой
    - StandardOptions.TRUNCATE_EXISTING позволяет при открытии файла задать длину 0 байт
- Пример. реализация **java8_nio/nio2/Main02D**
- ```
        System.out.printf(FORMAT, "Write Files :");
        path = Paths.get(".", "data", "nio");
        pathD = Paths.get(path.toString(), "result_k.txt");
        pathE = Paths.get(path.toString(), "result_k_small.txt");
        try {
    // bytes
            System.out.printf("Bytes:%n");
    // source
            Files.deleteIfExists(pathE);
            byte[] bytes = Files.readAllBytes(pathD);
    // write all bytes[]
            Files.write(pathE, bytes, StandardOpenOption.APPEND, StandardOpenOption.CREATE);
            pathR = Files.write(pathE, bytes, StandardOpenOption.APPEND);
            pathInfo(pathR);
    // write 25 bytes
            pathR = Files.write(pathE,
                    Arrays.copyOfRange(bytes, 0, 25), StandardOpenOption.TRUNCATE_EXISTING);
            pathInfo(pathR);
            System.out.printf("%s%n", new String(Files.readAllBytes(pathE), Charset.forName("KOI8-R")));
    // lines
            System.out.printf("Lines:%n");
            pathD = Paths.get(path.toString(), "result_u.txt");
            pathE = Paths.get(path.toString(), "result_k.txt");
    // source
            List<String> listK = Files.readAllLines(pathD, StandardCharsets.UTF_8);
            listK.addAll(Files.readAllLines(pathE, Charset.forName("KOI8-R")));
    // write all
            pathE = Paths.get(path.toString(), "result_k_small.txt");
            pathR = Files.write(pathE, listK,
                                Charset.forName("KOI8-R"), StandardOpenOption.TRUNCATE_EXISTING);
    // checkout
            System.out.printf("%s%n", new String(Files.readAllBytes(pathR), Charset.forName("KOI8-R")));
            System.out.printf("Truncate check:%n");
            Path pathF = Paths.get(path.toString(), "result_k_small_copy.txt");
            Files.copy(pathE, pathF, StandardCopyOption.REPLACE_EXISTING);
            System.out.printf("path:%s  size:%d%n", pathF.getFileName(), Files.size(pathF));
            listK = listK.subList(0, 2);
            pathR = Files.write(pathF, listK,
                                Charset.forName("KOI8-R"), StandardOpenOption.TRUNCATE_EXISTING);
            System.out.printf("path:%s  size:%d%n", pathF.getFileName(), Files.size(pathF));
    // checkout
            System.out.printf("%s%n", new String(Files.readAllBytes(pathR), Charset.forName("KOI8-R")));

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(br, bw, in, out);
        }
  ```
-
  ```

Write Large Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Write Large Files прочитать на базе BufferedWriter
    - Files.newBufferedWriter(Path,Charset,Options) открыть буфер на запись для больших файлов
    - Files.newOutputStream(Path,Options) открыть поток на запись для больших файлов
    - Files.newOutputStream(Path, StandardOpenOption.DELETE_ON_CLOSE) удалить при закрытии потока
- Пример. реализация **java8_nio/nio2/Main02D**
-

```
        System.out.printf(FORMAT, "Write Large Files :");
        path = Paths.get(".", "data", "nio");
        pathD = Paths.get(path.toString(), "result_u.txt");
        pathE = Paths.get(path.toString(), "result_u_large.txt");
        br = null; bw = null; in = null; out = null;
        try {
// source
            URL url = pathD.toUri().toURL();
            br = new BufferedReader(new InputStreamReader(url.openStream(), StandardCharsets.UTF_8));
            br.lines().forEach(s -> System.out.printf("%s%n", s));
            br.close();
// new file
            br = new BufferedReader(new InputStreamReader(url.openStream(), StandardCharsets.UTF_8));
            Files.deleteIfExists(pathE);
            bw = Files.newBufferedWriter(pathE, StandardCharsets.UTF_8, StandardOpenOption.CREATE_NEW);
            String str;
            while ((str = br.readLine()) != null) {
                bw.write(str);
                bw.newLine();
            }
            bw.flush();  // обязательно перед чтением файла
            bw.close(); br.close();
            br = Files.newBufferedReader(pathE, StandardCharsets.UTF_8);
            br.lines().forEach(s -> System.out.printf("%s%n", s));
            br.close();
// copy
            pathR = Files.createTempFile(Paths.get("./data/temp"), "tmp", null); // dst
            Files.copy(pathE, pathR, StandardCopyOption.REPLACE_EXISTING); // temp file loaded
// write to tmp autodelete
            pathD = Paths.get(path.toString(), "result_k.txt");              // src
            br = Files.newBufferedReader(pathD, Charset.forName("KOI8-R"));         // dst
            out = Files.newOutputStream(pathR, StandardOpenOption.DELETE_ON_CLOSE);
            bw = new BufferedWriter(new OutputStreamWriter(out, StandardCharsets.UTF_8));
            while ((str = br.readLine()) != null) {
                bw.write(str);
                bw.newLine();
            }
            bw.flush();  // обязательно перед копированием незакрытого файла
// copy back
            Files.copy(pathR, pathE, StandardCopyOption.REPLACE_EXISTING); // large file loaded
            bw.close(); br.close();
            br = Files.newBufferedReader(pathE, StandardCharsets.UTF_8); // checkout
            br.lines().forEach(s -> System.out.printf("%s%n", s));
            br.close();
// temp
            System.out.printf(FORMAT,"Write Truncated:");
            pathE = Paths.get(path.toString(),"result_truncated.txt");
            Files.copy(pathD, pathE, StandardCopyOption.REPLACE_EXISTING);
            System.out.printf("path:%s  size:%d%n", pathE.getFileName(), Files.size(pathE));
            out = Files.newOutputStream(pathE,  StandardOpenOption.TRUNCATE_EXISTING);
            System.out.printf("path:%s  size:%d%n", pathE.getFileName(), Files.size(pathE));
            String s = "Truncated строка в KOI8-R";
            out.write(s.getBytes(Charset.forName("KOI8-R")));
            out.close();
            System.out.printf("path:%s  size:%d%n", pathE.getFileName(), Files.size(pathE)); // checkout
            in = Files.newInputStream(pathE);
            System.out.printf("UTF-8 :%s%n", new String(in.readAllBytes(), StandardCharsets.UTF_8));
            in.close();
            in = Files.newInputStream(pathE);
            System.out.printf("KOI8-R:%s%n", new String(in.readAllBytes(), Charset.forName("KOI8-R")));
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(br, bw, in, out);
        }
```

Random Access File NIO2

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Random Access File NIO2 новые методы для произвольного доступа к файлу
    - FileChannel.open() возвращает FileChannel канал к файлу то есть это уже NIO2
    - FileChannel.position() возвращает SeakableByteChannel канал подключенный к файлу
    - FileChannel .truncate(long) возвращает SeakableByteChannel канал подключенный к файлу
    - SeekableByteChannel .read(ByteBiffer) возвращает число прочитанных в ByteBuffer байт
    - если задать новую позицию данные перечитываются как из массива
    - SeekableByteChannel .write(ByteBiffer) возвращает число записанных из ByteBuffer байт
    -
- Пример. реализация получение FileAttribute **java8_nio/nio2/Main03F**
- ```
  public static FileAttribute<List<AclEntry>> getFileAttribute(final Path path) {
      return new FileAttribute<>() {
          @Override
          public String name() {
              return "acl:acl";
          }

          @Override
          public List<AclEntry> value() {
              try {
                  final List<AclEntry> list = new ArrayList<>();
                  for (Object o : (List) Files.getAttribute(path, "acl:acl")) {
                      AclEntry aclEntry = (AclEntry) o;
                      list.add(aclEntry);
                  }
                  return list;
              } catch (IOException | UnsupportedOperationException e) {
                  System.out.printf("Exception:%s%n", e);
              }
              return null;
          }
      };
  }
  ```

- Пример. реализация SeekableByteChannel output **java8_nio/nio2/Main03F**
- ```
  public static int outSeekableChannel(SeekableByteChannel sc, Charset charset) throws IOException {
      ByteBuffer b = ByteBuffer.allocate(RECORD_LEN);
      ByteArrayOutputStream bout = new ByteArrayOutputStream();
      int len;
      while ((len = sc.read(b)) > 0) {
          b.flip();                                   // ограничивает реальным числом байт
          bout.write(b.array(), 0, len);
          b.clear();
      }
      b = ByteBuffer.wrap(bout.toByteArray());
      System.out.printf("%s", charset.decode(b));

      return b.array().length;
  }
  ```
- Пример. реализация SeekableByteChannel Truncate **java8_nio/nio2/Main03F**
- ```
  // read at pos 0 to ByteBuffer(sc.size)
      System.out.printf(FORMAT, "newByteChannel() EnumSet Truncate and Read All to ByteBuffer at position:");
      sc = Files.newByteChannel(pathE, StandardOpenOption.READ,StandardOpenOption.WRITE); // get Seekable ch
      sc.truncate(125);
      sc.position(0);
      b = ByteBuffer.allocate((int) sc.size());
      while ((len = sc.read(b)) > 0) {
           System.out.printf("%s", new String(b.array(), 0, len, StandardCharsets.UTF_8));
          b.clear();
      }
      System.out.printf("%n");
      len = b.array().length;
      System.out.printf("buffer length:%d%n", len);
      sc.close();
  ```
-
  ```

Random Access File NIO2 Example

- Random Access File NIO2 Example
- Пример. реализация Random Access with SeekableByteChannel **java8_nio/nio2/Main03F**
-
```
        Path pathD = Paths.get((".", "data", "nio", "result_u.txt");
        Path pathE = Paths.get((".", "data", "nio"), "result_channel.txt");
        FileChannel fc = null;
        SeekableByteChannel sc = null;
        OpenOption[] options = new OpenOption[]{  StandardOpenOption.CREATE, StandardOpenOption.WRITE,
                StandardOpenOption.READ, StandardOpenOption.SYNC  };
        List<OpenOption> listOptions = Arrays.stream(options).collect(Collectors.toList());
        try {
            final int RECORD_LEN = 25;
            Files.deleteIfExists(pathE);              // delete and copy file
            Files.copy(pathD, pathE, StandardCopyOption.REPLACE_EXISTING);
// write at position
            System.out.printf(FORMAT, "Seekable Channel Write at position:");
            fc = FileChannel.open(pathE, options);
            sc = fc.position(RECORD_LEN * 3);                 // get SeekableByteChannel
            ByteBuffer b = ByteBuffer.wrap("<John Doe>".getBytes());
            System.out.printf("b:%d%n", b.array().length);
            sc.write(b);
// read at position
            sc.position(0);
            b = ByteBuffer.allocate(RECORD_LEN);
            System.out.printf("b:%d%n", b.array().length);
            ByteArrayOutputStream outByteStream = new ByteArrayOutputStream();
            int len;
            while ((len = sc.read(b)) > 0) {
                b.flip();                                 // ограничивает реальным числом байт
                outByteStream.write(b.array(), 0, len);
                b.clear();
            }
            b = ByteBuffer.wrap(outByteStream.toByteArray());
            System.out.printf("b:%d%n", b.array().length);
            System.out.printf("%s", StandardCharsets.UTF_8.decode(b));
            sc.close();
// write at position with attributes
            System.out.printf(FORMAT, "newByteChannel() EnumSet and FileAttribute Read/Write at position:");
            FileAttribute<List<AclEntry>> fAttr = MainACL.getFileAttribute(pathE);
            fc = FileChannel.open(pathE, new HashSet<>(listOptions), fAttr);
            sc = fc.position(RECORD_LEN * 4);                 // get SeekableByteChannel
            b = ByteBuffer.wrap("Super John Link".getBytes());
            System.out.printf("b:%d%n", b.array().length);
            sc.write(b);
            sc.close();
// read at position with attributes
            System.out.printf(FORMAT, "newByteChannel() EnumSet and FileAttribute Read at position:");
            sc = Files.newByteChannel(pathE, EnumSet.of(StandardOpenOption.READ), fAttr); // get Seekable ch
            sc.position(0);
            len = FileUtils.outSeekableChannel(sc, StandardCharsets.UTF_8);
            System.out.printf("buffer length:%d%n", len);
            sc.close();
// read at pos to ByteArrayOutputStream
            System.out.printf(FORMAT, "newByteChannel() EnumSet Read at position:");
            sc = Files.newByteChannel(pathE, EnumSet.of(StandardOpenOption.READ)); // get Seekable channel
            sc = sc.position(RECORD LEN);
            len = FileUtils.outSeekableChannel(sc, StandardCharsets.UTF_8);
            System.out.printf("buffer length:%d%n", len);
            sc.close();
// read at pos 0 to ByteBuffer(sc.size)
            System.out.printf(FORMAT, "newByteChannel() EnumSet Read all to ByteBuffer at position:");
            sc = Files.newByteChannel(pathE, StandardOpenOption.READ); // get Seekable channel
            sc.position(0);
            b = ByteBuffer.allocate((int) sc.size());
            while ((len = sc.read(b)) > 0) {
                System.out.printf("%s", new String(b.array(), 0, len, StandardCharsets.UTF_8));
                b.clear();
            }
            System.out.printf("buffer length:%d%n", b.array().length);
        } catch (IOException e) {
        } finally {
            IOUtils.close(sc);
        }
```

Create Directory

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Create Directory создать каталоги
    - Files.createDirectory(Path) создает каталог с атрибутами по умолчанию
    - Files.createDirectory(Path,FileAttrinbute) создает каталог с заданными атрибутами ACL или Posix
    - Files.createDirectories(Path) создает цепочку каталогов, если они не существуют
- Пример. реализация создание каталогов ACL и POSIX **java8_nio/nio2/Main03F**

```
        System.out.printf(FORMAT, "Creating Directories:");
        path = Paths.get(".", "data", "nio2");
        pathC = Paths.get(path.toString(), "result.txt");
        pathD = Paths.get(path.toString(), "dir","net","work");
        pathE = Paths.get(path.toString(), "folder");

        boolean result = false;
        DosFileAttributeView dv = null;
        DosFileAttributes da = null;
        try {
            Files.createDirectories(pathD);
// file read only
            File folder = pathD.toFile();
            dv = Files.getFileAttributeView(pathD, DosFileAttributeView.class);
            dv.setReadOnly(true);
            System.out.printf("dir  readonly: %-25s exists:%b readOnly:%b%n", folder.toPath(),
                                                folder.exists(), dv.readAttributes().isReadOnly());
            Files.getFileAttributeView(pathD, DosFileAttributeView.class).setReadOnly(false);
            System.out.printf("dir  readonly: %-25s exists:%b readOnly:%b%n", folder.toPath(),
                                                folder.exists(), dv.readAttributes().isReadOnly());
            File file = pathC.toFile();
            result = file.setReadOnly();
            dv = Files.getFileAttributeView(pathC, DosFileAttributeView.class);
            System.out.printf("file readonly: %-25s exists:%b readOnly:%b%n", file.toPath(),
                                                file.exists(), dv.readAttributes().isReadOnly());
            dv.setReadOnly(false);
            System.out.printf("file readonly: %-25s exists:%b readOnly:%b%n", file.toPath(),
                                                file.exists(), dv.readAttributes().isReadOnly());
            System.out.printf(FORMAT, "Creating Directories Attributes:");
// attributes read only
            dv = Files.getFileAttributeView(pathD, DosFileAttributeView.class);
            da = Files.readAttributes(pathD, DosFileAttributes.class);
            System.out.printf("default: %s  read:%-5b hidden:%-5b system:%-5b archive:%-5b%n", folder,
                    da.isReadOnly(), da.isHidden(), da.isSystem(), da.isArchive());
            dv.setReadOnly(true);
            dv.setSystem(true);
            dv.setHidden(true);
            dv.setArchive(true);
            MainFileUtils.outAttributes(pathD, "set");
            MainFileUtils.setAttributes(pathD, true, false, false, false);
            MainFileUtils.outAttributes(pathD, "back");
// attributes
            fileAttr = MainFileUtils.getFileAttribute(path);
            Files.deleteIfExists(pathE);
            Files.createDirectory(pathE, fileAttr);
//posix
            System.out.printf(FORMAT, "Creating Directories Posix:");
            try {
                Path pathF = Paths.get(path.toString(), "posix");
                Files.deleteIfExists(pathF);
                posixAttr = MainFileUtils.getPosixAttribute("rw-------");
                Files.createDirectory(pathF, posixAttr);
            } catch (UnsupportedOperationException e) {
                System.out.printf("Exception:%s%n", e);
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
//            IOUtils.close(fc, sc);
        }
```

- Пример. реализация утилиты работы с DosFileAttribute атрибутами **java8_nio/nio2/Main03F**
- ```
  public static void setAttributes(Path path, boolean read, boolean hide, boolean sys, boolean arc) throws
  IOException {
      DosFileAttributeView dv = Files.getFileAttributeView(path, DosFileAttributeView.class);
      dv.setReadOnly(read);
      dv.setHidden(hide);
      dv.setSystem(sys);
      dv.setArchive(arc);
  }
  public static void outAttributes(Path path,String prefix)throws IOException {
      DosFileAttributeView dv = Files.getFileAttributeView(path, DosFileAttributeView.class);
      DosFileAttributes da = dv.readAttributes();
      System.out.printf("%-7s: %s  read:%-5b hidden:%-5b system:%-5b archive:%-5b%n", prefix,path,
              da.isReadOnly(), da.isHidden(), da.isSystem(), da.isArchive());
  }
  ```

Create Temp Directory

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Create Temp Dirtectory создать temp каталог
  - Files.createTempDirectory(Path, String) создать временный каталог
  - Files.createTempDirectory(Path, String,FileAttribute) создать временный каталог с атрибутами
- DeleteOnExit
  - File.deleteOnExit() задает удаление файла или каталога на выходе из программы
  - Runtime.addShutdownHook() задает Thread(Runtime) удаление файлов на выходе из программы
- Exception
  - Thread.currentThread().getStackTrace() возвращает массив StackTraceElement[]
  - Exception.setStackTrace() задает внешний StackTraceELement[] для Exception
  - Throwable.setStackTrace() задает внешний StackTraceELement[] для Throwable
  - new Exception(String,Throwable) создать свой Exception для выброса с заданным cause
  - Throwable.setStackTrace() задает свой стек для Exception
  - new Exception(String) создать свой собственный Exception
  - Exception.setStackTrace() задает свой стек для Exception
  - throw exception выбрасывает User Exception
- Пример. реализация **java8_nio/nio2/Main03F**
- ```
          System.out.printf(FORMAT, "Creating Temporary Directories:");
          pathD = Paths.get(".", "data", "nio", "temp");
          result = false;
          String prefix = "temp";
          try {
              if (Files.exists(pathD) && pathD.toFile().listFiles() != null) {
               result = Files.list(pathD).map(p -> p.toFile().delete()).reduce((b1, b2)->b1&b2).orElse(false);
                  if (!result) {
                      Throwable t = new Throwable();                              // with cause
                      t.setStackTrace(Thread.currentThread().getStackTrace());
  //                    throw new IOException("Can't delete temp directory",t);
  // with direct exception
                      IOException e = new IOException("Can't delete temp directory");
                      e.setStackTrace(Thread.currentThread().getStackTrace());
                      throw e;
                  }
              } else {
                  Files.createDirectory(pathD);
              }
  // temp dir
              Files.createTempDirectory(pathD, prefix);  // simple
              Files.createTempDirectory(pathD, prefix);
  // delete on Exit
              File[] files = pathD.toFile().listFiles();
              if (files == null) throw new IOException();
              for (File file : files) {
                  file.deleteOnExit();
              }
  ```

- Пример. реализация Create Temp Directory продолжение **java8_nio/nio2/Main03F**

```
// attributes
            List<Path> listTmp = new ArrayList<>();
            fileAttr = MainFileUtils.getFileAttribute(path);
            pathR = Files.createTempDirectory(pathD, prefix, fileAttr);  // with attributes
            listTmp.add(pathR);
            pathR = Files.createTempDirectory(pathD, prefix);
            listTmp.add(pathR);
            Thread hook = new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        for (Path path : listTmp) {
                            Files.deleteIfExists(path);
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
            Runtime.getRuntime().addShutdownHook(hook);
//posix
            System.out.printf(FORMAT, "Creating Temp Directories Posix:");
            try {
                List<Path> list = new ArrayList<>();
                posixAttr = MainFileUtils.getPosixAttribute("rw-------");
                pathR = Files.createTempDirectory(pathD, prefix, posixAttr);
                list.add(pathR);
                pathR = Files.createTempDirectory(pathD, prefix, posixAttr);
                list.add(pathR);
                for (Path path2 : list) {
                    path.toFile().deleteOnExit();
                }
            } catch (UnsupportedOperationException e) {
                System.out.printf("Exception:%s%n", e);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
//            IOUtils.close(fc, sc);
        }
```

Listing Directory

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Listing Directory прочитать список файлов в каталоге
  - Files.list() возвращает Stream<Path> как обычный поток, с которым можно работать
  - Files.newDirectoryStream(Path) возвращает Iterable<Path>
  - Files.newDirectoryStream(Path, Filter<Path>) возвращает Iterable<Path> по фильтру
  - Files.newDirectoryStream(Path, String glob) возвращает Iterable<Path> по фильтру Glob syntax
- Filter
  - Filter<Path> работает как стандартный Predicate<Path>
  - в качестве Matcher либо стандартный Regex и Pattern/Matcher
  - в качестве Matcher либо PathMatcher в режиме Regex или Glob Syntax

PathMatcher

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- PathMatcher тоже что Matcher только для Path
  - форматы поддерживает форматы REGEX и GLOB
  - regex PathMatcher("regex:"+pattern)
  - PathMatcher FileSystems.getDefault().getPathMatcher("glob:"+pattern
  - glob PathMatcher("glob:"+pattern)
  - PathMatcher FileSystems.getDefault().getPathMatcher("glob:"+pattern

Glob Syntax

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Glob Syntax применяется в качестве синтаксиса в PathMatcher
    - синтаксис регулярных выражений в Glob формате
    - * любое количество любых символов
    - ** тоже самое но по всему пути включая каталоги,не только имя файла
    - [abc] любой из символов a,b,c
    - {abc,demo, text*} слова abc,demo,text*
- Пример. реализация **java8_nio/nio2/Main03F**
-

```
// list dir
        System.out.printf(FORMAT, "Listing Directory list().stream:");
        pathD = Paths.get(".", "data", "nio");
        pathE = Paths.get(".", "data", "stream");
        Stream<Path> stream = null;
        DirectoryStream<Path> ds = null;
        final AtomicInteger aInt = new AtomicInteger(0);
        try {
            stream = Files.list(pathD);  // поток списка файлов
            aInt.set(0);
            stream.forEach(p -> {
                System.out.printf("%-25s ", p.getFileName());
                int i = aInt.incrementAndGet();
                if ((i + 1) % 5 == 0) System.out.printf("%n");
            });
            System.out.printf("%n");

            System.out.printf(FORMAT, "Listing new DirectoryStream().iterator:");
            ds = Files.newDirectoryStream(pathE);
            int count = 0;
            for (Path p : ds) {
                System.out.printf("%-25s ", p.getFileName());
                count++;
                if (count % 5 == 0) System.out.printf("%n");
            }
            System.out.printf("%n");

            System.out.printf(FORMAT, "Listing new DirectoryStream() Filter Regex:");
            String regex = "[rsb].*$";
            DirectoryStream.Filter<Path> f = p -> Pattern.compile(regex)
                                                    .matcher(p.getFileName().toString()).matches();
            ds = Files.newDirectoryStream(pathE, f);
            MainFileUtils.outStream(ds);
            ds.close();

            System.out.printf(FORMAT, "Listing new DirectoryStream() Filter PatternMatcher Regex:");
            DirectoryStream.Filter<Path> fr = p ->
                    FileSystems.getDefault().getPathMatcher("regex:" + regex).matches(p.getFileName());

            DirectoryStream<Path> dsR = Files.newDirectoryStream(pathE,fr);
            MainFileUtils.outStream(dsR);
            dsR.close();

            System.out.printf(FORMAT, "Listing new DirectoryStream() Filter PatternMatcher Glob:");
            String glob = "{bu,re,st}*.*";
            DirectoryStream.Filter<Path> fg = p ->
                    FileSystems.getDefault().getPathMatcher("glob:" + glob).matches(p.getFileName());
            DirectoryStream<Path> dsG = Files.newDirectoryStream(pathE, fg);
            MainFileUtils.outStream(dsG);
            dsG.close();

            System.out.printf(FORMAT, "Listing new DirectoryStream() Filter(String):");
            String globStr = "{bu,re,st}*.*";
            DirectoryStream<Path> dsS = Files.newDirectoryStream(pathE, globStr);
            MainFileUtils.outStream(dsS);
            dsS.close();

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(ds);
        }
```

- Пример. реализация утилиты вывода Stream<Path> и Iterable<Path> **java8_nio/nio2/Main03F**
- ```
  public static void outStream(Stream<Path> stream) {
      final UserInt value = UserInt.newInstance();
      StringBuilder sb = new StringBuilder();
      stream.forEach(p -> {
          sb.append(String.format("%-25s ", p.getFileName()));
          int i = value.increment();
          if (i % 5 == 0) sb.append(String.format("%n"));
      });
      if (value.getValue() % 5 != 0) sb.append(String.format("%n"));
      System.out.printf("%s", sb.toString());
  }
  public static void outStream(DirectoryStream<Path> ds) {
      int i = 0;
      Formatter f = new Formatter(Locale.ENGLISH);
      for (Path p : ds) {
          f.format("%-25s ", p.getFileName());
          i++;
          if (i % 5 == 0) f.format("%n");
      }
      if (i % 5 != 0) f.format("%n");
      f.flush();
      System.out.printf("%s", f.toString());
  }
  ```

Copy Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Copy Files копирование файлов
  - Files.copy(InputStream,Path,Options) копирование InputStream to Path, с CopyOptions
  - Files.copy(Path, OutputStream,Options) копирование Path to OutpuStream, с CopyOptions
  - Files.copy(Path,Path,Options) копирование Path to Path с CopyOptions
- Trick Java 5 способов скопировать файлы
- ВНИМАНИЕ. Похоже опция COPY_ATTRIBUTES на самом деле НЕ РАБОТАЕТ.
- Пример. реализация **java8_nio/nio2/Main03F**
- ```
          System.out.printf(FORMAT, "Copy Files :");
          path = Paths.get(".", "data", "nio2");
          pathC = Paths.get(path.toString(), "result_k.txt");
          pathD = Paths.get(path.toString(), "result_k_out_in_p.txt");
          pathE = Paths.get(path.toString(), "result_k_out_p_out.txt");
          pathR = Paths.get(path.toString(), "result_k_out_p_p.txt");

          BufferedReader br = null;
          BufferedWriter bw = null;
          InputStream in = null;
          OutputStream out = null;
          Charset charset = Charset.forName("KOI8-R");
          try {
  // copy in to p
              in = new FileInputStream(pathC.toString());
              Files.copy(in, pathD, StandardCopyOption.REPLACE_EXISTING);
              MainFileUtils.outPath(pathD,charset); // check
              pathD = Paths.get(path.toString(), "result_k_out_in_p_url.txt");
              Files.copy(pathC.toUri().toURL().openStream(), pathD, StandardCopyOption.REPLACE_EXISTING);
              MainFileUtils.outPath(pathD,charset); // check
  // copy p to out
              out = new FileOutputStream(pathE.toString());
              Files.copy(pathC, out);
              MainFileUtils.outPath(pathE,charset); // check
  // copy p to p
              Files.deleteIfExists(pathR);
              MainFileUtils.setAttributes(pathC,false,false,false,false);
              Files.copy(pathC, pathR,StandardCopyOption.REPLACE_EXISTING,StandardCopyOption.COPY_ATTRIBUTES);
              MainFileUtils.outAllAttributes(pathC,"original");
              MainFileUtils.outAllAttributes(pathR,"copy");
              MainFileUtils.outChannel(pathR,charset);
              MainFileUtils.outToChannel(pathR,charset);
          } catch (IOException e) {
          } finally {
              IOUtils.close(br, bw, in, out);
          }
  ```

- Пример. реализация утилиты вывода на печать потока и канала **java8_nio/nio2/Main03F**

```
    public static void outPath(Path path, Charset charset) throws IOException {

        FileInputStream in = new FileInputStream(path.toAbsolutePath().toString());
        BufferedReader br = new BufferedReader(new InputStreamReader(in, charset));
        Formatter f = new Formatter(Locale.ENGLISH);
        try {
            String s;
            while ((s = br.readLine()) != null) {
                f.format("%s%n", s);
            }
            f.flush();
            System.out.printf("%s", f.toString());
        } finally {
            IOUtils.close(br, in, f);
        }
    }
```

```
    public static void outToChannel(Path path, Charset charset) throws IOException {
        FileChannel fc = FileChannel.open(path, StandardOpenOption.READ);
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        WritableByteChannel wc = Channels.newChannel(out);
        Formatter f = new Formatter(Locale.ENGLISH);
        try {
            fc.transferTo(0,fc.size(),wc);
            f.format("%s",out.toString(charset));
            f.flush();
            System.out.printf("%s", f.toString());
        } finally {
            IOUtils.close(fc, wc, f, out);
        }
    }
    public static void outChannel(Path path, Charset charset) throws IOException {
        FileChannel fc = FileChannel.open(path, StandardOpenOption.READ);
        ByteArrayOutputStream out = new ByteArrayOutputStream();
//        BufferedOutputStream bout = new BufferedOutputStream(out);// it's not necessary
        WritableByteChannel wc = Channels.newChannel(out);          // channel is buffered by ByteBuffer
        ByteBuffer b = ByteBuffer.allocate(8);
        Formatter f = new Formatter(Locale.ENGLISH);
        try {
            while (fc.read(b) > 0) {
                b.flip();
                wc.write(b);
                b.compact();  // clear only data that was read
            }
            out.flush();
//*****************************
//            byte[] bytes = out.toByteArray();
//            String s = new String(bytes, charset);
//            f.format("%s%n", s);
//*****************************
            f.format("%s",out.toString(charset));
            f.flush();
            System.out.printf("%s", f.toString());
        } finally {
            IOUtils.close(fc, wc, f, out);
        }
    }
```

Moving Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Moving Files перемещение файлов
    - move(Path,Path, CopyOption) перемещает файлы между каталогами
- ВНИМАНИЕ. Все DirectoryStream<Path> ЗАКРЫВАТЬ ОБЯЗАТЕЛЬНО иначе держут каталоги до завершения
- ВНИМАНИЕ. Files.list() создает поток ЗАКРЫТЬ ОБЯЗАТЕЛЬНО блокирует каталог **java8_nio/nio2/Main03F**
- Пример. реализация утилиты удаления и вывода списка файлов **java8_nio/nio2/Main03F**
- ```
  public static void outFolder(Path path) throws IOException {
      DirectoryStream<Path> ds = null;
      if (!Files.exists(path)) return;
      try {
          ds = Files.newDirectoryStream(path);
          for (Path p : ds) {
              System.out.printf("%s%n", p);
          }
      } finally {
          IOUtils.close(ds);
      }
  }

  public static void deleteFolder(Path path) throws IOException {
      DirectoryStream<Path> ds = null;
      if (!Files.exists(path)) return;
      try {
          ds = Files.newDirectoryStream(path, "*");  // поток ЗАКРЫТЬ ОБЯЗАТЕЛЬНО
          for (Path p : ds)
              Files.deleteIfExists(p);
          Files.deleteIfExists(path);
      } finally {
          IOUtils.close(ds);
      }
  }

  public static void deleteFolderGlobe(Path path, String globe) throws IOException {
      DirectoryStream<Path> ds = null;
      if (!Files.exists(path)) return;
      DirectoryStream.Filter<Path> filter = p -> FileSystems.getDefault()
              .getPathMatcher("glob:" + globe)
              .matches(p.getFileName());
      try {
          ds = Files.newDirectoryStream(path, filter);  // поток ЗАКРЫТЬ ОБЯЗАТЕЛЬНО
          for (Path p : ds)
              Files.deleteIfExists(p);
          Files.deleteIfExists(path);
      } finally {
          IOUtils.close(ds);
      }
  }

  public static void deleteFolderRegex(Path path, String regex) throws IOException {
      DirectoryStream<Path> ds = null;
      if (!Files.exists(path)) return;
      DirectoryStream.Filter<Path> fr = p -> FileSystems.getDefault()
              .getPathMatcher("regex:" + regex)
              .matches(p.getFileName());
      try {
          ds = Files.newDirectoryStream(path, fr);  // поток ЗАКРЫТЬ ОБЯЗАТЕЛЬНО
          for (Path p : ds)
              Files.deleteIfExists(p);
          Files.deleteIfExists(path);
      } finally {
          IOUtils.close(ds);
      }
  }
  ```
-
  ```

- Пример. реализация Moving FIles **java8_nio/nio2/Main03F**

```
        System.out.printf(FORMAT, "Moving Files and Directories:");
        path = Paths.get(".", "data", "nio2");
        pathC = Paths.get(".", "data", "nio2", "move");
        pathD = Paths.get(path.toString(), "result_w.txt");
        pathE = Paths.get(pathC.toString(), "result_w.txt");
        pathR = Paths.get(".", "data", "nio2", "lost");
        try {
            path = Paths.get(".", "data", "nio", "result_w.txt");
            Files.copy(path.resolve(pathD.getFileName()), pathD);

            if (!Files.exists(pathC)) {
                Files.createDirectory(pathC);  // create
            }
            Files.move(pathD, pathE, StandardCopyOption.REPLACE_EXISTING);
            MainFileUtils.outToChannel(pathE);
            MainFileUtils.outToChannel(pathE, Charset.forName("WINDOWS-1251"));
// move dir
            MainFileUtils.deleteFolder(pathR);
            Files.move(pathC, pathR, StandardCopyOption.REPLACE_EXISTING);
// ATTENTION close Stream
            Stream<Path> sp = Files.list(pathR);
            sp.forEach(s -> System.out.printf("%s%n", s));
            sp.close();
            MainFileUtils.outFolder(pathR); // Attention CLOSE STREAM
            MainFileUtils.outChannel(pathR.resolve(path.getFileName()), Charset.forName("WINDOWS-1251"));
            MainFileUtils.deleteFolderRegex(pathR,".*");
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Delete Files

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Delete Files удаление файлов и каталогов
    - Files.delete(Path) удалить файл или каталог
    - если файла или каталога нет выбросить NoSuchFileException
    - Files.deleteIfExists(Path) удалить если существует
    - для каталогов в которых есть файлы выбросит DirectoryNotEmptyException
    - перед удалением каталога надо почистить его от файлов
- Пример. реализация удаление файлов и чистка и удаление каталога **java8_nio/nio2/Main03F**

```
System.out.printf(FORMAT, "Delete Files :");
path = Paths.get(".", "data", "nio2");
pathC = Paths.get(".", "data", "nio2", "lost");
pathD = Paths.get(path.toString(), "result.txt");
pathE = Paths.get(pathC.toString(), "result.txt");

try {
    if (!Files.exists(pathC)) {
        Files.createDirectory(pathC);
    }
    if (Files.exists(pathE) && (Boolean) Files.getAttribute(pathE, "dos:readonly")) {
        Files.getFileAttributeView(pathE, DosFileAttributeView.class).setReadOnly(false);
    }

    Files.copy(pathD, pathE, StandardCopyOption.REPLACE_EXISTING);

    if ((Boolean) Files.getAttribute(pathE, "dos:readonly")) {
        Files.getFileAttributeView(pathE, DosFileAttributeView.class).setReadOnly(false);
    }
    System.out.printf("path: %-32s  exist:%b%n", pathC, Files.exists(pathC));
    System.out.printf("path: %-32s  exist:%b%n", pathE, Files.exists(pathE));
    MainFileUtils.deleteFolderGlobe(pathC,"*");
    System.out.printf("path: %-32s  exist:%b%n", pathC, Files.exists(pathC));
    System.out.printf("path: %-32s  exist:%b%n", pathE, Files.exists(pathE));

} catch (IOException e) {
    e.printStackTrace();
} finally {
    IOUtils.close(br, bw, in, out);
}
```

Symbolic and Hard Links

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Symbolic and Hard Links ссылки на файлы Symbolic и Hard
    - Files.createSymbolicLink(path,path) создать Symbolic link ссылку
    - Windows создается но не поддерживается
    - Files.createLink(path,path) создать Hard Link ссылку,
    - НЕ РАБОТАЕТ с каталогами
    - поддерживается и работает как файл
    - не отличима от оригинального файла
- ВНИМАНИЕ. Для запуска зайти в Admin Console и запустить с командной строки
- Пример. реализация **java8_nio/nio2/Main03F**
- ```
  // Run in Administrative mode
  // Run Admin console
  //D:__cources_sandbox\java_questions>java -cp out/production/java_nio nio2.links.MainLinks
          System.out.printf(FORMAT, "Symbolic link:");
          Path path = Paths.get(".", "data", "nio2");
          Path pathC = Paths.get(".", "data", "nio2","links");
          Path pathD = Paths.get(path.toString(), "result.txt");
          Path pathE = Paths.get(pathC.toString(), "result_link_s.txt");
          Path pathR = Paths.get(pathC.toString(), "result_link_h.txt");
          try {
              if(Files.exists(pathC)) {
                  MainFileUtils.deleteFolderRegex(pathC,".*");
                  Files.createDirectory(pathC);
              }
              Files.copy(pathD,pathC.resolve(pathD.getFileName()),StandardCopyOption.REPLACE_EXISTING);
  // file links
              System.out.printf(FORMAT,"File links:");
              Files.createSymbolicLink(pathE,pathD);
              Files.createLink(pathR,pathD);
              System.out.printf("path:%-32s  exists:%b  symbolic:%b  regular:%b%n",pathE,
                      Files.exists(pathE), Files.isSymbolicLink(pathE),Files.isRegularFile(pathE));

              System.out.printf("path:%-32s  exists:%b  symbolic:%b  regular:%b%n",pathR,
                      Files.exists(pathR), Files.isSymbolicLink(pathR),Files.isRegularFile(pathR));

              System.out.printf(FORMAT,"Directory links:");
              path = Paths.get(".", "data", "nio2");
              pathC = Paths.get(".", "data", "nio2","links");
              pathD = Paths.get(path.toString(), "move");
              pathE = Paths.get(pathC.toString(), "move_link_s");
              pathR = Paths.get(pathC.toString(), "move_link_h");

              if(!Files.exists(pathD)) Files.createDirectory(pathD);

              System.out.printf("pathD:%-32s  exists:%b%n",pathD,Files.exists(pathD));
              System.out.printf("pathE:%-32s  exists:%b%n",pathE,Files.exists(pathE));
              System.out.printf("pathR:%-32s  exists:%b%n",pathR,Files.exists(pathR));

              System.out.printf("Create symbolic:%n");
              Files.createSymbolicLink(pathE,pathD);
              System.out.printf("sym  :%-32s  exists:%b%n",pathE,Files.exists(pathE));
              try {
                  System.out.printf("Create hard:%n");
                  Files.createLink(pathR, pathD);
                  System.out.printf("hard :%-32s  exists:%b%n", pathR, Files.exists(pathR));
              }catch (AccessDeniedException e) {
                  System.out.printf("Exception Hard Link not Allowed on Folder:%s%n",e);
              }
              System.out.printf("path:%-12s  exists:%b  symbolic:%b  regular:%b%n",pathE.getFileName(),
                      Files.exists(pathE), Files.isSymbolicLink(pathE),Files.isRegularFile(pathE));

              System.out.printf("path:%-12s  exists:%b  symbolic:%b  regular:%b%n",pathR.getFileName(),
                      Files.exists(pathR), Files.isSymbolicLink(pathR),Files.isRegularFile(pathR));

          } catch (IOException e) {
              System.out.printf("Exception:%s%n",e);
          }
      }
  ```
  ```

File Tree Walk

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- File Tree Walk работа с деревом каталогов
    - Files.walkFileTree(path, SimpleVisitor) метод обхода дерева каталогов
    - производится методом Depth For Search то есть в глубину
- SimpleVisitor<Path> класс который отрабатывает методы прохода Tree
    - для работы надо создать потомка и переопределить методы
    - напрямую SimpleVisitor использовать не получится, у него protected Constructor
    - preVisitDirectory(Path dir, BasicFileAttributes attrs) вызывается до входа в каталог
    - postVisitDirectory(Path dir, IOException exc) после визита всей иерархии содержимого
    - visitFile(Path file, BasicFileAttributes attrs) вызывается для каждого файла
    - visitFileFailed(Path file, IOException exc) вызывается при встрече ошибок
- FileVisitResults
    - CONTINUE нормальный проход, продолжать
    - SKIP_SIBLINGS не делать обход на том же уровне
    - SKIP_SUBTREE не делать обход содержимого каталога
    - TERMINATE завершить обход
- Процедура
    - создать класс потомок SimpleVisitor
    - определить методы preVisitDirectory(), postVisitDirectory(), visitFile(), visitFileFailed()
    - именно эти методы позволяют задать что будет делать walkFileTree()
    - сбор и вывод данных, копирование или перемещение файлов и каталогов
    - вызвать метод Files.walkFileTree()
- Пример. реализация утилиты для WalkTree **java8_nio/nio2/Main04T**
- ```
  public static void createTree(Path path) throws IOException {
      if (!Files.exists(path)) {
          Files.createDirectory(path);
      }

      Path pathC = path.getParent();
      Path pathD = Paths.get(pathC.toString(), "result.txt");
      Path pathE = Paths.get(pathC.toString(), "result_k.txt");
      Path pathR;

      pathR = Paths.get(path.toString(), "walk");
      if (!Files.exists(pathR)) Files.createDirectory(pathR);
      Files.copy(pathD, pathR.resolve(pathD.getFileName()), StandardCopyOption.REPLACE_EXISTING);
      Files.copy(pathE, pathR.resolve(pathE.getFileName()), StandardCopyOption.REPLACE_EXISTING);

      pathR = Paths.get(path.toString(), "wmode");
      if (!Files.exists(pathR)) Files.createDirectory(pathR);
      Files.copy(pathD, pathR.resolve(pathD.getFileName()), StandardCopyOption.REPLACE_EXISTING);
      Files.copy(pathD, pathR.resolve("result_k.txt"), StandardCopyOption.REPLACE_EXISTING);

      pathR = Paths.get(path.toString(), "wnet");
      if (!Files.exists(pathR)) Files.createDirectory(pathR);
  }

  public static void removeTree(Path path) throws IOException {
      if (!Files.exists(path)) return;
      MainFileUtils.deleteFolder(path);  // recursive
  }
  ```
  ```

File Tree Walk Example

- Пример. реализация ViewVisitor класс для WalkTree **java8_nio/nio2/Main04T**

```
public class ViewVisitor extends SimpleFileVisitor<Path> {
        @Override
        public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) throws IOException {
            LocalDateTime time = LocalDateTime.ofInstant(attrs.lastModifiedTime()
                                              .toInstant(),ZoneId.systemDefault());
            System.out.printf("pre dir    :%1$-40s modified : %2$tD %2$tT  size:%3$d %n",
                                              dir,time,attrs.size());
            return super.preVisitDirectory(dir, attrs);
        }
        @Override
        public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
            LocalDateTime time = LocalDateTime.ofInstant(attrs.lastModifiedTime()
                                              .toInstant(),ZoneId.systemDefault());
            System.out.printf("visit file :%1$-40s modified : %2$tD %2$tT  size:%3$d %n",
                                              file,time,attrs.size());
            return super.visitFile(file, attrs);
        }
        @Override
        public FileVisitResult visitFileFailed(Path file, IOException exc) throws IOException {
            System.out.printf("failed file: %-40s : %-40s %n",file,exc);
            return super.visitFileFailed(file, exc);
        }
        @Override
        public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws IOException {
            System.out.printf("post dir   :%-40s exception: %-40s %n",dir,exc);
            return super.postVisitDirectory(dir, exc);
        }
    }
```

- Пример. реализация Read WalkTree **java8_nio/nio2/Main04T**

```
        System.out.printf(FORMAT, "Walk Tree:");
        Path path = Paths.get(".", "data", "nio2");
        Path pathD = Paths.get(path.toString(), "result.txt");
        Path pathE = Paths.get(path.toString(), "result_k.txt");
        Path pathR;

        try {
            WalkUtils.createTree(path.resolve("walktree"));
            Files.walkFileTree(path.resolve("walktree"), new ViewVisitor());
//            WalkUtils.removeTree(path.resolve("walktree"));

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

File Tree Walk Copy Example

- Пример. реализация CopyVisitor класс для WalkTree **java8_nio/nio2/Main04T**

```
public class CopyVisitor extends SimpleFileVisitor<Path> {
    private static final StandardCopyOption option = StandardCopyOption.REPLACE_EXISTING;
    private Path fromPath;
    private Path toPath;
    public CopyVisitor(Path fromPath, Path toPath) {
        this.fromPath = fromPath;
        this.toPath = toPath;
        System.out.printf("from:%-40s to:%-40s%n", fromPath, toPath);
    }
    @Override
    public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) throws IOException {
        Path pSrc = fromPath.relativize(dir);
        Path pDst = toPath.resolve(pSrc);
        System.out.printf("dir:%-40s pSrc:%-40s pDst:%-40s%n", dir, pSrc, pDst);
        if (!Files.exists(pDst)) {
            Files.createDirectories(pDst);
        }
        return FileVisitResult.CONTINUE;
    }
    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
        Path pSrc = fromPath.relativize(file);
        Path pDst = toPath.resolve(pSrc);
        System.out.printf("file:%-40s pSrc:%-40s pDst:%-40s%n", file, pSrc, pDst);
        Files.copy(file, pDst, option);
        return FileVisitResult.CONTINUE;
    }
    @Override
    public FileVisitResult visitFileFailed(Path file, IOException exc) throws IOException {
        System.err.printf("%s%n", exc);
        return FileVisitResult.CONTINUE;
    }
    @Override
    public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws IOException {
        Path pSrc = fromPath.relativize(dir);
        Path pDst = toPath.resolve(pSrc);
        if (exc != null) throw exc;
        FileTime lastModifiedTime = Files.getLastModifiedTime(dir);
        Files.setLastModifiedTime(pDst, lastModifiedTime);
        return FileVisitResult.CONTINUE;
    }
}
```

- Пример. реализация Copy WalkTree **java8_nio/nio2/Main04T**

```
        System.out.printf(FORMAT, "Copy Tree:");
        path = Paths.get(".", "data", "nio2");
        pathD = Paths.get(path.toString(), "walktree");
        pathE = Paths.get(path.toString(), "temp","copy","walktree2");
        try {
            WalkUtils.createTree(pathD);
// source not exists
            if (!Files.exists(pathD)) {
                IOException ex = new IOException(String.format("path:%s Source not Exists%n", pathD));
                ex.setStackTrace(Thread.currentThread().getStackTrace());
                throw ex;
            }
// source is file, dest not exists or exists file or folder
            if (!Files.isDirectory(pathD)) {
                if (Files.exists(pathE) && Files.isDirectory(pathE)) {
                    pathE = pathE.resolve(path.getFileName());              // dest/source_file
                }
                Files.copy(path, pathE, StandardCopyOption.REPLACE_EXISTING);
            }
// source is folder dest exists file
            if (Files.exists(pathE) && !Files.isDirectory(pathE)) {
                IOException ex = new IOException(String.format("path:%s Target is File%n", pathE));
                ex.setStackTrace(Thread.currentThread().getStackTrace());
                throw ex;
            }
            EnumSet<FileVisitOption> set = EnumSet.of(FileVisitOption.FOLLOW_LINKS);
            CopyVisitor copyVisitor = new CopyVisitor(pathD, pathE);
            Files.walkFileTree(pathD, set, Integer.MAX_VALUE, copyVisitor);
        } catch (IOException e) {
        }
```

File Tree Walk Delete Example

- Пример. реализация DeleteVisitor класс для WalkTree **java8_nio/nio2/Main04F**
- ```
  public class DeleteVisitor extends SimpleFileVisitor<Path> {

      @Override
      public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) throws IOException {
          return FileVisitResult.CONTINUE;
      }

      @Override
      public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
            if(Files.exists(file)){
              if(Files.deleteIfExists(file)) {
                  System.out.printf("file:%-40s  deleted", file);
              }else {
                  throw new IOException(String.format("can't delete file:%s",file));
              }
          }
          return FileVisitResult.CONTINUE;
      }

      @Override
      public FileVisitResult visitFileFailed(Path file, IOException exc) throws IOException {

          System.err.printf("%s%n",exc);
          return FileVisitResult.CONTINUE;
      }

      @Override
      public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws IOException {
          if(exc != null) throw exc;

          if(Files.exists(dir)){
              if(Files.deleteIfExists(dir)) {
                  System.out.printf("dir :%-40s  deleted", dir);
              }else {
                  throw new IOException(String.format("can't delete folder:%s",dir));
              }
          }
           return super.postVisitDirectory(dir, exc);
      }
  }
  ```
- Пример. реализация Delete WalkTree **java8_nio/nio2/Main04T**
- ```
  System.out.printf(FORMAT, "Delete Tree:");
  path = Paths.get(".", "data", "nio2");
  pathD = Paths.get(path.toString(), "walktree");
  pathE = Paths.get(path.toString(), "temp","copy","walktree2");

  try {
      if(!Files.exists(pathD)) {
          WalkUtils.createTree(pathD);
      }

      EnumSet<FileVisitOption> set = EnumSet.of(FileVisitOption.FOLLOW_LINKS);
      DeleteVisitor deleteVisitor = new DeleteVisitor();
      Files.walkFileTree(pathD, set, Integer.MAX_VALUE, deleteVisitor);

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

File Tree Walk Move Example

- Пример. реализация MoveVisitor класс для WalkTree **java8_nio/nio2/Main04T**

```
public class MoveVisitor extends SimpleFileVisitor<Path> {
    private static final StandardCopyOption option = StandardCopyOption.REPLACE_EXISTING;
    private Path fromPath;
    private Path toPath;

    public MoveVisitor(Path fromPath, Path toPath) {
        this.fromPath = fromPath;
        this.toPath = toPath;
        System.out.printf("from:%-40s to:%-40s%n", fromPath,toPath);
    }

    @Override
    public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) throws IOException {
        Path pSrc = fromPath.relativize(dir);
        Path pDst = toPath.resolve(pSrc);

        Files.copy(dir,pDst,StandardCopyOption.REPLACE_EXISTING,StandardCopyOption.COPY_ATTRIBUTES);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
        Path pSrc = fromPath.relativize(file);
        Path pDst = toPath.resolve(pSrc);
        Files.move(file,pDst,StandardCopyOption.REPLACE_EXISTING,StandardCopyOption.ATOMIC_MOVE);

        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult visitFileFailed(Path file, IOException exc) throws IOException {

        System.err.printf("%s%n",exc);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws IOException {
         if(exc != null) throw exc;
            if(!Files.deleteIfExists(dir)){
                throw new IOException(String.format("dir :%s can't delete folder%n",dir));
            }
         return FileVisitResult.CONTINUE;
    }
}
```

- Пример. реализация Move WalkTree **java8_nio/nio2/Main04T**

```
System.out.printf(FORMAT, "Move Tree:");
path = Paths.get(".", "data", "nio2");
pathD = Paths.get(path.toString(), "temp","walktree");
pathE = Paths.get(path.toString(), "temp","copy","walktree2");

try {
    if(!Files.exists(pathE)) {
        WalkUtils.createTree(pathE);
    }

    EnumSet<FileVisitOption> set = EnumSet.of(FileVisitOption.FOLLOW_LINKS);
    MoveVisitor moveVisitor = new MoveVisitor(pathE,pathD);
    Files.walkFileTree(pathE, set, Integer.MAX_VALUE, moveVisitor);

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

-

Files Streams Methods

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Methods методы порождающие stream потоки
    - Files.find() создает stream файлов, подходящих под заданный поиск
    - Files.lines(Path) создает Stream<String> поток строк прочитанных с файла
    - Files.lines(Path,Chipset) тоже самое в кодировке
    - Files.list(Path) создает поток Stream<Path> для файлов каталога
    - Files.walk() создает поток Stream<Path> для файлов каталога и подкаталогов с глубиной
- ВНИМАНИЕ. Все streams открытые для каталога ОБЯЗАТЕЛЬНО ЗАКРЫТЬ,так как блокируют каталог до выхода
    - ГЛАВНОЕ если до выхода изменить каталог, приложение выбросит Exception, и каталог не удалить
    - вообще даже в проводнике
- Пример. реализация **java8_nio/nio2/Main04T**
- ```
  System.out.printf(FORMAT, "Files Stream Methods:");
  path = Paths.get(".", "data", "nio2");
  pathD = Paths.get(path.toString(), "result.txt");
  pathE = Paths.get(path.toString(), "result_k.txt");

  Stream<String> ss = null;
  Stream<Path> sp = null;

  String regex = "(.*_k.*)|(t.*)";
  try {
      System.out.printf(FORMAT, "Files.find():");
      BiPredicate<Path, BasicFileAttributes> matcher = (p, a) -> a.isRegularFile() &&
              p.getFileName().toString().matches("(.*_k.*)|(t.*)");
      sp = Files.find(path, 10, matcher);
      sp.forEach(p -> System.out.printf("path:%s%n", p));
      sp.close();

      System.out.printf(FORMAT, "Files.lines():");
      ss = Files.lines(pathE, Charset.forName("KOI8-R"));
      ss.forEach(s -> System.out.printf("path:%s%n", s));
      ss.close();

      System.out.printf(FORMAT, "Files.list():");

      sp = Files.list((path));
      sp.filter(p -> p.getFileName().toString().matches(regex))
              .forEach(p -> System.out.printf("path:%s%n", p));
      sp.close();

      System.out.printf(FORMAT, "Files.walk():");
      Predicate<Path> predicate = p -> p.getFileName().toString().matches(regex);

      sp = Files.walk(path);
      sp.forEach(p -> System.out.printf("path:%-60s  name:%-32s match:%b%n", p, p.getFileName(), predicate));
      sp.close();
      System.out.printf("%n");
      System.out.printf(FORMAT, "Files.walk() predicate:");
      sp = Files.walk(path);
      sp.filter(predicate). forEach(p ->
              System.out.printf("path:%-60s  name:%-32s match:%b%n", p, p.getFileName(),predicate));
      sp.close();

  } catch (IOException e) {
      e.printStackTrace();
  } finally {
      IOUtils.close(ss, sp);
  }
  ```
-
  ```

Watching Directories

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Watching Directories отслеживание событий с каталогами
    - Watchable интерфейс отслеживаемого объекта
    - WatchEvent<T> интерфейс отслеживаемого события
    - WatchEvent<T>.Kind вложенный интерфейс вида события
    - WatchKey интерфейс ключа при регистрации объекта
    - WatchService интерфейс службы отсдеживания событий
- Методы
    - FilSystems.getDefault() получить объект FileSystem по умолчанию
    - FileSystem.newWatchService() получить службу отслеживания событий
    - FileSystem.getPath(String) получить объект Path для заданного каталога
    - Path.register() зарегистрировать каталог в службе и получить WatchKey
    - WatchService.take() получить WatchKey зарегистрированный в службе
    - WatchKey.pollEvents() получить объект Iterable<WatchEvent> регистрированных событий
    - WatchKey.reset() сбросить события WatchEvent после обработки WatchKey
- ВНИМАНИЕ. Paths.get() это СОКРАЩЕННАЯ форма метода FileSystems.getDefault().getPath()
- StandardWatchEventKinds события отслеживаемые
    - ENTRY_CREATE создание объекта
    - ENTRY_DELETE удаление объекта
    - ENTRY_MODIFY изменение объекта
    - OVERFLOW событие потеряно
- Процедура
    - создать WatchService службу, которая будет отслеживать один или несколько каталогов
    - регистрация каждого каталога, который будет отслеживаться
    - при регистрации задаются события WatchEvent.Kind, и возвращается WatchKey
    - ожидание реализовать петлю ожидания событий, при обнаружении получить Watchkey
    - обработка получить все ключи, соответствующие событиям, получить событие и обработать
    - сбросить ключи и перейти к пункту ожидание
    - завершение закрыть службу отслеживания
- ВНИМАНИЕ. ОСНОВНЫЕ МОМЕНТЫ работы с каталогами
    - показывает изменения только на первом уровне каталога даже если изменения в глубине
    - показывает изменения атрибутов только каталогов первого уровня
    - файлы вообще здесь не работают, не видно создания или удаления файлов
    - удаление одного из каталогов которые отслеживаются делает key нерабочим
- ВНИМАНИЕ. Если WatchKey стал isValid()=false он уже НИКОГДА НЕ БУДЕТ получать WatchEvent и нужна
- полная перерегистрация WatchService и всех каталогов
- Пример. реализация **java8_nio/nio2/Main04T**

```
System.out.printf(FORMAT, "Folder Watchers:");
 path = Paths.get(".", "data", "nio2");
 pathD = Paths.get(path.toString(), "result.txt");
 pathE = Paths.get(path.toString(), "result_k.txt");
 try {
     Runtime.getRuntime().exec("cmd /c start call java -ea -cp " +
             "out/production/java_nio nio2.watchers.MainFolder");
     MainWatch.main(args);
 } catch (IOException e) {
     e.printStackTrace();
 }
```

Watching Directories Example

- Watching Directories Example
  - watchers/MainWatch — отслеживает работу двух каталогов temp и walkfolder
  - изменения на первом уровне видны, на нижних нет
  - watchers/MainFolder — собственно производит изменения с каталогами
  - запуск — модулей идет на уровне JVM как двух разных независимых приложений
- Пример. реализация **java8_nio/nio2/Main04T**

```
    public static void main(String[] args) {
        System.out.printf(FORMAT, "Folder Watchers:");

        Path path = Paths.get(".", "data", "nio2");
        Path pathD = Paths.get(path.toString(), "temp", "copy", "walktree");
        Path pathE = Paths.get(path.toString(), "temp", "walktree");
        Path pathR = path.resolve("walkfolder");
        WatchService ws = null;

        try {
            WalkUtils.createUnderTree(pathD, path);
            WalkUtils.createUnderTree(pathR, path);
            WalkUtils.removeIfTree(pathE);


            WatchEvent.Kind[] kinds = new WatchEvent.Kind[]{
                    StandardWatchEventKinds.ENTRY_CREATE,
                    StandardWatchEventKinds.ENTRY_MODIFY,
                    StandardWatchEventKinds.ENTRY_DELETE,
                    StandardWatchEventKinds.OVERFLOW
            };
            ws = FileSystems.getDefault().newWatchService();
        //    path.register(ws, kinds);      // nio2
            pathE.getParent().register(ws, kinds);      // temp
            pathR.register(ws, kinds);                  // walkfolder

            while (true) {
                WatchKey key;
                key = ws.take();
                for (WatchEvent event : key.pollEvents()) {
                    Path p = (Path) event.context();
                    WatchEvent.Kind k = event.kind();
                    if (event.kind() == StandardWatchEventKinds.OVERFLOW) { //it's possible==as static final
                        System.out.printf("path:%s event:%s%n", p, k);
                        continue;
                    }
                    System.out.printf("path:%s event:%s%n", p, k);
                }
                if(!key.reset()) {
                    System.out.printf("Key is invalid:%n");
                    break;
                }
            }

        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(ws);
        }
    }
}
```

-
-
- f

Java NIO1 Update

Java NIO1 Update

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Дополнения по Selector, Channels и Socket MultiThread

Java NIO1 Components

- Selectors(NIO)
- Selectors
- Работа с Selector
- Selector for Pipes
- Server Socket Selector Example Procedure
- Server Socket Examples
- Server Socket Message Example
- Server Socket Threads Example Procedure
- ServerSocket Threads Message Example
- Server Socket Threads Worker Example
- Char Server
- Chat Server Socket Threads
    - Server Socket Chat Server Threads Procedure
    - Server Socket Chat Server Threads Demo Example
- Chat Server Channels Selector
    - Server Socket Chat Server Example Procedure
    - Server Socket Chat Server Demo Example
- Дополнительные данные
    - Concurrent ArrayList
    - CMD characters Encoding
    - Использование накопителей и селектора на запись
- UDP Socket Threads
    - DatagramSocket Echo Server Threads Examle
    - UDP EchoServe на базе Sockets
    - UDP EchoServe на базе Channels
- UDP Channel Selector
    - DatagramChannel Echo Server Selector Examle
    - UDP EchoServer на базе Channels и Selector
-
-

Selectors

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Selectors
    - SelectableChannel класс канала, который может быть multiplexed via a Selector
    - blocking mode блокирующий режим работы канала
    - non-blocking mode неблокирующий режим работы канала
- Channels которые поддерживают этот режим
- Pipe
    - Pipe.SinkChannel класс
    - Pipe.SourceChannel класс
- Network
    - ServerSocketChannel ласс к
    - SocketChannel ласс к
    - DatagramChannel с к
- SCTP Channels клиент и сервер на Java
    - специальный канал, наподобие UDP, для передачи сообщений
    - свойства safe for multithreading, support concurrent 1 read and 1 write at time most
    - многоканальные endpoint, каждая поддерживает несколько направлений
    - example официальный пример и дополнительный пример кода
    - SctpChannel класс
    - SctpMultiChannel класс
    - SctpServerChannel класс
- Пример. реализация SCTP Channel **java8_nio/nio1/sctp**
- ВНИМАНИЕ. SCTP каналы НЕ ПОДДЕРЖИВАЕТСЯ Windows.
-

Работа с Selector

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Работа с Selector
    - открытие создаются SelectableChannels каналы передачи данных
    - регистрация каналы регистрируются в Seelctor, можно подключить свой класс в attachment
    - каждая регистрация возвращает SelectionKey, имеет наборы interest set и ready set
    - interest set операции будут проверяться в вызове select() регистрация
    - ВНИМАНИЕ. Операции interest set используются ПРИ РЕГИСТРАЦИИ канала
    - ready set операции, которые канал готов выполнить получение
    - ВНИМАНИЕ. Операции ready set используются ПРИ АНАЛИЗЕ SelectionKey
    - получение select() блокирующий метод, ожидает готовности каналов
    - select(time) неблокирующий метод возвращает текущий результат после time
    - проверка readyOps() выдает bitwise набор готовых к работе операций
    - удаление removeKey() удаляет отработанный ключ ОБЯЗАТЕЛЬНО

Selector for Pipes

- Pipes
    - уровень работают внутри application на уровне Threads
    - завершение если передающая сторона не закрыта, то канал вернет 0 при отсутствии данных
    - если передающая сторона закрыта, то канал вернет -1 при попытке чтения
- Пример. реализация завершения **java8_nio/nio1/selectors/pipes**
- PipeSource
    - PipeSource >> sender обычная передающая сторона
    - PipeSource >> senderLimited передающая сторона завершает передачу без закрытия канала
    - канал закрывается снаружи по timeout
    - PipeSource >> senderLimitedEof передающая сторона завершает передачу с закрытием канала
    - канал закрывается по len=-1 и interest.set(0) затирает ключ
- Завершение
    - Pipes допускает принудительное закрытие канала снаружи
    - Selector позволяет определить завершение чтения или записи Pipe
    - при чтении len=-1 означает что канал передачи закрыт, надо закрывать ключ и pipe
    - при записи выдает IOException означает канал приема закрыт, надо закрывать ключ и pipe
- Процедура
    - остановить Thread в котором работал pipe, Thread.shutdown() и Thread.awaitTermination()
    - закрыть оба канала pipe Pipe.Sink, Pipe.Source
    - закрыть SelectionKey при помощи key.cancel
- ВНИМАНИЕ. Если канал закрыт на передачу, то необходимо удалить ключ key.cancel() селектора
    - Если канал не закрыт на передачу, то надо принудительно закрыть канал по timeout
    - Если канал на прием закрыт, выбросит IOException и надо закрыть канал по timeout
    - Если канал на прием не закрыт, , то надо принудительно закрыть канал по timeout

Пример Pipe and Selectors

- SelectorPipes два блока по 6 каналов Pipes, в каждом три неблокирующих, три блокирующих
    - три канала с разным автоматическим движком, unlimited, limited и limitedEof
- UserPipeSource класс автоматического генератора данных и канала приема для Selector isReadable()
- UserPipeSink класс автоматического приемника данных и канала передачи для Selector isWritable()

Selector for Sockets

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Sockets
    - уровень работают между applications на уровне JVM
    - обмен server слушает sockets
    - client принимает и передает данные от сервера
- ВНИМАНИЕ. НЕТ СПОСОБА сделать нормальное честное завершение Thread где запущена System.in.read()
- Есть вариант, использовать System.in.available() при этом введенные символы не видны
- Есть вариант, использовать Thread.setDaemopn(true) при этом реально Thread не завершается

Server Socket Selector Example Procedure

- Server Socket Selector Example работа с SocketChannel на базе Selector
    - составляющие Selector, ServerSocketChannel, SocketChannel, Selector
- Selector
    - создается один Selector который отрабатывает как ServerSocketChannel и SocketChannel запросы
    - ServerSocketChannel.accept() запрос на подключение очередного клиента
    - SocketChannel.read(), write() запросы на передачу данных из и в канал клиента
    - select() метод, блокирующий без аргументов, получает число готовых каналов
    - selectedKeys() метод получает Set<SelectionKey> позволяет отработать все готовые каналы
    - специализация канала определяется набором операций при регистрации register()
    - после обработки КЛЮЧ надо УДАЛИТЬ
    - при закрытии канала ВСЕ КЛЮЧ связанный с ним УДАЛЯЕТСЯ
- ServerSocketChannel
    - создается один канал сервера, регистрируется Selector (OP_ACCEPT) и принимает много клиентов
    - accept() получает канал клиента,который регистрируется в Selector(OP_READ|OP_WRITE)
- SocketChannel
    - создается один для данного примера ТОЛЬКО ОДИН канал клиента для обмена данными
    - обмен данными разнесен в isReadable(), isWritable() между ними данные переносит String message

Server Socket Examples

- Server Socket Examples примеры server socket code,
- Пример. реализация Selector with Input Stream **java8_nio/nio1/selectors/sockets/console**
    - один сервер адреса 9990 и 9991, два ключа ServerScoketChannel
    - после accept подменяет ключи server(OP_ACCEPT) на client(OP_READ|OP_WRITE)
    - два клиента InputStream in работает с консолью, два ключа SocketChannel
    - aa выдать время с сервера
    - cc закрыть клиент и уведомить сервер для удаления ключа
- Пример. реализация Selector with Messages **java8_nio/nio1/selectors/sockets/message**
    - один сервер адреса 9990 и 9991, два ключа ServerScoketChannel
    - после accept подменяет ключи server(OP_ACCEPT) на client(OP_READ|OP_WRITE)
    - два клиента String[] MESSAGES как источник данных, "aa" выдать время,"cc"закрыть клиент
- Пример. реализация Selector with ServerSocketFactory **java8_nio/nio1/selectors/sockets/factory**
    - два порта адреса 9990..9991 создаются фабрикой ServerSocketFactory
    - регистрация каналов, accept() проводится методами Factory
    - четыре клиента String[] MESSAGE на каждый порт, "aa" выдать время с сервера,"cc"закрыть клиент
- ВНИМАНИЕ. Реализация ПО ОДНОМУ КЛИЕНТУ НА ПОРТ сделана в BRANCH multiple_port

Server Socket Message Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Server Socket Examples примеры server socket code
    - создает один сервер и один клиент и передает запросы и ответы между ними автоматом
- Пример. реализация ServerSocketChannel **java8_nio/nio1/selectors/sockets/message**
- ```
  public class UserServerSocket {
      private static final int DEFAULT_PORT = 9990;
      private static final long SESSION_LENGTH = 50000;

      public static void main(String[] args) {
          int port = DEFAULT_PORT;
          if (args != null && args.length > 0) {
              try {
                  port = Integer.parseInt(args[0]);
              } catch (NumberFormatException e) {
                  System.out.printf("Use default port:%d", port);
              }
          }

          try {
              String cp = "out/production/java_nio";
              String name = "nio1.selectors.sockets_CHECK_THIS.message.UserClientSocket";
              Runtime.getRuntime().exec("cmd /c start java -cp " + cp + " " + name);
              Runtime.getRuntime().exec("cmd /c start java -cp " + cp + " " + name);
              Runtime.getRuntime().exec("cmd /c start java -cp " + cp + " " + name);
          } catch (IOException e) {
              e.printStackTrace();
          }
  // Server
          ServerSocketChannel ssc = null;
          ServerSocket ss = null;
          ByteBuffer b = ByteBuffer.allocate(2000);                    // message
          LocalDateTime sessionTime = LocalDateTime.now().plus(SESSION_LENGTH, ChronoUnit.MILLIS);
          String message = "";
          try {
              Selector selector = Selector.open();
              ssc = ServerSocketChannel.open();                        // channel
              ss = ssc.socket();                                       // socket
              ss.bind(new InetSocketAddress("localhost", port));
              ssc.configureBlocking(false);
              ssc.register(selector, SelectionKey.OP_ACCEPT, "SSC");     // пока единственная операция

              System.out.printf("Server  started  local:%s%n", ssc.getLocalAddress());
              while (!LocalDateTime.now().isAfter(sessionTime)) {
                  int n = selector.select(2500);
                  if (n == 0) {
                      Set<SelectionKey> set = selector.keys();
                      if (set.size() > 0) {
                          SelectableChannel sc = set.iterator().next().channel();
                          if (sc instanceof ServerSocketChannel) {
                              System.out.print(".");
                          }
                      }
                      continue;
                  }

                  Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                  while (it.hasNext()) {
                      SelectionKey key = it.next();
                      if (key.isAcceptable()) {                        // принять соединение
                          SocketChannel sc = ((ServerSocketChannel) key.channel()).accept();
                          if (sc == null) continue;
                          System.out.printf("%naccepted local:%s remote:%s%n",
                                  sc.getLocalAddress(), sc.getRemoteAddress());
  // ВНИМАНИЕ СДЕЛАТЬ НЕСКОЛЬКО КЛИЕНТОВ НА ОДИН И ТОТ ЖЕ АДРЕС
  // OP_ACCEPT КЛЮЧ НЕ УДАЛЯТЬ И НЕ РЕГИСТРИРОВАТЬ

                          sc.configureBlocking(false);
                          String attachment = String.format("SC%d", sc.socket().getPort());
                          sc.register(selector, SelectionKey.OP_READ | SelectionKey.OP_WRITE, attachment);
                          String s = String.format("accepted:%s port:%d %s %s%n", attachment,
                                    ssc.socket().getLocalPort(), sc.getLocalAddress(),sc.getRemoteAddress());
                          sendChannel(sc, s, b);
                      }
  ```
  ```

- Пример. реализация ServerSocketChannel продолжение java8_nio/nio1/selectors/sockets/message
-
```
                    else if (key.isReadable()) {
                        SocketChannel sc = (SocketChannel) key.channel();
                        String s = readChannel(sc, b);
                        if (s != null) {
                            System.out.printf("%s:%s", key.attachment(), s);
                            if (s.matches("cc\\s*")) {
                                key.cancel();
                                sc.close();
                                System.out.printf("client closed%n");
                            }
                            message = s.replaceAll("\\s*", "") + key.attachment();
                        }
                    } else if (key.isWritable()) {
                        if (message.matches("aa" + key.attachment())) {
                            SocketChannel sc = (SocketChannel) key.channel();
                            String s = String.format("answer %s: at:%2$tT %2$tD%n",
                                                     key.attachment(), LocalDateTime.now());
                            sendChannel(sc, s, b);
                            message = "";
                        }
                    }
                    it.remove();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeChannel(ssc, ss);
        }
    }

    private static void sendChannel(SocketChannel sc, String s, ByteBuffer b) throws IOException {
        b.clear();
        b.put(s.getBytes(Charset.defaultCharset()));
        b.flip();
        while (b.hasRemaining()) {
            sc.write(b);
        }
    }

    private static String readChannel(SocketChannel sc, ByteBuffer b) throws IOException {
        b.clear();
        if (sc.read(b) == 0) return null;
        b.flip();
        return new String(b.array(), 0, b.limit(), Charset.defaultCharset());
    }
}
```

- Пример. реализация ClientSocket продолжение **java8_nio/nio1/selectors/sockets/message**

```
public class UserClientSocket {
    private static final int DEFAULT_PORT = 9990;
    private static final long SESSION_LENGTH = 50000;
    private static final String[] MESSAGES = {
            "Message from client",
            "Required new books in library",
            "aa",
            "Please go to the room entrance",
            "Wait for minute in the shop",
            "aa",
            "Second moment client say \"close connection\"!",
            "cc"
    };

    public static void main(String[] args) throws InterruptedException {
        int port = DEFAULT_PORT;
        if (args != null && args.length > 0) {
            try {
                port = Integer.parseInt(args[0]);
            } catch (NumberFormatException e) {
                System.out.printf("Use default port:%d", port);
            }
        }

// Client
        SocketChannel sc = null;
        ByteBuffer b = ByteBuffer.allocate(50);
        try {
            sc = SocketChannel.open();
            InetSocketAddress address = new InetSocketAddress("localhost", port);
            sc.connect(address);
            sc.configureBlocking(false);

            b.clear();
            for (int i = 0; i < MESSAGES.length; i++) {
                b.clear();
                String s = String.format("%s%n", MESSAGES[i]);
                byte[] bytes = s.getBytes(Charset.defaultCharset());
                b.put(bytes);
                b.flip();
                while (b.hasRemaining()) {
                    sc.write(b);
                }
                b.rewind();
                s = new String(b.array(), 0, b.limit(), Charset.defaultCharset());
                System.out.printf("%s", s);
                if (s.contains("cc")) break;

// waiting response 1 sec
                b.clear();
                int count = 5;
                while (count > 0) {
                    if (sc.read(b) > 0) {
                        b.flip();
                        s = new String(b.array(), 0, b.limit(), Charset.defaultCharset());
                        System.out.printf("%s", s);
                    }
                    count--;
                    Thread.sleep(100);
                }
            }

            Thread.sleep(100);
            while (!sc.finishConnect()) {
                System.out.print(".");
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        } finally {
            IOUtils.closeChannel(sc);
        }

        System.out.printf("client closed...%n");
        Thread.sleep(10000);
    }
}
```

Server Socket Threads Example Procedure

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Server Socket Threads Example chat server на базе Sockets работающих в разных потоках
  - main() поток работает ОТДЕЛЬНО чтобы прервать ServerSocket.accept() из другого потока
  - ServerSocket.accept() можно прервать только по SocketException или SocketTimeout
  - SocketException срабатывает если закрыть ServerSocket из другого потока
  - SocketTimeout срабатывает если превышено SocketServer.setSoTimeout()
- AcceptService выделяется один поток, который принимает соединения ServerAcceptService()
  - создает Socket sc связи с клиентами
  - запускает потоки ClientService(Socket sc)
  - прерывание возможно полное если закрыть ServerSocket что выбросит SocketException
  - SocketException прервет блокирующий метод ServerSocket.accept()
- ClientService на каждый Socket клиента выделяется поток на сервере ServerClientService()
  - обмен BufferedReader br = Socket sc.getInputStream() Charset.forName ("CP866")
  - BufferedWriter br = Socket.getOutputStream() Charset.forName ("CP866")
  - charset ОБЯЗАТЕЛЬНО такой, как сказано, связано с установками Windows для console
  - чтение блокирующее BufferedReader.readLine(), позволяет отследить закрытие клиента
  - блокирущее чтение ПРЕДПОЧТИТЕЛЬНЕЕ при работе с Telnet или Putty
  - неблокирующее через BufferedReader.ready() не обнаруживает закрытие клиента
  - неблокирующее чтение удобно если Client поддерживает текстовый выход
- ВНИМАНИЕ. Проверка удаленного Socket на закрытие
  - isCLosed() если socket был закрыт корректно, то выдаст реальное состояние ИСПОЛЬЗУЮ ЭТО
  - read()=-1 если socket был закрыт как то по старому, потому что в новом выдает Exception
- Encoding кодировка для работы с клиентами
  - клиенты Telnet CP866 внешний интерфейс с ClientService
  - Putty CP866 внешний интерфейс с ClientService
  - Java cmd CP866 System.in и внешний CP866 для совместимости
  - Java IDEA UTF-8 System.in и внешний CP866 для совместимости
- ClientServiceApp запускается вообще в отдельном приложении app JVM как удаленные устройства
  - telnet может работать как клиент
  - putty –raw может работать как клиент
  - чтение System.in Charset.forName ("CP866") если клиент запущен из под cmd
  - System.in Charset.forName ("UTF-8") если клиент запущен из под IDEA

ServerSocket Threads Message Example

- ServerSocket Message Example
  - работа сервер запускает несколько клиентов
  - каждый клиент отправляет несколько сообщений и запросов "aa","cc" автоматом
  - составляющие ServerAcceptService, ServerClientService
- ServerAcceptService в цикле принимает Socket от клиентов и создает ServerClientService потоки
  - выходит либо по закрытию ssc.close() либо по ssc.setSoTimeout(4500)
- ServerClientService работает с клиентом через Socket sc,
  - выход по команде "cc", либо Thread.interrupt()
- Пример. реализация **java8_nio/nio1/selectors/threads/messsage**

Server Socket Threads Worker Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Server Socket Example сделан на базе примера server socket code
    - составляющие ServerAcceptService, ServerClietnService
- MultiThreadedServer main() поток
    - создает ServerAcceptService поток
    - ожидает 20 sec
    - закрывает ServerAcceptService.ssc.close() ServerSocket
    - в результате по Exception ServerSocket.accept() вылетает из while() и из потока
- ServerAcceptService
    - в цикле метод accept() получает Socket от клиентов
    - создает ServerClientService поток на каждый полученный Socket клиента
    - выходит по закрытию ssc.close() от main()
    - по timeout ssc.setSoTimeout(60000)
- ServerClientService
    - работает с клиентом через Socket sc, создается по запросу от клиента
    - выход сразу же после отправки сообщения на сервер
- Пример. реализация **java8_nio/nio1/selectors/threads/worker**

Server Socket Chat Server Threads Procedure

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Procedure составляющие main(), ServerAcceptService(), ServerClientService(),ClientServiceApp
- Server.main() запускает сервер в работу
    - создает SertviceAcceptService() Runnable и запускает его
    - в цикле while() отслеживает время и количество работающих клиентов
    - прерывает цикл если число клиентов равно нулю или время вышло
    - завершение закрывает ServerSocket.close() в службе ServerAcceptService()
    - ServerSocket.close() прерывает блокирующий метод ServereSocket.accept()
    - число клиентов можно отследить ТОЛЬКО если есть список List<ServerClientService>
- ВНИМАНИЕ. List<> на базе CopyOnWriteArrayList() Thread Safe версия ArrayList чтение/запись 90/10
    - List<> на базе Collections.synchronizedList() медленный НЕ ИСПОЛЬЗУЕТСЯ чтение/запись 50/50
- ServiceAcceptService
    - в цикле принимает клиентов методом ServerSocket.accept()
    - получает канал Socket клиента, создает ServerClientServce() Runnable и запускает ее
    - добавляет службу в список List<ServerClientService> для последующей рассылки
    - прерывание ServerAcceptService возможно только закрытием ServerSocket.close()
    - при этом прерывается блокирующий метод ServerScoket.accept()
- ServerClientService
    - в цикле принимает сообщения от клиента блокирующим методом br.readLine()
    - передает сообщения к клиенту блокирующим методом bw.write()
    - сообщения Chat передаются всем клиентам через bw.write() списка клиентов
    - прерывание ServerClientService() возможно если блокирующий метод br.readLine()
    - возвращает null что означает Client закрыт
    - возвращает кодове слово "disconnect" от клиента, по которому служба закрывается
    - при закрытии закрывает канал с клиентом Socket.close()
    - удаляет себя ServerClientService.this из списка List<ServerClientService>
- Server.main()
    - ожидает первые 5 секунд чтобызапустились клиенты и крутится в цикле пока список не пуст
    - что означает все службы ServerClientService закрыты
    - далее ServerSocket.close() прерывает и закрывает службу ServerAcceptService()
    - и наконец Server завершает работу, так как все службы закрыты
    - ExecutorService используется для запуска всех служб ExecutorService.execute()
    - для остановки служб ExecutorService.shutdown(),ExecutorService.shutdownNow()
    - для контроля ExecutorService. awaitTermination (), ExecutorService.isTerminate()
- ClientServiceApp Telnet, Putty, ClientServiceApp клиенты
- Пример. реализация **java8_nio/nio1/selectors/threads/chat/chatm/ClientService**

Concurrent ArrayList

- Collections.synchronizedList() vs CopyOnWriteArrayList
    - CopyOnWriteArrayList is a good choice if number of reads is significantly higher than writes.
    - Collections.synchronizedList() is a good choice when writes outnumber reads
    - CopyOnWriteArrayList can't be used if you want your iterators to detect concurrent modification or
    - if you want to update the collection through the iterator
    - Collections.synchronizedList() used for such cases

Server Socket Chat Server Threads Demo Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Server Socket Chat Server пример code example
    - concurrent Arraylist две модификации и когда использовать
    - Collections.synchronizedList() только один поток может работать все операции
    - CopyOnWriteArrayList() работает много потоков, эффективно только чтение
- Пример. реализация Server.main() **java8_nio/nio1/selectors/threads/chat/demo**

```
public class ChatServer {
    private static final int PORT = 9990;
    private static final String HOST = "localhost";
    private static final long SESSION = 50000;
    private static List<ClientService> list = new CopyOnWriteArrayList<>();
    public static void main(String[] args) {
        LocalDateTime finishTime = LocalDateTime.now().plus(SESSION, ChronoUnit.MILLIS);
        ExecutorService exec = Executors.newCachedThreadPool();
        try {
            AcceptService as = new AcceptService();
            exec.execute(as);
// clients
            Runtime.getRuntime().exec("cmd /c start telnet " + HOST + " " + PORT);
            Runtime.getRuntime().exec("cmd /c start telnet " + HOST + " " + PORT);
            nio1.selectors.threads.chat.chatm.ClientService.run(HOST,PORT);
            nio1.selectors.threads.chat.chatm.ClientService.runPutty(HOST,PORT);
            Thread.sleep(5000);

            while (!LocalDateTime.now().isAfter(finishTime) && !list.isEmpty()) {
                Thread.sleep(1000);
            }
            as.ssc.close();
            exec.shutdown();
            exec.awaitTermination(100, TimeUnit.MILLISECONDS);
        } catch (IOException | InterruptedException e) {
            System.out.printf("%s%n", e);
        }
        if (!exec.isTerminated())  System.out.printf("can't close server%n");
        else   System.out.printf("server closed%n");
    }
```

- Пример. реализация AcceptService() **java8_nio/nio1/selectors/threads/chat/demo**

```
    private static class AcceptService implements Runnable {
        private ServerSocket ssc;
        public AcceptService() throws IOException {
            this.ssc = new ServerSocket();
            this.ssc.bind(new InetSocketAddress(HOST, PORT));
        }
        @Override
        public void run() {
            ExecutorService exec = Executors.newCachedThreadPool();
            try {
                while (true) {
                    Socket sc = ssc.accept();  // блокирующий метод выход по ssc.close SocketException
                    if (sc == null) continue;
                    ClientService cs = new ClientService(sc);
                    list.add(cs);
                    exec.execute(cs);
                }
            } catch (IOException e) {
                System.out.printf("%s%n", e);
            }
            try {
                exec.shutdown();
                exec.awaitTermination(100, TimeUnit.MILLISECONDS);
            } catch (InterruptedException e) {
                e.printStackTrace();
            } finally {
                IOUtils.close(ssc);
            }
            if (!exec.isTerminated()) System.out.printf("can't close accept service...%n");
            else  System.out.printf("accept service terminated...%n");
        }
    }
```

- Пример. реализация ClientService() java8_nio/nio1/selectors/threads/chat/demo
-
```
private static class ClientService implements Runnable {
        private static final Charset TELNET_CHARSET = Charset.forName("CP866");
        private final Socket sc;
        private final String name;
        private boolean isClosed;
        private BufferedReader br;
        private BufferedWriter bw;


        public ClientService(Socket sc) throws IOException {
            if (sc == null) throw new IllegalArgumentException();
            this.sc = sc;
            this.br = new BufferedReader(new InputStreamReader(sc.getInputStream(), TELNET_CHARSET));
            this.bw = new BufferedWriter(new OutputStreamWriter(sc.getOutputStream(), TELNET_CHARSET));

            this.name = String.format("SC%d", sc.getPort());
            this.isClosed = false;
        }

        @Override
        public void run() {
            try {
                String s = String.format("Welcome to Socket IO Chat!%n");
                bw.write(s);
                bw.flush();

                while (!isClosed) {
                    s = br.readLine();
                    if (s == null) break;             // termination upon client closing
                    s = String.format("%s:%s%n", name, s);
                    broadcast(s);
                }
                isClosed = true;
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.close(sc, br, bw);
            }
            list.remove(this);
            System.out.printf("%s:terminated and removed...%n",name);

        }

        private void broadcast(String s) throws IOException {
            for (ClientService cs : list) {
                if (cs == null || cs.isClosed) continue;
                cs.bw.write(s);
                cs.bw.flush();
            }
        }

    }
}
```
-

Server Socket Chat Server Example Procedure

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Procedure составляющие Server.main(), ClientServiceApp
- Server.main() запускает сервер в работу
    - создает Selector
- ServerSocketChannel
    - открывает ServerSocketChannel.open()
    - регистрирует ServerSocketChannel.register(Selector, OP_ACCEPT) в селекторе
    - в цикле while() отслеживает время и количество Selector.keys()
    - прерывает цикл если Selector.keys()= 1, все клиенты вышли или время вышло
    - завершение закрывает ServerSocketChannel и Selector
- while()
    - Selector.select() получает активные SelectionKey для каналов которые готовы работать
    - key.isAcceptable() ServerSocketChannel.accept() подключает нового клиента SocketChannel
    - регистрирует SocketChannel.register(Selector, OP_READ) канал в селекторе
    - key.isReadable() SocketChannel получает данные от клиента и отсылает их клиентам broadcast()
    - если чтение с канала возвращает len = -1 значит канал закрыт удаленно
    - поэтому SocketChannel закрывается и здесь, автомато отключаются SelectionKey
    - broadcast() метод получает все SelectionKeys в селекторе методом Selector.keys()
    - отрабатывает только те которые имеют OP_READ то есть каналы клиентов
    - отправляет им сообщения
- Server.main()
    - запускается с флагом старта, что пока клиентов нет и это нормально count = 0
    - при подключении хотя бы одного клиента count++
    - обнаруживает что Selector.keys() == 1 и count > 0 что значит клиенты были и отключились
    - и выходит из цикла, закрывает ServerSocketChannel, Selector и завершает работу
- ClientServiceApp
    - Telnet работает ПО СИМВОЛЬНО поэтому передавать Channel.ID нельзя напрямую
    - chatm использует StringBuilder на каждый канал, накапливает строку до \r\n
    - Putty работает построчно, можно использовать Channel.ID напрямую
    - ClienService работает построчно, можно использовать Channel.ID напрямую

Использование накопителей и селектора на запись

- Проект chatm использует следующие функции
    - накопитель Map<String,StringBuilder> накапливает символы в строку до появления \r\n
    - по символу \r\n отправляет строку broadcast
    - по символу \r\n обеспечивает распознавание "disconnect"
    - селектор записи Queue<UserTask> накапливает запросы на запись клиентам
    - класс UserTask использует два поля SocketChannel и String
    - при отработке SelectionKey.isWritable() канал прогоняется по Queue.iterator()
    - если канал найден, SocketChannel.write(String) пишет message в канал
    - элемент UserTask удаляется из очереди методом Iterator<UserTask>.remove()
- Пример. реализация **java8_nio/nio1/selectors/sockets/chat/chatm**
- ВНИМАНИЕ. Использование селектора записи с очередью позволяет КОРРЕКТНО писать в каналы
    - базовое решение НЕ ИСПОЛЬЗУЕТ селектор на OP_WRITE и пишет в каналы блокирующим способом

Server Socket Chat Server Demo Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Server Socket Chat Server Example
- Пример. реализация Server.main() **java8_nio/nio1/selectors/sockets/chat/demo**
-

```
public class ChatServer {
    private static final String HOST = "localhost";
    private static final int PORT = 9990;
    private static final long SESSION = 50000;
    private static final Charset TELNET_CHARSET = Charset.forName("CP866");

    private static final ByteBuffer b = ByteBuffer.allocate(2048);
    private static int count = 0;

    public static void main(String[] args) {
        ServerSocketChannel ssc = null;
        SocketChannel sc = null;
        Selector selector = null;
        try {
            ssc = ServerSocketChannel.open();
            ssc.bind(new InetSocketAddress(HOST, PORT));
            LocalDateTime finishTime = LocalDateTime.now().plus(SESSION, ChronoUnit.MILLIS);
            ssc.configureBlocking(false);
            selector = Selector.open();
            ssc.register(selector, SelectionKey.OP_ACCEPT);
            System.out.printf("server started...%n");
// clients
            ClientService.runTelnet(HOST, PORT);
            ClientService.runTelnet(HOST, PORT);
            ClientService.run(HOST, PORT);
            ClientService.runPutty(HOST, PORT);

            while (!LocalDateTime.now().isAfter(finishTime)) {
                int n = selector.select(100);
                if (n == 0) {
                    if (selector.keys().size() > 1 || count == 0) continue;
                    else break;
                }
                Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                while (it.hasNext()) {
                    SelectionKey key = it.next();
                    it.remove();
                    if (key.isAcceptable()) {
                        sc = ((ServerSocketChannel) key.channel()).accept();
                        if (sc == null) continue;
                        sc.configureBlocking(false);
                        sc.register(selector, SelectionKey.OP_READ, String.format("SC%d",
sc.socket().getPort()));
                        sendMessage(sc, String.format("Welcome to NIO Chat%n"));
                        System.out.printf("%s:entered to chatwide%n", sc.getRemoteAddress());
                        count++;

                    } else if (key.isReadable()) {  // data arrived from channel
                        sc = (SocketChannel) key.channel();
                        if (sc == null) continue;
// read
                        String s = readMessage(sc);
                        if (s == null) {
                            System.out.printf("%s:left chatwide%n", sc.getRemoteAddress());
                            sc.close();         // closes all keys connected with channel
                        } else if (s.length() > 0) {
                            broadcast(selector, key, s);
                        }
                    }
                }
            }

        } catch (IOException e) {
            System.out.printf("Exception:%s%n", e);
        } finally {
            IOUtils.close(ssc, selector);
        }
        System.out.printf("server closed...%n");
    }
```

- Пример. реализация Server.main() продолжение **java8_nio/nio1/selectors/sockets/chat/demo**

```
    private static String readMessage(SocketChannel sc) throws IOException {
        try {
            b.clear();
            int len;
            if ((len = sc.read(b)) > 0) {
                b.flip();
                return new String(b.array(), 0, b.limit(), TELNET_CHARSET);

            } else if (len == -1) {
                return null;
            }
        } catch (IOException e) {
            return null;
        }
        return "";
    }

    private static void sendMessage(SocketChannel sc, String s) throws IOException {
        b.clear();
//        String message = String.format("SC%d:%s", sc.socket().getPort(), s);
//        b.put(message.getBytes(TELNET_CHARSET));
        b.put(s.getBytes(TELNET_CHARSET));
        b.flip();
        while (b.hasRemaining()) {
            sc.write(b);
        }
    }

    private static void broadcast(Selector selector, SelectionKey src, String s) throws IOException {
        Set<SelectionKey> set = selector.keys();

        for (SelectionKey key : set) {
            if (key == src) continue;
            if (key.isValid() && (key.interestOps() & SelectionKey.OP_READ) > 0) {
                SocketChannel sc = (SocketChannel) key.channel();
                sendMessage(sc, s);
            }
        }
    }

    private static class UserTask {
        private final SocketChannel sc;
        private final String message;

        public UserTask(SocketChannel sc, String message) {
            this.sc = sc;
            this.message = message;
        }
    }
}
```

-
-
-
- f

CMD characters Encoding

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- CMD characters Encoding кодировка cmd для System.in установлена cp866
    - ввод с консоли cmd по System.in возможен только с Charset.forName("CP866") статья по console
- CMD вывод символов на консоль еще статья о кодировке вообще
    - вывод символов не пригодился, так как при вводе я вижу вводимые символы
    - консоль Windows cmd НЕ ПРИСПОСОБЛЕНА для вывода UTF-8 в принципе
    - работает ПЛОХО и ТОЛЬКО ЧЕРЕЗ ЯРЛЫК так как нужна установка шрифта Lucinda Console
- Файлы
    - EncodingOutputApp запуск окна консоли cmd
    - EncodingOutput собственно вывод разных шрифтов
    -
- ВНИМАНИЕ. Статья вывод символов независимо от cjodepage cmd ТУПАЯ СТАТЬЯ все нормально выводится
- Пример. реализация вывода через ярлык **java8_nio/nio1/selectors/threads/encoding**
    - вызов консоли прозводится через ярлык c.cmd.lnk и в EncodingApp.main()
    - смена страницы кодировки chcp 65001 то есть UTF-8 задается в командной строке c.cmd
    - вывод сделан напрямую и из файлов через BufferedInputStream и BufferedReader
    - BufferedReader корректно работает ТОЛЬКО через массивы char[]
    - BufferedinputStream корректно работает ТОЛЬКО через массивы byte[]
- ВНИМАНИЕ. НЕ ИСПОЛЬЗОВАТЬ BufferedReader.readLine() если кодировка сложная, только через массив char[]
- CMD запуск нескольких команд в одной строке не пригодилось
    - смена codepage не пригодилась, но сам механизм работает
    - cmd несколько команд в одной строке

```
start "MyWindow" cmd /c "ping localhost & ipconfig & pause"
("cmd /c start call cmd /c \"chcp 1251 && java -cp " + cp + " " + name+"\"");
```

- Файлы
    - EncodingApp запуск окна консоли cmd
    - Encoding собственно ввод с консоли и демонстрация ГРУППЫ команд в вызове cmd
- Пример. реализация **java8_nio/nio1/selectors/threads/encoding**
-

```
public static void main(String[] args) throws IOException {
        int n;
        byte[] bytes = new byte[1024];
        System.out.printf("Введите строку (консоль по умолчанию Charset.forName(\"CP866\"): %n");
        InputStream in = System.in;
        int len = in.read(bytes);
        bytes = Arrays.copyOfRange(bytes, 0, len);
        System.out.print("bytes:[");
        for (byte b : bytes) {
            System.out.printf("%03d(0x%02X) ", (int) b & 0xFF, b);
        }
        System.out.println("]");
        String s = new String(bytes, 0, len, Charset.forName("CP866"));
        System.out.printf("%s%n", s);
        s = new String(bytes, 0, len, Charset.forName("WINDOWS-1251"));
        System.out.printf("%s%n", s);
        s = new String(bytes, 0, len, Charset.forName("UTF-8"));
        System.out.printf("%s%n", s);
        System.out.printf("Введите строку Scanner(ADAPTIVE CHARSET)): %n");
        String encoding = System.getProperty("console.encoding","cp866");
        if(System.getProperty("file.encoding").equals("UTF-8")) {
            encoding = System.getProperty("console.encoding","utf8");
        }
        System.out.printf("console:%s file:%s%n",encoding,System.getProperty("file.encoding"));
//        ps = new PrintStream(System.out, false, encoding);
        Scanner sc = new Scanner(System.in, encoding);
        System.out.printf("%s%n", sc.nextLine());
    }
```

DatagramSocket Echo Server Threads Examle

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- DatagramSocket Echo Server Threads Exampe данные от Telnet на UDP Server и обратно на базе Threads
    - составляющие ServerSocket регистрирует Socket от клиент Telnet/Putty
    - Socket обеспечивает передачу данных от и к клиенту Telnet/Putty
    - DatagramSocket сервер ожидает UDP пакеты и отправляет Echo обратно
    - DatagramSocket клиент получает строку от Socket и отправляет UDP Server
- Важные моменты работы с DatagramSocket
    - bind() всегда производится автоматом, если сокет создается конструктором
    - создается с адресом SocketAddress new DatagamSocket(SocketAddress)
    - создается с адресом 0.0.0.0/0.0.0.0 new DatagamSocket()
    - SocketAddress является предком InetSocketAddress можно подставлять напрямую
    - connect() вызывается ТОЛЬКО на стороне клиента
    - receive(p) принимает данные в DatagramPacket(byte[], len, SocketAddress)
    - send(p) отправляет данные из DatagramPacket(byte[], len, SocketAddress)
    - вызов receive(p) затем send(p) на том же SocketAdress ТРЕБУЕТ REUSAGE DatagramPacket
    - ОБЯЗАТЕЛЬНО после приема в DatagramPacket p ИМЕННО его использовать в send()
    - в противном случае он будет зациклен принимая свои же данные
    - вызов send(p) затем recevie(p2) для того же SocketAdress ТРЕБУЕТ НОВЫЙ DatagramPacket
    - ОБЯЗАТЕЛЬНО после передачи p на прием использовать НОВЫЙ DatagramPacket p2
    - в противном случае он примет свои же данные
- ВНИМАНИЕ. Вызов receive() и send() на одном адресе ТРЕБУЕТ особого использования DatagramPacket
- Закрытие
    - receive(p) блокирующий метод, прервать его можно только если ЗАКРЫТЬ DatagramSocket
- Encoding
    - для telnet используется Charset.forName("CP866")
    - для putty используется Charset.forName("UTF-8")
- Пример. реализация приема и затем передачи на том же адресе **java8_nio/nio1/selectors/threads/echo**

```
while (true) {
    if (exec.isTerminated()) {
        break;
    }
    dc.receive(p);                          // блокирующий метод
    if (p.getLength() > 0) {
        String s = "UDP:" + new String(p.getData(), 0, p.getLength(), UDP_CHARSET);
        bytes = s.getBytes(UDP_CHARSET);

        p.setData(bytes);                   // В ДАННОМ СЛУЧАЕ REUSE ОБЯЗАТЕЛЕН
        dc.send(p);
    }
    Thread.sleep(100);
}
```

- Пример. реализация передачи и затем приема на том же адресе **java8_nio/nio1/selectors/threads/echo**

```
dc = new DatagramSocket();
dc.connect(udpAddress);
while (true) {
    String s;
    if ((s = br.readLine()) == null) break;  //блокирующий метод client closed
// udp
    p.setData(s.getBytes(UDP_CHARSET));
    dc.send(p);
    dc.receive(p2);                         // ЗДЕСЬ НАОБОРОТ REUSE ЗАПРЕЩЕН
    s = new String(p2.getData(), 0, p2.getLength(), UDP_CHARSET);
    pw.println(s);
}
```

UDP EchoServe на базе Sockets

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- UDP EchoServe на базе Sockets **java8_nio/nio1/selectors/threads/echo/UDPEchoSocket**
- Пример. реализация
- ```
  public class UDPEchoSocket {
      private static final Charset TELNET_CHARSET = Charset.forName("CP866");
      private static final Charset UDP_CHARSET = Charset.forName("UTF-8");
      private static final String HOST = "localhost";
      private static final int PORT = 9990;
      private static final int UDP_PORT = 9920;
      private static final String[] PUTTY_WELCOME = {
              "Welcome to echo server!",
              "To close Putty without warning message:",
              "Change Putty>>Window>>Behaviour Settings then Save Default Session",
              "Type any text<Enter> (closewindow to exit):"
      };
      private static int parsePort(String[] args, int port) throws NumberFormatException {
          if (args.length > 0) {
              try {
                  port = Integer.parseInt(args[0].replaceAll("\\p{Punct}", ""));
              } catch (NumberFormatException e) {
                  e.printStackTrace();
              }
          }
          return port;
      }
      private static boolean parseBlock(String[] args, boolean isBlocking) {
          if (args.length > 1) {
              return !args[1].replaceAll("\\p{Punct}", "").toLowerCase().contains("n");
          }
          return isBlocking;
      }
      public static void runTelnet(String host, int port) throws IOException {
          Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
      }
      private static void runPutty(String host, int port) throws IOException {
          Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
      }
      public static void main(String[] args) {
          System.out.printf("udp server started...%n");
          DatagramSocket dc = null;
          try {
              ExecutorService exec = Executors.newCachedThreadPool();
              runPutty(HOST, PORT);
              InputServer inputServer = new InputServer(UDP_CHARSET, PORT, UDP_PORT);  // for putty
              exec.execute(inputServer);  // for putty
  //            runTelnet(HOST, PORT);
  //            exec.execute(new InputServer(TELNET CHARSET, PORT, UDP PORT);  // for telnet
              exec.shutdown();
              SocketAddress socketAddress = new InetSocketAddress(HOST, UDP_PORT); // udp server
              dc = new DatagramSocket(socketAddress);  // bind
              DatagramPacket p = new DatagramPacket(new byte[1024], 1024);
              inputServer.setDatagramSocket(dc);
              byte[] bytes;
              while (true) {
                  if (exec.isTerminated()) {
                      break;
                  }
                  dc.receive(p);                              // блокирующий метод
                  if (p.getLength() > 0) {
                      String s = "UDP:" + new String(p.getData(), 0, p.getLength(), UDP_CHARSET);
                      bytes = s.getBytes(UDP_CHARSET);
                      p.setData(bytes);                       // В ДАННОМ СЛУЧАЕ REUSE ОБЯЗАТЕЛЕН
                      dc.send(p);
                  }
                  Thread.sleep(100);
              }
          } catch (IOException | InterruptedException e) {
              System.out.printf("Exception:%s%n",e);
          } finally {
              IOUtils.close(dc);
          }
          System.out.printf("udp server closed...%n");
      }
  ```
  ```

- Пример. реализация UDP Client с приемом данных от Socket
-
```
    private static class InputServer implements Runnable {
        private BufferedReader br;
        private PrintWriter pw;
        private Charset consoleCharset;
        private int port;
        private int udpPort;
        private DatagramSocket ssds;
```
-
```
        public InputServer(Charset consoleCharset, int port, int udpPort) {
            this.consoleCharset = consoleCharset;
            this.port = port;
            this.udpPort = udpPort;
        }

        synchronized public void setDatagramSocket(DatagramSocket ssds) {
            this.ssds = ssds;
        }

        synchronized public void closeDatagramSocket() {  // close server datagram socket
            if (ssds != null) {
                ssds.close();
            }
        }

        @Override
        public void run() {
// socket
            ServerSocket ssc = null;
            Socket sc = null;
            System.out.printf("socket server started...%n");
// udp
            DatagramSocket dc = null;
            InetSocketAddress udpAddress = new InetSocketAddress(HOST, udpPort);

            DatagramPacket p = new DatagramPacket(new byte[1024], 1024, udpAddress);
            DatagramPacket p2 = new DatagramPacket(new byte[1024], 1024, udpAddress);
            try {
                ssc = new ServerSocket();
                ssc.bind(new InetSocketAddress(HOST, port));

                sc = ssc.accept();
                br = new BufferedReader(new InputStreamReader(sc.getInputStream(), consoleCharset));
                pw = new PrintWriter(sc.getOutputStream(), true, consoleCharset); // flush by /r/n
                pw.println(Arrays.stream(PUTTY_WELCOME).collect(Collectors.joining(String.format("%n"))));

//udp
                dc = new DatagramSocket();
                dc.connect(udpAddress);
                while (true) {
                    String s;
                    if ((s = br.readLine()) == null) break;  //блокирующий метод client closed
// udp
                    p.setData(s.getBytes(UDP_CHARSET));
                    dc.send(p);
                    dc.receive(p2);                          // ЗДЕСЬ НАОБОРОТ REUSE ЗАПРЕЩЕН
                    s = new String(p2.getData(), 0, p2.getLength(), UDP_CHARSET);
                    pw.println(s);
                }

                closeDatagramSocket();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.close(ssc, sc, br, pw, dc);
            }
            System.out.printf("server socket closed...%n");
        }
    }
}
```
-

UDP EchoServe на базе Channels

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- UDP EchoServe на базе Channels java8_nio/nio1/selectors/threads/echo/UDPEchoChannel
    - свойства используется неблокирующий доступ к чтению канала
- Пример. реализация java8_nio/nio1/selectors/threads/echo/UDPEchoChannel
- ```
  public class UDPEchoChannel {
      private static final Charset TELNET_CHARSET = Charset.forName("CP866");
      private static final Charset UDP_CHARSET = Charset.forName("UTF-8");
      private static final String HOST = "localhost";
      private static final int PORT = 9990;
      private static final int UDP_PORT = 9910;
      private static final int UDP_PORT2 = 9910;

      private static final String[] PUTTY_WELCOME = {
              "Welcome to echo server!",
              "To close Putty without warning message:",
              "Change Putty>>Window>>Behaviour Settings then Save Default Session",
              "Type any text<Enter> (closewindow to exit):"
      };

      public static void runTelnet(String host, int port) throws IOException {
          Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
      }

      private static void runPutty(String host, int port) throws IOException {
          Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
      }

      public static void main(String[] args) {

          System.out.printf("udp server started...%n");
          DatagramChannel dc = null;
          try {
              ExecutorService exec = Executors.newCachedThreadPool();
              runPutty(HOST, PORT);
              exec.execute(new InputServer(UDP_CHARSET, PORT, UDP_PORT));  // for putty

  //            runTelnet(HOST, PORT);
  //            exec.execute(new InputServer(TELNET_CHARSET, PORT, UDP_PORT);  // for telnet
              exec.shutdown();
  // udp server
              InetSocketAddress inetSocketAddress = new InetSocketAddress(HOST, UDP_PORT);
              dc = DatagramChannel.open();
              dc.socket().bind(inetSocketAddress); // listening port
              dc.configureBlocking(false);

              ByteBuffer b = ByteBuffer.allocate(1024);

              while (true) {
                  if (exec.isTerminated()) {
                      break;
                  }
                  b.clear();
                  SocketAddress socketAddress = dc.receive(b);
                  if (socketAddress != null) {
                      b.flip();
                      String s = "UDP:" + new String(b.array(), 0, b.limit(), UDP_CHARSET);

                      b.clear();
                      b.put(s.getBytes(UDP_CHARSET));
                      b.flip();
                      dc.send(b, socketAddress); // не постоянное соединение, поэтому всегда отвечаем адресату
                  }
                  Thread.sleep(100);
              }

          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
          } finally {
              IOUtils.close(dc);
          }
          System.out.printf("udp server closed...%n");
      }
  ```
  ```

- Пример. реализация UDP Client **java8_nio/nio1/selectors/threads/echo/UDPEchoChannel**
-
```
    private static class InputServer implements Runnable {
        private BufferedReader br;
        private PrintWriter pw;
        private Charset consoleCharset;
        private int port;
        private int udpPort;

        public InputServer(Charset consoleCharset, int port, int udpPort) {
            this.consoleCharset = consoleCharset;
            this.port = port;
            this.udpPort = udpPort;
        }

        @Override
        public void run() {
// socket
            ServerSocket ssc = null;
            Socket sc = null;
            System.out.printf("socket server started...%n");
// udp
            DatagramChannel dc = null;
            ByteBuffer b = ByteBuffer.allocate(1024);
            InetSocketAddress udpAddress = new InetSocketAddress(HOST, udpPort);

            try {
                ssc = new ServerSocket();
                ssc.bind(new InetSocketAddress(HOST, port));

                sc = ssc.accept();
                br = new BufferedReader(new InputStreamReader(sc.getInputStream(), consoleCharset));
                pw = new PrintWriter(sc.getOutputStream(), true, consoleCharset); // flush by /r/n
                pw.println(Arrays.stream(PUTTY_WELCOME).collect(Collectors.joining(String.format("%n"))));

//udp
                dc = DatagramChannel.open();
                dc.connect(udpAddress);

                while (true) {
                    String s;
                    if ((s = br.readLine()) == null) break;  // client closed
// udp
                    b.clear();
                    b.put(s.getBytes(UDP_CHARSET));
                    b.flip();
                    dc.write(b);                        //  dc.send(b,udpAddress);


                    b.clear();
                    dc.receive(b);
                    b.flip();
                    s = new String(b.array(), 0, b.limit(), UDP_CHARSET);
                    pw.println(s);


                }


            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.close(ssc, sc, br, pw, dc);
            }
            System.out.printf("server socket closed...%n");
        }
    }
}
```
-

DatagramChannel Echo Server Selector Examle

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- DatagramChannel Echo Server Selector Exampe данные от Telnet на UDP Server и обратно на базе Selector
    - составляющие UDP Server, InputServer
    - UDP Server DatagramChannel сервер ожидает UDP пакеты и отправляет Echo обратно
    - Input Server ServerSocketChannel регистрирует SocketChannel от клиента Telnet/Putty
    - SocketChannel обеспечивает передачу данных от и к клиенту Telnet/Putty
    - DatagramChannel клиент получает строку от Socket и отправляет UDP Server
- Важные моменты работы
- UDP Server main() работает на базе Selector, регистрирует DatagramChannel в режиме bind()
    - bind() всегда производится автоматом, если сокет создается конструктором
    - создается с адресом SocketAddress new DatagamSocket(SocketAddress)
    - создается с адресом 0.0.0.0/0.0.0.0 new DatagamSocket()
    - SocketAddress является предком InetSocketAddress можно подставлять напрямую
- UDP Client -работает на базе Selector в отдельном потоке InputServer
    - connect() DatagramChannelвызывается ТОЛЬКО на стороне клиента
    - ByteBuffer Socket и Datagram каналы работают с ПЕРЕКРЕСТНЫМИ ByteBuffer bSC, bDC
    - Socket заполняет bSC и записывает его в Datagram
    - Datagram заполняет bDC и записывает его в Socket
    - перекодировка ВООБЩЕ не используется, так как обмен идет на уровне байт
    - telnet так как Telnet передает по символьно, bSC анализируется на \r\n
    - и только при наличии \r\n идет передача в канал Datagram
- Пример. реализация парсеров String[] args и запуска клиентов
- ```
  // parsers
  private static int parsePort(String[] args, int port) throws NumberFormatException {
      if (args.length > 0) {
          try {
              port = Integer.parseInt(args[0].replaceAll("\\p{Punct}", ""));
          } catch (NumberFormatException e) {
              e.printStackTrace();
          }
      }
      return port;
  }

  private static boolean parseBlock(String[] args, boolean isBlocking) {
      if (args.length > 1) {
          return !args[1].replaceAll("\\p{Punct}", "")
                  .toLowerCase()
                  .contains("n");
      }
      return isBlocking;
  }

  // runners
  public static void runTelnet(String host, int port) throws IOException {
      Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
  }

  private static void runPutty(String host, int port) throws IOException {
      Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
  }
  ```
-
-
- f
  ```

UDP EchoServer на базе Channels и Selector

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- UDP EchoServe на базе Channels и Selector **java8_nio/nio1/selectors/sockets/echo/UDPEchoChannel**
    - свойства используется неблокирующий доступ к чтению канала
- Пример. реализация UDP Server main() **java8_nio/nio1/selectors/sockets/echo/UDPEchoChannel**
-

```
public class UDPEchoChannel {
    private static final Charset TELNET_CHARSET = Charset.forName("CP866");
    private static final Charset UDP_CHARSET = Charset.forName("UTF-8");
    private static final String HOST = "localhost";
    private static final int PORT = 9990;
    private static final int UDP_PORT = 9910;
    private static final int UDP_PORT2 = 9910;
    private static final String[] PUTTY_WELCOME = {
            "Welcome to udp server!",
            "To close Putty without warning message:",
            "Change Putty>>Window>>Behaviour Settings then Save Default Session",
            "Type any text<Enter> (closewindow to exit):"
    };
    public static void runTelnet(String host, int port) throws IOException {     // runners
        Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
    }
    private static void runPutty(String host, int port) throws IOException {
        Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
    }
    private static void registerDatagramChannel(Selector selector,String host, int port) throws IOException{
        DatagramChannel dc = DatagramChannel.open();
        dc.bind(new InetSocketAddress(host,port)); // listening port
        dc.configureBlocking(false);
        dc.register(selector, SelectionKey.OP_READ);
    }
    public static void main(String[] args) {
        System.out.printf("udp server started...%n");
        Selector selector = null;
        ByteBuffer b = ByteBuffer.allocate(1024);
        SocketAddress socketAddress = null;
        try {
            ExecutorService exec = Executors.newCachedThreadPool();
            runPutty(HOST, PORT);
            exec.execute(new InputServer(UDP_CHARSET, PORT, UDP_PORT));  // for putty
            runTelnet(HOST, PORT + 1);
            exec.execute(new InputServer(TELNET_CHARSET, PORT + 1, UDP_PORT + 1));  // for telnet
            exec.shutdown();
            selector = Selector.open();// selector
// udp server
            registerDatagramChannel(selector,HOST,UDP_PORT);
            registerDatagramChannel(selector,HOST,UDP_PORT+1);
            while (true) {
                if (exec.isTerminated())   break;
                int n = selector.select(100);
                if (n == 0) continue;
                Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                while (it.hasNext()) {
                    SelectionKey key = it.next();
                    it.remove();
                    if (key.isReadable()) {
                        DatagramChannel dc = (DatagramChannel) key.channel();
                        b.clear();
                        socketAddress = dc.receive(b); // accumulate in buffer
                        if (socketAddress == null) continue;
                        b.flip();
                        byte[] header = "UDP:".getBytes(UDP_CHARSET);
                        byte[] bytes = new byte[b.remaining()];
                        b.get(bytes).compact().put(header).put(bytes).flip();
                        dc.send(b, socketAddress); // не постоянное соединение, всегда отвечаем адресату
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close( selector);
        }
        System.out.printf("udp server closed...%n");
    }
```

- Пример. реализация UDP Client Input Server **java8_nio/nio1/selectors/sockets/echo/UDPEchoChannel**

```
    private static class InputServer implements Runnable {
        private BufferedReader br;
        private PrintWriter pw;
        private Charset consoleCharset;
        private int port;
        private int udpPort;

        public InputServer(Charset consoleCharset, int port, int udpPort) {
            this.consoleCharset = consoleCharset;
            this.port = port;
            this.udpPort = udpPort;
        }

        @Override
        public void run() {
// socket
            ServerSocket ssc = null;
            Socket sc = null;
            System.out.printf("socket server at s:%d u:%d started...%n",port,udpPort);
// udp
            DatagramChannel dc = null;
            ByteBuffer b = ByteBuffer.allocate(1024);
            InetSocketAddress udpAddress = new InetSocketAddress(HOST, udpPort);

            try {
                ssc = new ServerSocket();
                ssc.bind(new InetSocketAddress(HOST, port));

                sc = ssc.accept();
                br = new BufferedReader(new InputStreamReader(sc.getInputStream(), consoleCharset));
                pw = new PrintWriter(sc.getOutputStream(), true, consoleCharset); // flush by /r/n
                pw.println(Arrays.stream(PUTTY_WELCOME).collect(Collectors.joining(String.format("%n"))));

//udp
                dc = DatagramChannel.open();
                dc.connect(udpAddress);

                while (true) {
                    String s;
                    if ((s = br.readLine()) == null) break;  // client closed
                    if (s.isEmpty()) s = "\r";
// udp
                    b.clear();
                    b.put(s.getBytes(UDP_CHARSET));
                    b.flip();
                    dc.write(b);                        //  dc.send(b,udpAddress);

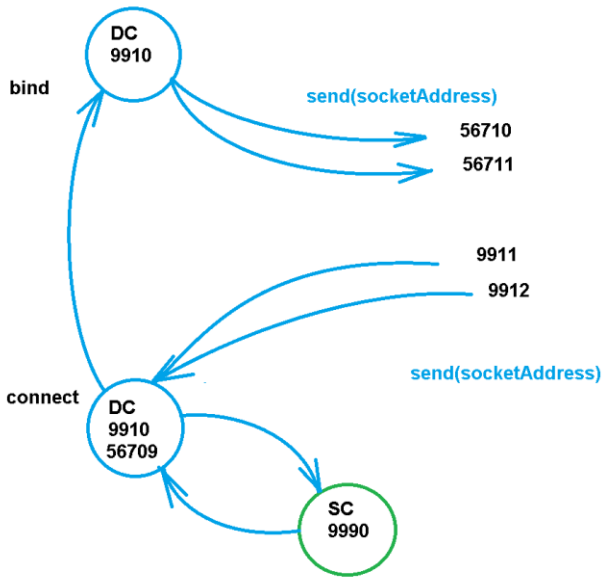

                    b.clear();
                    dc.receive(b);
                    b.flip();
                    s = new String(b.array(), 0, b.limit(), UDP_CHARSET);
                    pw.println(s);


                }


            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.close(ssc, sc, br, pw, dc);
            }
            System.out.printf("socket server at s:%d u:%d closed...%n",port,udpPort);
        }
    }
}
```

UDPChatServer Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- UDPChatServer Example
    - простой ChatServer на базе DatagramChannel и Selector
- Важные моменты
    - терминалы запускается три терминала, два Putty и один Telnet
- Клиенты
    - клиенты на каждый терминал создается и запускается в своем потоке клиент InputServer
    - клиент работает с терминалом через ServerSocketChannel и Selector
    - клиента работает с сервером через DatagramChannel и Selector
    - каналы клиента работают через Selector
    - DatagramChannel заполняет bDC и отправляет данные bDC в SocketChannel
    - получает данные от других клиентов прямой записью в него socketAddress
    - записывает данные командой dc.write() по пути dc.connect() наверх на сервер
    - SocketChannel заполняет bSC и записывает данные bSC в DatagramChannel
    - получает и записывает данные от клиента стандартными read() и write()
    - message при запуске клиента на сервер отправляется сообщение "SC12345 added to chat"
    - сделано чтобы ЗАРЕГИСТРИРОВАТЬ socketAddress в HashMap
- ВНИМАНИЕ. Сервер изначально НЕ ЗНАЕТ socketAdress каналов DatagramChannel для этого нужна HashMap
- Сервер
    - selector регистрирует три DatagramChannel,которые слушают каждый своего клиента
    - при получении данных от клиента, рассылка в DatagramChannel по socketAddress
    - map при первом же сообщении, заполняется map<SocketAddress,DatagramChannel>
    - любые сообщения от клиента в дальнейшем рассылаются по адресам map<>
    - завершение по закрытии канала SocketChannel автоматом закрывается клиент InputServer
    - когда закрыты все InputServer клиенты, сервер это понимает и закрывается сам
- Диаграмма связи UDPChatServer



- ВНИМАНИЕ. Работает с ФИКСИРОВАННЫМ Charset.forName("UTF-8"), с Telnet кракозябры, с Putty нормально

- Пример. реализация UDPChatServer **java8_nio/nio2/network/UDPChatServer**
- ```
  public class UDPChatServer {
      private static final Charset TELNET_CHARSET = Charset.forName("CP866");
      private static final Charset UDP_CHARSET = Charset.forName("UTF-8");

      private static final String HOST = "localhost";
      private static final int PORT = 9990;
      private static final int UDP_PORT = 9910;
      private static final int UDP_PORT2 = 9910;

      private static final String[] PUTTY_WELCOME = {
              "Welcome to udp server!",
              "To close Putty without warning message:",
              "Change Putty>>Window>>Behaviour Settings then Save Default Session",
              "Type any text<Enter> (closewindow to exit):"
      };

      // parsers
      private static int parsePort(String[] args, int port) throws NumberFormatException {
          if (args.length > 0) {
              try {
                  port = Integer.parseInt(args[0].replaceAll("\\p{Punct}", ""));
              } catch (NumberFormatException e) {
                  e.printStackTrace();
              }
          }
          return port;
      }

      private static boolean parseBlock(String[] args, boolean isBlocking) {
          if (args.length > 1) {
              return !args[1].replaceAll("\\p{Punct}", "")
                      .toLowerCase()
                      .contains("n");
          }
          return isBlocking;
      }

      // runners
      public static void runTelnet(String host, int port) throws IOException {
          Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
      }

      public static void runPutty(String host, int port) throws IOException {
          Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
      }

      private static void registerDatagramChannel(Selector selector, String host, int port) throws IOException
  {
          DatagramChannel dc = DatagramChannel.open();
          dc.bind(new InetSocketAddress(host, port)); // listening port
          dc.configureBlocking(false);
          dc.register(selector, SelectionKey.OP_READ);

      }
  ```
  ```

- Пример. реализация UDPChatServer **java8_nio/nio2/network/UDPChatServer**

```
    public static void main(String[] args) {
        System.out.printf("udp server started...%n");
        Selector selector = null;
        ByteBuffer b = ByteBuffer.allocate(1024);
        SocketAddress socketAddress = null;

        try {
            ExecutorService exec = Executors.newCachedThreadPool();
            exec.execute(new InputServer(UDP_CHARSET, PORT, UDP_PORT));  // for putty
            exec.execute(new InputServer(UDP_CHARSET, PORT + 1, UDP_PORT + 1));  // for telnet
            exec.execute(new InputServer(TELNET_CHARSET, PORT + 2, UDP_PORT + 2));  // for telnet
// running alone
            if(args.length == 0) {
                runPutty(HOST, PORT);
                runPutty(HOST, PORT+1);
                runTelnet(HOST, PORT + 2);
            }

            exec.shutdown();
// selector
            selector = Selector.open();
// udp server
            registerDatagramChannel(selector, HOST, UDP_PORT);
            registerDatagramChannel(selector, HOST, UDP_PORT + 1);
            registerDatagramChannel(selector, HOST, UDP_PORT + 2);
            Map<SocketAddress,DatagramChannel> map = new HashMap<>();

            while (true) {
                if (exec.isTerminated()) {
                    break;
                }
                int n = selector.select(100);
                if (n == 0) continue;
                Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                while (it.hasNext()) {
                    SelectionKey key = it.next();
                    it.remove();
                    if (key.isReadable()) {
                        DatagramChannel dc = (DatagramChannel) key.channel();
                        b.clear();
                        socketAddress = dc.receive(b); // accumulate in buffer
                        if (socketAddress == null) continue;
                        map.put(socketAddress,dc);

                        b.flip();
                        byte[] header = "UDP:".getBytes(UDP_CHARSET);
                        byte[] bytes = new byte[b.remaining()];
                        b.get(bytes).compact().put(header).put(bytes).flip();
                        for (SocketAddress address : map.keySet()) {
                            dc = map.get(address);
                            dc.send(b,address);
                            b.rewind();
                        }
                    }
                }
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(selector);
        }

        System.out.printf("udp server closed...%n");
    }
```

- Пример. реализация UDPChatServer **java8_nio/nio2/network/UDPChatServer**

```
    private static class InputServer implements Runnable {
        private BufferedReader br;
        private PrintWriter pw;
        private Charset consoleCharset;
        private int port;
        private int udpPort;

        public InputServer(Charset consoleCharset, int port, int udpPort) {
            this.consoleCharset = consoleCharset;
            this.port = port;
            this.udpPort = udpPort;
        }

        @Override
        public void run() {
// socket
            ServerSocketChannel ssc = null;
            Selector selector = null;
            System.out.printf("socket server at s:%d u:%d started...%n", port, udpPort);
// udp
            ByteBuffer bSC = ByteBuffer.allocate(1024);
            ByteBuffer bDC = ByteBuffer.allocate(1024);
            InetSocketAddress udpAddress = new InetSocketAddress(HOST, udpPort);

            try {
                selector = Selector.open();
                ssc = ServerSocketChannel.open();
                ssc.bind(new InetSocketAddress(HOST, port));
                SocketChannel sc = ssc.accept();
                sc.configureBlocking(false);
                sc.register(selector, SelectionKey.OP_READ | SelectionKey.OP_WRITE, "socket");

                for (String s : PUTTY_WELCOME) {
                    bDC.put(String.format("%s%n", s).getBytes(UDP_CHARSET));
                }
                bSC.put(String.format("SC%d added to chat%n",sc.socket().getPort()).getBytes(UDP_CHARSET));
// регистрация socketAddress в map
//udp
                DatagramChannel dc = DatagramChannel.open();
                dc.connect(udpAddress);
                dc.configureBlocking(false);
                dc.register(selector, SelectionKey.OP_READ | SelectionKey.OP_WRITE, "datagram");

                while (true) {
                    int n = selector.select(100);
                    if (n == 0) break;

                    Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                    while (it.hasNext()) {
                        SelectionKey key = it.next();
                        it.remove();
                        int len;
                        if (key.isReadable()) {
                            if (key.attachment().equals("socket")) {
                                sc = (SocketChannel) key.channel();
                                if ((len = sc.read(bSC)) == -1) {
                                    sc.close();
                                    dc.close();
                                    it.forEachRemaining(SelectionKey::cancel);
                                }
                            } else {
                                dc = (DatagramChannel) key.channel();
                                if ((len = dc.read(bDC)) == -1) {
                                    dc.close();
                                }
                            }
```

- Пример. реализация UDPChatServer **java8_nio/nio2/network/UDPChatServer**
-
```
                        } else if (key.isWritable()) {
                            if (key.attachment().equals("socket")) {
                                sc = (SocketChannel) key.channel();
                                bDC.flip();
                                while (bDC.hasRemaining()) {
                                    sc.write(bDC);
                                }
                                bDC.clear();
                            } else {
                                dc = (DatagramChannel) key.channel();
                                byte[] b = bSC.array();
                                boolean isFound = false;
                                for (int i = 0; i < bSC.position(); i++) {
                                    if (b[i] == '\r' || b[i] == '\n') {
                                        isFound = true;
                                        break;
                                    }
                                }
                                if (!isFound) continue;
                                bSC.flip();
                                while (bSC.hasRemaining()) {
                                    dc.write(bSC);
                                }
                                bSC.clear();
                            }
                        }
                    }

                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    IOUtils.close(ssc, selector);
                }
                System.out.printf("socket server at s:%d u:%d closed...%n", port, udpPort);
            }
        }
    }
```
-
- f

Java NIO2 продолжение

Asynchronous Channels

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- Asynchronous Channels
    - AsynchronousByteChannel
    - AsynchronousFileChannel
    - AsynchronousSocketChannel
    - AsynchronousServerSocketChannel
- Asynchronous Channel это интерфейс для асинхронных каналов, работающих в background
    - есть два способа завершения работы канала instant и callback
    - instant канал сразу возвращает текущее состояние операции обмена и объект Future<T>
    - callback канал завершает операцию обмена данными и вызывает callback по завершении
- Методы
    - operation() Future<T> возвращает объект Future<T> с текущим состоянием обмена
    - Thread должна все это время активно опрашивать канал на Future<T>
    - opetation() CompletionHandler работает до завершения обмена, и затем вызывает Callback
    - Thread просто переводится в состояние Thread.joint() и ждет прерывания от канала
    - Thread.join() перевести Thread в ожидание и выйти если будет послано прерывание
- ВНИМАНИЕ. AsynchronousChannel поддеживают MultiThreading Safe
- ВНИМАНИЕ. AsynchronousChannel используют ByteBuffer которые НЕ ПОДДЕРЖИВАЮТ MultiThreading
- AsynchronousByteChannel интерфейс для асинхронных каналов и памяти
    - read() чтение данных с завершением Future<T> и CompletionHandler
    - write() запись данных с завершением Future<T> и CompletionHandler
- Asynchronous FileChannel интерфейс для асинхронных каналов файлов
    - поддерживает позиционирование в файле, но только в начале операции
    - НЕ ПОДДЕРЖИВАЕТ текущую позицию, так как порядок операций не определен
    - read(position) чтение данных с завершением Future<T> и CompletionHandler
    - write(position) запись данных с завершением Future<T> и CompletionHandler
    - force() форсирует запись данных на носитель
    - truncate() обрезает файл до размера данных, но НЕ увеличивает если файл больше
    - lock() блокирует файл или регион файла
    -

AsynchronousFileChannel Procedure

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousFileChannel Procedure работа с AsycnhronousFileChannel
    - составляющие Future<T>, Callback<T>
    - Future<T> мгновенный возврат результата с данными которые доступны на текущий момент
    - Callback<T> операция блокируется в background и возврат в Callback метод по завершении\
- AsynchronousFileChannel Future<T>
    - open() открыть канал к файлу с Path и опциями StandardOpenOption.READ, WRITE, CREATE
    - read(b, pos) читать в ByteBuffer с позиции pos и вернуть объект Future<T> результат
    - Future<T> чтобы дождаться завершения, ОБЯЗАТЕЛЬНО ожидать Future<T>.isDone()
    - read(b,pos, call) читать в ByteBuffer с позиции pos, результат вернуть в CompletionHandler объект
    - CompletionHandler.completed() вызывается асинхронно по завершении чтения
    - write(b, pos) записать в ByteBuffer с позиции pos и вернуть объект Future<T> результат
    - Future<T> чтобы дождаться завершения, ОБЯЗАТЕЛЬНО ожидать Future<T>.isDone()
    - write(b,pos, call) записать в ByteBuffer с позиции pos, результат вернуть в CompletionHandler объект
    - CompletionHandler.completed() вызывается асинхронно по завершении чтения
- Особые условия
    - Future<T> объект, который отслеживает завершение операции ввода, вывода
    - метод isDone() позволяет определить завершение операции
    - метод get() позволяет получить число байт обмена или -1 если все завершено
- AsynchronousFileChannel CompletionHandler<? extends Integer, A>
    - open() открыть канал к файлу с Path и опциями StandardOpenOption.READ, WRITE, CREATE
    - read(b,pos, call) читать в ByteBuffer с позиции pos, результат вернуть в CompletionHandler объект
    - CompletionHandler.completed() вызывается асинхронно по завершении чтения
    - write(b,pos, call) записать в ByteBuffer с позиции pos, результат вернуть в CompletionHandler объект
    - CompletionHandler.completed() вызывается асинхронно по завершении чтения
- Методы
    - completed() параметры <Integer> Result и Attachment<A> произвольного типа
    - Result возвращает число байт прочитанных или записанных в файл
    - Attachment<A> объект для работы в методе
    - attachment может быть любой объект
    - ByteBuffer тот самый, который был в команде read() в нем будут данные чтения
    - Thread.currentThread() можно безопасно использовать Thread.interrupt()
    - UserClass объект пользователя с любым комплектом объектов
    - failed() параметры Throwable exc и Attachment<A>
    - используется для отработки ошибок обмена данными с файлом
- Особые условия
    - synchronized из completed() нет прямого доступа к полям класса потока main()
    - объекты InternalInt и ByteOutputStream используются для работы в потоках
    - InternalInt клас пользователя, поля boolean isDone, int value и синхронным доступом к ним
    - ByteOutputStream является thread safe, так как у него все методы synchronized
    - поэтому он напрямую работает в потоках main() и CompletionHandler
    - group AsynchronousFileChannel НЕ МОЖЕТ иметь свою группу,
    - но можно сделать в конструкторе задать свою ExecutorService смотреть Callback
- Пример. реализация CompletionHandler с user attachment **java8_nio/nio2/asynch/AsyncFileCallbackThread**

AsynchronousFileChannel Future Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousFileChannel Future Example обмен данными с файлом через объект Future<T>
- Пример. реализация **java8_nio/nio2/asynch/AsyncFileFuture**
- ```
  public class AsyncFileFuture {
      public static void main(String[] args) {
          System.out.printf(FORMAT, "Asynchronous Channel Future<T>:");
          Path path = Paths.get(".", "data", "nio2");
          Path pathC = Paths.get(path.toString(), "async");
          Path pathD = path.resolve("result_k.txt");
          Path pathE = path.resolve("result_w.txt");
          Path pathR;
          BufferedReader br = null;
          FileInputStream in = null;
          AsynchronousChannel ai = null;
          AsynchronousChannel ao = null;

          OpenOption[] options = new OpenOption[]{StandardOpenOption.READ, StandardOpenOption.WRITE,
                  StandardOpenOption.CREATE};
          ByteArrayOutputStream out = null;
          ByteBuffer b = ByteBuffer.allocate(50);
          try {
              ai = AsynchronousFileChannel.open(pathD, options);
              out = new ByteArrayOutputStream(50);
              byte[] bytes = new byte[50];
              int pos = 0;
              int len;
      // read Future
              while (true) {
                  Future<Integer> result = ((AsynchronousFileChannel) ai).read(b, pos);
                  while (!result.isDone()) {
                      Thread.sleep(100);
                      System.out.print(".");
                  }
                  if (result.get() == -1) break;
                  b.flip();
                  while ((len = b.remaining()) > 0) {
                      b.get(bytes, 0, len);
                      pos += len;
                      out.write(bytes, 0, len);
                  }
                  b.compact();
              }
              String s = out.toString(Charset.forName("KOI8-R"));
              System.out.printf("%s%n", s);
      // write future
              pathR = pathC.resolve(pathE.getFileName()); // result_w
              ao = AsynchronousFileChannel.open(pathR, options);
              bytes = s.getBytes(Charset.forName("WINDOWS-1251"));
              pos = 0;
              b.clear();

              while (true) {
                  len = bytes.length - pos > b.limit() ? b.limit() : bytes.length - pos;
                  if(len == 0) break;
                  b.put(bytes, pos, len);
                  b.flip();
                  Future<Integer> result = ((AsynchronousFileChannel) ao).write(b, pos);
                  while (!result.isDone()) {  // ожидать результата завершения IO у Future<T>
                      Thread.sleep(10);
                  }

                  if (result.get() == -1) break;
                  pos += len;
                  b.compact();
              }
          } catch (IOException | InterruptedException | ExecutionException e) {
              e.printStackTrace();
          } finally {
              IOUtils.close(ai, ao);
          }
      }
  }
  ```
-
  ```

AsynchronousFileChannel Callback Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousFileChannel Callback Example обмен данными с файлом через объект CompletionHandler
    - в данном примере используется производный класс, который отправляет в CompletionHandler
    - объект ByteOutputStream для накопления переданных данных
    - размер буфера ByteOutpuStream растет с увеличением размера данных
    - group AsynchronousFileChannel НЕ МОЖЕТ иметь свою группу,
    - но можно сделать в конструкторе задать свою ExecutorService смотреть Callback
- ВНИМАНИЕ. Смотреть здесь ПОДСТАНОВКУ вместо группы своей ExecutorService
- Пример. реализация **java8_nio/nio2/asynch/AsyncFileCallback**

```
public class FileCallback {
    private static final Object lock = new Object();
    private static boolean isDone;
    private static class Callback implements CompletionHandler<Integer, ByteBuffer> {
        private boolean isDone;
        private int result;
        private ByteArrayOutputStream bOut;
        synchronized public boolean isDone() {
            return isDone;
        }
        synchronized public int get() {
            return result;
        }
        synchronized public void set(int result) {
            this.result = result;
            this.isDone = true;
        }
        public Callback(ByteArrayOutputStream bOut) {
            this.isDone = false;
            this.result = -1;
            this.bOut = bOut;
        }
        @Override
        public void completed(Integer result, ByteBuffer b) {
            int len;
            if (result > 0) {
                b.flip();
                bOut.write(b.array(), 0, b.limit());
                set(result);
            } else set(-1);
        }
        @Override
        public void failed(Throwable exc, ByteBuffer b) {
            System.out.printf("Exception:%s%n", exc);
        }
    }
    private static class InternalInt {
        private int value = -1;
        private boolean isDone = false;
        private void set(int value) {
            synchronized (lock) {
                isDone = true;
                this.value = value;
            }
        }
        private int get() {
            synchronized (lock) {
                return value;
            }
        }
        private boolean isDone() {
            synchronized (lock) {
                return isDone;
            }
        }
    }
```

- Пример. реализация **java8_nio/nio2/asynch/AsyncFileCallback**

```
    private static class Factory implements ThreadFactory {
        private ThreadFactory factory;
        private List<Thread> list;

        public Factory(ThreadFactory factory) {
            this.factory = factory;
            this.list = new ArrayList<>();
        }

        @Override
        public Thread newThread(Runnable r) {
            Thread thread = factory.newThread(r);
            list.add(thread);
            return thread;
        }

        private void interruptAll() {
            for (Thread t : list) {
                if (t != null && t.isAlive()) t.interrupt();
            }
        }
    }

    public static void main(String[] args) {
        System.out.printf(FORMAT, "Asynchronous Channel Future<T>:");
        Path path = Paths.get(".", "data", "nio2");
        Path pathC = Paths.get(path.toString(), "async");
        Path pathD = path.resolve("result_w.txt");
        Path pathE = path.resolve("result_k.txt");
        Path pathR;

        AsynchronousChannel ai = null;
        AsynchronousChannel ao = null;

        OpenOption[] options = new OpenOption[]{StandardOpenOption.READ, StandardOpenOption.WRITE,
                StandardOpenOption.CREATE};
        ByteArrayOutputStream byteArrayOutputStream = null;
        ByteBuffer b = ByteBuffer.allocate(50);
        ThreadFactory f = Executors.defaultThreadFactory();
        AsynchronousChannelGroup group = null;
        ThreadFactory factory = null;
        try {
// group of Executors or ThreadPool
            factory = new Factory(Executors.defaultThreadFactory());
            ExecutorService exec = Executors.newCachedThreadPool(factory);
            FileAttribute<List<AclEntry>> fileAttr = MainACL.attributes(path);
            ai = AsynchronousFileChannel.open(pathD, new HashSet<>(),exec);
            byteArrayOutputStream = new ByteArrayOutputStream(50); // init size
// read Future
            int pos = 0;
            int len;
            while (true) {
                Callback callback = new Callback(byteArrayOutputStream);
                b.clear();
                ((AsynchronousFileChannel) ai).read(b, pos, b, callback);  // вместо null можно любой канал
                while (!callback.isDone()) {  // synchronized method
                    Thread.sleep(100);
                    System.out.print(".");
                }
                len = callback.get();
                if (len == -1) break;
                pos += len;
            }
            System.out.println();
            String s = byteArrayOutputStream.toString(Charset.forName("WINDOWS-1251"));
            System.out.printf("%s%n", s);
// write future
            pathR = pathC.resolve(pathE.getFileName()); // result_w
            ao = AsynchronousFileChannel.open(pathR, options);
            ByteBuffer c = ByteBuffer.wrap(s.getBytes(Charset.forName("KOI8-R")));
```

- Пример. реализация **java8_nio/nio2/asynch/AsyncFileCallback**
- ```
  // class for sync
              pos = 0;
              while (true) {
                  b.clear();
                  c.limit(c.capacity());
                  if (c.remaining() == 0) break;
                  len = c.remaining() > b.limit() ? b.limit() : c.remaining();
                  c.limit(pos + len);
                  b.put(c);
                  b.flip();
                  final InternalInt iInt = new InternalInt();
                  ((AsynchronousFileChannel) ao).write(b, pos, null, new CompletionHandler<Integer, Void>() {
                      @Override
                      public void completed(Integer result, Void attachment) {
                          iInt.set(result);
                      }

                      @Override
                      public void failed(Throwable exc, Void attachment) {
                          System.out.printf("Exception:%s%n", exc);
                          iInt.set(-1);
                      }
                  });
                  while (!iInt.isDone()) {  // ожидать результата завершения IO у Future<T>
                      System.out.print(".");
                      Thread.sleep(10);
                  }
                  if (iInt.get() == -1) break; // synchronized method
                  pos = c.position();
              }


          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
          } finally {
              IOUtils.close(ai, ao);
          }
      }
  }
  ```
-
  ```

AsynchronousServerSocketChannel Disclaimer

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousServerSocketChannel Disclaimer
- ВНИМАНИЕ. Пример на AsynchronousServerSocketChannel и AsynchronousSocketChannel НЕ РЕКОМЕНДУЕТСЯ
    - В качестве обучения ЛУЧШЕ взять Chat Server Example

AsynchronousServerSocketChannel Class

- AsynchronousServerSocketChannel класс для асинхронных каналов сервера
    - реализует AsynchronousChannel так как не поддерживает методы read(), write()
    - опции SocketOptions.SO_RCVBUF заставляет сообщить размер буфера данных
    - SocketOptions.SO_REUSEADDR что непонятно, только для MulticastSockets
- Процедура
    - open() открывает канал и он при этом не привязан ни к какому адресу
    - bind() привязывает канал к конкретному адресу SERVER:PORT
    - accept() запускает прослушивание канала, то есть разрешение на любое соединение
- CompletionHandler<SocketChannel,<? extends A>
    - accept() запускает заново прослушивание на новые входящие соединения кроме текущего
    - read() запускает процедуру чтение данных клиента с другим CompetionHandler<Integer,A>
    - теперь это CompletionHandler<Integer,? extends A> первый тип Integer
- CompletionHandler<Integer, <? extends A>
    - Integer result передает число байт, прочитанных в ByteBuffer
    - Attachment содержит ByteBuffer, в котором данные от клиента,их можно использовать далее
    - после обработки данных буфера просто выйти из метода completed()
- ВНИМАНИЕ. AsynchronousServerSocketChannel является Safe for MultiThreading
- Пример. реализация AsynchronousServerSocket **java8_nio/nio2/Main05A**

```
public class MainServerSocket {
    private static final int PORT = 9090;
    private static final String HOST = "localhost";

    public static void main(String[] args) {
        AsynchronousServerSocketChannel serverChannel = null;
        try {
            serverChannel = AsynchronousServerSocketChannel.open();
            serverChannel.bind(new InetSocketAddress(HOST, PORT));

            System.out.printf("Server listening at:%s%n", serverChannel.getLocalAddress());
            AttachmentServer attachment = new AttachmentServer();
            attachment.serverChannel = serverChannel;
            attachment.serverThread = Thread.currentThread();
            serverChannel.accept(attachment, new MyConnHandler());

            System.out.printf("Waiting 10 sec for connections...%n");
            Thread.currentThread().join(0);

        } catch (IOException e) {
            System.out.printf("Unable to open or bind AsynchronousServerSocketChannel:%s%n", e);
        } catch (InterruptedException e) {
            System.out.printf("Server terminating:%s%n", e);
        } finally {
            IOUtils.close(serverChannel);
        }
        System.out.printf("Server closed%n");

        try {
            Thread.sleep(500);   // должно быть больше чем у сервера при завершении
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

- Пример. реализация Attachment и CompletionHandler для Server **java8_nio/nio/Main05A**

```
public class AttachmentServer {
    public AsynchronousServerSocketChannel serverChannel;
    public AsynchronousSocketChannel clientChannel;
    public boolean isReadMode;
    public ByteBuffer buffer;
    public SocketAddress clientAddress;
    public Thread serverThread;
}
public class MyConnHandler implements CompletionHandler<AsynchronousSocketChannel,AttachmentServer> {
    public  static final int BUFFER_SIZE = 400;
    public static final Charset CHARSET = Charset.forName("KOI8-R");//StandardCharsets.UTF_8;
    public static final String CLOSE_SERVER ="close";
    @Override
    public void completed(AsynchronousSocketChannel clientChannel, AttachmentServer attachment) {
        try {
            SocketAddress clientAddress = clientChannel.getRemoteAddress();
            System.out.printf("Accepted connection from:%s local:%s%n",clientAddress,
                                                                    clientChannel.getLocalAddress());
            attachment.serverChannel.accept(attachment,this);     // используется только поле serverChannel
            AttachmentServer myAttachment = new AttachmentServer();
            myAttachment.serverChannel = attachment.serverChannel;
            myAttachment.clientChannel = clientChannel;
            myAttachment.isReadMode = true;
            myAttachment.buffer = ByteBuffer.allocate(BUFFER_SIZE);
            myAttachment.clientAddress = clientAddress;
            myAttachment.serverThread = attachment.serverThread;
            RWHandlerServer rwHandler = new RWHandlerServer();
            clientChannel.read(myAttachment.buffer,myAttachment,rwHandler);
        } catch (IOException e) {           // attachment предоставляет rwHandler доступ к собственно буферу
        }
    }
    @Override
    public void failed(Throwable exc, AttachmentServer attachment) {
        System.out.printf("Server failed to accept connection%n");
    }
}
public class RWHandlerServer implements CompletionHandler<Integer, AttachmentServer> {
    @Override
    public void completed(Integer result, AttachmentServer attachment) {
        if (result == -1) {  // accept not provided
            try {
                SocketAddress address = attachment.clientChannel.getRemoteAddress();
                attachment.clientChannel.close();
                System.out.printf("Stopped listening to client from:%s%n", address);
                return;
            } catch (IOException e) {
            }
        }
        if (attachment.isReadMode) {                            // read
            attachment.buffer.flip();
            int limit = attachment.buffer.limit();
            byte[] bytes = new byte[limit];
            attachment.buffer.get(bytes); // буфер прямо в размер данных
            String s = new String(bytes, MyConnHandler.CHARSET);
            System.out.printf("Client sent:%s%n", s); // text demo
            attachment.buffer.clear();
            attachment.buffer.put(s.getBytes(MyConnHandler.CHARSET));
            attachment.buffer.flip();
            attachment.isReadMode = false;
            attachment.buffer.rewind();
            attachment.clientChannel.write(attachment.buffer, attachment, this);
            if (s.matches(MyConnHandler.CLOSE_SERVER)) {       // close server by command
                attachment.serverThread.interrupt();
            }
        } else {                                                // write
            attachment.isReadMode = true;
            attachment.buffer.clear();
            attachment.clientChannel.read(attachment.buffer, attachment, this); //this Attachment type
        }
    }
    @Override
    public void failed(Throwable exc, AttachmentServer attachment) {
        System.out.printf("Connection with client broken%n");
    }
}
```

AsynchronousSocketChannel Disclaimer

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousSocketChannel Disclaimer
- ВНИМАНИЕ. Пример на AsynchronousServerSocketChannel и AsynchronousSocketChannel НЕ РЕКОМЕНДУЕТСЯ
    - В качестве обучения ЛУЧШЕ взять Chat Server Example

AsynchronousSocketChannel Class

- AsynchronousSocketChannel класс для асинхронных каналов клиента
    - реализует AsynchronousByteChannel
    - опции SocketOptions.SO_RCVBUF требует сообщить размер буфера приема
    - SocketOptions.SO_REUSEADDR что непонятно, только для MulticastSockets
    - SocketOptions.SO_SNDBUF требует сообщить размер буфера передачи
    - SocketOptions.SO_KEEPALIVE держит Socket alive в течение 2х часов
    - SocketOptions.TCP_NODELAY данные на передачу не буферизуются
- Процедура
    - open() открывает клиент, который не привязан ни к какому адресу
    - connect() подключается к серверу, если сервер еще не слушает порт, выбрасывает Exception
    - NotYetConnectedException
    - getRemoteAddress() позволяет ОПРЕДЕЛИТЬ подключен канал к серверу или нет
- ВНИМАНИЕ. AsynchronousSocketChannel является Safe for MultiThreading
- ВНИМАНИЕ. getRemoteAddress() возвращает null если канал НЕ ПОДКЛЮЧЕН к серверу или РЕАЛЬНЫЙ адрес
- Пример. реализация Attachment и CompletionHandler для Client **java8_nio/nio/Main05A**

```
public class MainClientSocket {
    private static final int PORT = 9090;
    private static final String HOST = "localhost";

    public static void main(String[] args) {
        AsynchronousSocketChannel clientChannel = null;
        try {
            clientChannel = AsynchronousSocketChannel.open();
            clientChannel.connect(new InetSocketAddress(HOST, PORT));
            System.out.printf("Client at %s connected%n", clientChannel.getLocalAddress());

            AttachmentClient attachment = new AttachmentClient();
            attachment.clientChannel = clientChannel;
            attachment.isReadMode = false;
            attachment.buffer = ByteBuffer.allocate(MyConnHandler.BUFFER_SIZE);
            attachment.clientThread = Thread.currentThread();

// записать первое сообщение серверу
            byte[] bytes =
                "Type сообщение(<close> to close Server, Client and exit...".getBytes(MyConnHandler.CHARSET);
            attachment.buffer.put(bytes); // все данные в буфер
            attachment.buffer.flip();
            clientChannel.write(attachment.buffer, attachment, new RWHandlerClient());

            System.out.printf("Client sent data to Server and waiting for answer...%n");
            System.out.printf("Type message(<Enter> to exit...%n");

// ожидаем работы с сервером через CompletionHandler<Integer,<? extends A> RWHandlerClient
//             Thread.currentThread().join();  // тоже самое
            attachment.clientThread.join();
        } catch (IOException | IllegalStateException e) {
            System.out.printf("Unable to open AsynchronousSocketChannel:%s%n", e);
        } catch (InterruptedException e) {
            System.out.printf("Client interrupted:%s%n", e);
        } finally {
            IOUtils.close(clientChannel);
        }
        System.out.printf("Client closed%n");
    }
}
```

- Пример. реализация AsynchronousSocket для Client **java8_nio/nio/Main05A**
- ```
  public class AttachmentClient {
      public AsynchronousSocketChannel clientChannel;
      public boolean isReadMode;
      public ByteBuffer buffer;
      public Thread clientThread;
  }
  ```
- ```
  public class RWHandlerClient implements CompletionHandler<Integer, AttachmentClient> {
      private BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); // клиент с клавиатурой

      @Override
      public void completed(Integer result, AttachmentClient attachment) {
          if (attachment.isReadMode) {
  // read
              attachment.buffer.flip();
              int limit = attachment.buffer.limit();
              byte[] bytes = new byte[limit];
              attachment.buffer.get(bytes); // буфер прямо в размер данных
              String s =new String(bytes, MyConnHandler.CHARSET);
  // text demo
              System.out.printf("Server responded:%s%n", s);
  // text demo
              if(s.matches(MyConnHandler.CLOSE_SERVER)) {
                  try {
                      Thread.sleep(1000);
                  } catch (InterruptedException e) {
                      e.printStackTrace();
                  }
                  attachment.clientThread.interrupt();
                  return;
              }
  // new message
              try {
                  s = "";
                  while (s.length() == 0) {
                      s = br.readLine();
                      if (s.isEmpty()) {
                          attachment.clientThread.interrupt(); // interrupt Client and exit from handler
                          return;
                      }
                  }
                  attachment.isReadMode = false;
                  attachment.buffer.clear();
                  bytes = s.getBytes(MyConnHandler.CHARSET);
                  attachment.buffer.put(bytes);
                  attachment.buffer.flip();
                  attachment.clientChannel.write(attachment.buffer, attachment, this);
              } catch (IOException e) {
                  System.out.printf("Unable read from console:%s%n", e);
              }
          } else {
  // read
              attachment.isReadMode = true;
              attachment.buffer.clear();
              attachment.clientChannel.read(attachment.buffer, attachment, this); // this  Attachment type
          }
      }

      @Override
      public void failed(Throwable exc, AttachmentClient attachment) {
          System.out.printf("Connection with client broken%n");
          attachment.clientThread.interrupt();
      }
  }
  ```
-

Asynchronous Channel Groups

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousChannelGroup группирование AsynchronousChannels в группы для разделения ресурсов
    - group имеет ассоциированный ThreadPool, в котором Threads с потоками подписаны в группу
    - default group это группа куда метод open() подписывает ServerSocket или Socket по умолчанию
- Методы
    - withCachedThreadPool создает AsynchronousChannelGroup с неограниченным числом потоков
    - использует Executors.newCachedThreadPool() ExecutorService
    - withFixedThreadPool создает AsynchronousChannelGroup с фиксированным числом потоков
    - использует Executors.defaultThreadFactory() ThreadFactory
    - withSingleThreadPool создает AsynchronousChannelGroup с одним потоком исполнения
    - использует Executors.newSingleThreadExecutor() ExecutorService
    - Executors.defaultThreadFactory фабрика потоков, которую использует AsynchronousChannelGroup
    - shutdown() работает только если ВСЕ каналы ЗАКРЫТЫ и ВСЕ CompletionHandler ЗАВЕРШЕНЫ
- ВНИМАНИЕ. Для вызова group.shutdown() ОБЯЗАТЕЛЬНО ЗАКРЫТЬ каналы, и ВЫЙТИ из CompletionHandler
- Реализация
    - создается группа, которая является по сути ThreadPool и будет раздавать Thread каналам
    - работа группа прописывается в канале при открытии
    - завершение группы, как обычной ExecutroService, главное если есть незакрытые каналы или
    - незавершенный CompletionHandler, который скажем ожидает ввода с System.in,
    - группа не закроется, так как группа НЕ ЗАНИМАЕТСЯ прерыванием потоков.
- Пример. реализация для AsynchronousServerSocketChannel **java8_nio/nio/Main05A**
- ```
  public class MainServerSocket {
      private static final int PORT = 9090;
      private static final String HOST = "localhost";
      public static void main(String[] args) {
          AsynchronousServerSocketChannel serverChannel = null;
          AsynchronousChannelGroup group = null;
          try {
              group = AsynchronousChannelGroup.withCachedThreadPool(Executors.newCachedThreadPool(), 2);
              serverChannel = AsynchronousServerSocketChannel.open(group);
              serverChannel.bind(new InetSocketAddress(HOST, PORT));
              System.out.printf("Server listening at:%s%n", serverChannel.getLocalAddress());
              AttachmentServer attachment = new AttachmentServer();
              attachment.serverChannel = serverChannel;
              attachment.serverThread = Thread.currentThread();
              serverChannel.accept(attachment, new MyConnHandler());
              System.out.printf("Waiting 10 sec for connections...%n");
              System.out.printf("**********************************%n");
              Thread.currentThread().join(10000);
          } catch (IOException e) {
              System.out.printf("Unable to open or bind AsynchronousServerSocketChannel:%s%n", e);
          } catch (InterruptedException e) {
              System.out.printf("Server terminating:%s%n", e);
          } finally {
              IOUtils.close(serverChannel);
          }
          System.out.printf("Server closed%n");
          try {
              if (group != null) {
                  System.out.printf("Shutdown group...%n");
                  group.shutdown();
                  if (!group.isTerminated()) group.awaitTermination(5, TimeUnit.SECONDS);

                  if (group.isTerminated())   System.out.printf("group is shutdown%n");
                  else    System.out.printf("group is NOT shutdown%n");
              }
          } catch (InterruptedException e) {
              e.printStackTrace();
          }
      }
  }
  ```
  ```

AsynchronousChannelGroup Example with Interrupt of CompletionHandler

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- AsynchronousChannelGroup Example with Interrupt of CompletionHandler
  - закрытие группы происходит за счет прерывания Thread of CompletionHandler
  - CompletionHandler сделан с нарушением, там бесконечный while() ввода System.in
  - решение сделано за счет вытаскивания Thread объектов из custom ThreadFactory и рассылки
  - прерываний в все созданные Thread перед тем как закрыть группу shutdown()
- Пример. реализация для AsynchronousSocketChannel **java8_nio/nio/Main05A**

```
public class MainClientSocket {
    private static final int PORT = 9090;
    private static final String HOST = "localhost";
    public static void main(String[] args) {
        AsynchronousSocketChannel clientChannel = null;
        AsynchronousChannelGroup group = null;
        Factory factory = null;
        try {
//            group = AsynchronousChannelGroup.withCachedThreadPool(Executors.newCachedThreadPool(), 1);
            factory = new Factory(Executors.defaultThreadFactory());
            group = AsynchronousChannelGroup.withFixedThreadPool(2, factory);
            clientChannel = AsynchronousSocketChannel.open(group);
...
        try {
            System.out.printf("Shutdown group...%n");
            if (group != null) {
                factory.interruptAll();
                group.shutdown();
                if (!group.isTerminated()) {
                    group.shutdownNow();
                    group.awaitTermination(2, TimeUnit.SECONDS);
                }
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
        if (group.isTerminated()) {
            System.out.printf("group is shutdown%n");
        } else {
            System.out.printf("group is NOT shutdown%n");
        }
// demo
    }
}
```

- Пример. реализация работы AsynchronousChannelGroup класс custom Factory **java8_nio/nio/Main05A**

```
    public class Factory implements ThreadFactory {
        private ThreadFactory factory;
        private List<Thread> list;

        public Factory(ThreadFactory factory) {
            this.factory = factory;
            this.list = new ArrayList<>();
        }

        @Override
        public Thread newThread(Runnable r) {
            Thread thread = factory.newThread(r);
            list.add(thread);
            return thread;
        }

        private void interruptAll() {
            for (Thread t : list) {
                if (t != null && t.isAlive()) t.interrupt();
            }
        }
    }
```

NetworkChannel

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- NetworkChannel Interface NetworkChannel
    - интерфейс который расширяет операции bind() и option() SocketChannel интерфейса
- Методы
    - bind(SocketAddress) привязывает Listener канал к адресу
    - getLocalAddress() возвращает адрес Socket к которому привязан канал
    - getOption(SocketOption<T>) возвращает значение SocketOption<T>
    - setOption(SocketOption<T>, T value) задает значение SocketOption<T>
    - supportedOptions() возвращает Set<SocketOption<?>>
    - open(Path,Set<OpenOption>, ExecutorService, FileAttribute) открывает канал в заданной группе
- Новые Методы
    - DatagramChannel.getRemoteAddress() возвращает удаленный адрес Socket
    - ServerSocketChannel.bind(SocketAddress, int) задает МАКСИМАЛЬНОЕ соединений в ожидании
    - SocketChannel.shutdownInput() закрывает соединение по чтению, канал не закрывает
    - SocketChannel.shutdownOutput() закрывает соединение по записи, канал не закрывает
- Пример. реализация ChatServer NetworkChannel и адаптивным Charset **java8_nio/nio2/network/ChatServer**
- ```
  public class ChatServer {
      private static final String HOST = "localhost";
      private static final int PORT = 9990;
      private static final long SESSION = 50000;
      private static final Charset TELNET_CHARSET = Charset.forName("CP866");
      private static final Charset PUTTY_CHARSET = Charset.forName("UTF-8");
      private static final ByteBuffer b = ByteBuffer.allocate(2048);
      private static int count = 0;

      public static void runTelnet(String host, int port) throws IOException {
          Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
      }

      public static void runPutty(String host, int port) throws IOException {
          Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
      }

      public static void main(String[] args) {
          ServerSocketChannel ssc = null;
          SocketChannel sc = null;
          Selector selector = null;
          try {
              ssc = ServerSocketChannel.open();
  //            ssc.socket().bind(new InetSocketAddress(HOST, PORT));     // вместо socket().bind()
              ssc.bind(new InetSocketAddress(HOST, PORT));                  // bind()
              LocalDateTime finishTime = LocalDateTime.now().plus(SESSION, ChronoUnit.MILLIS);
              ssc.configureBlocking(false);
              selector = Selector.open();
              ssc.register(selector, SelectionKey.OP_ACCEPT);
              System.out.printf("server started at:%s...%n", ssc.getLocalAddress());  // getLocalAddress()
              System.out.println("Supported Options:");
              for (SocketOption<?> option : ssc.supportedOptions()) {
                  System.out.printf("%-10s:%s%n", option.name(), option.type().getSimpleName());
              }
  // clients
              runTelnet(HOST, PORT);
              runTelnet(HOST, PORT);
              runPutty(HOST, PORT);
  ```
  ```

- Пример. реализация ChatServer NetworkChannel и адаптивным Charset **java8_nio/nio2/network/ChatServer**

```
            while (!LocalDateTime.now().isAfter(finishTime)) {
                int n = selector.select(100);
                if (n == 0) {
                    if (selector.keys().size() > 1 || count == 0) continue;
                    else break;
                }
                Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                while (it.hasNext()) {
                    SelectionKey key = it.next();
                    it.remove();
                    if (key.isAcceptable()) {
                        sc = ((ServerSocketChannel) key.channel()).accept();
                        if (sc == null) continue;
                        sc.configureBlocking(false);
                        UserAttachment attachment = new UserAttachment(sc.socket().getPort());
                        sc.register(selector, SelectionKey.OP_READ, attachment);
                        sendMessage(sc, String.format("Welcome to NIO Chat%n"), attachment.charset());
                        System.out.printf("%s %s:entered to chat%n",
                                                                sc.getRemoteAddress(), sc.getLocalAddress());
                        count++;

                    } else if (key.isReadable()) {  // data arrived from channel
                        sc = (SocketChannel) key.channel();
                        if (sc == null) continue;
// read
                        String s = readMessage(sc, key);
                        if (s == null) {
                            System.out.printf("%s:left chat%n", sc.getRemoteAddress());
                            sc.close();         // closes all keys connected with channel
                        } else if (s.length() > 0) {
                            broadcast(selector, key, s);
                        }
                    }
                }
            }

        } catch (IOException e) {
            System.out.printf("Exception:%s%n", e);
        } finally {
            IOUtils.close(ssc, selector);
        }
        System.out.printf("server closed...%n");
    }

    private static String readMessage(SocketChannel sc, SelectionKey key) throws IOException {
        UserAttachment att = (UserAttachment) key.attachment();
        try {
            b.clear();
            int len;
            if ((len = sc.read(b)) > 0) {
                b.flip();
                String s = att.checkCharset(new String(b.array(), 0, b.limit(), att.charset()));
                return s;
            } else if (len == -1) {
                return null;
            }
        } catch (IOException e) {
            return null;
        }
        return "";
    }
    private static void sendMessage(SocketChannel sc, String s, Charset charset) throws IOException {
        b.clear();
//        String message = String.format("SC%d:%s", sc.socket().getPort(), s);
//        b.put(message.getBytes(TELNET_CHARSET));

        b.put(s.getBytes(charset));
        b.flip();
        while (b.hasRemaining()) {
            sc.write(b);
        }
    }
```

- Пример. реализация ChatServer NetworkChannel и адаптивным Charset **java8_nio/nio2/network/ChatServer**

```
    private static void broadcast(Selector selector, SelectionKey src, String s) throws IOException {
        Set<SelectionKey> set = selector.keys();
        for (SelectionKey key : set) {
            if (key == src) continue;
            if (key.isValid() && (key.interestOps() & SelectionKey.OP_READ) > 0) {
                SocketChannel sc = (SocketChannel) key.channel();
                Charset charset = ((UserAttachment) key.attachment()).charset();
                sendMessage(sc, s, charset);
            }
        }
    }
```

```
    private static class UserTask {
        private final SocketChannel sc;
        private final String message;

        public UserTask(SocketChannel sc, String message) {
            this.sc = sc;
            this.message = message;
        }
    }
```

```
    private static class UserAttachment {
        private final String message;
        private boolean isChanged;
        private Charset charset;

        public UserAttachment(int port) {
            this.message = String.format("SC%d", port);
            this.isChanged = false;
            this.charset = TELNET_CHARSET;
        }

        private void changeCharset() {
            isChanged = true;
            charset = PUTTY_CHARSET;
        }

        public Charset charset() {
            return charset;
        }

        public boolean isChanged() {
            return isChanged;
        }

        public void setChanged() {
            isChanged = true;
        }

        public String checkCharset(String s) {
            if (!isChanged) {
                for (char c : s.toCharArray()) {
                    if (c > 0x2000) {
                        changeCharset();
                        return new String(b.array(), 0, b.limit(), charset);
                    }
                }
                for (char c : s.toCharArray()) {
                    if (c > 0x0400) {
                        setChanged();
                        return s;
                    }
                }
            }
            return s;
        }
    }
}
```

-
- f

MultiCastChannel

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- MultiCastChannel
    - интерфейс расширяет NetworkChannel и реализован в DatagramChannel.class
    - позволяет рассылать Datagram группе клиентов
    - multicast group это группа адресов 224.0.0.1 - 239.255.255.255
    - receiver может join() присоединиться к группе и drop() выйти из группы
- Методы
    - join() возвращает MemberShipKey присоединяет клиента к группе
    - NetworkInterface объект возвращает метод NetworkInterface.getByInetAddress(InetAddress)
    - join(group, NetworkInterface, source) позволяет получать от конкретного адреса в группе
    - block() заблокировать сообщения от конкретного адреса
    - unblock() сгять блокировку
    - channel() возвращает канал с которого MemberShipKey был получен
    - drop() выйти и группы
    - isValid() проверяет валидность членства в группе
    - networkInterface() возвращает NetworkInterface с которого MemberShipKey получен
- ВНИМАНИЕ. Методы можно объединять в цепь, так как все они возвращают MemberShipKey
- Важные моменты
    - при создании DatagramChannel ОБЯЗАТЕЛЬНО указать протокол обмена INET(IPv4), INET6(IPv6)
    - ChannelSocket должен быть привязан к ГРУППЕ адресов, иначе MultCasting не сработает
    - SO_REUSEADDR должна быть разрешена ДО операции binding к socket
- Присоединение к группе
    - уточняющее то есть сначала присоединиться к группе, затем уточнить конкретные источники
    - join().join(group,source) называется include-mode filtering
    - блокирующее присоединиться к группе, затем заблокировать некоторые источники
    - join().block(source) называется exclude-mode filtering

MultiCastChannel Server Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- MultiCastChannel Server Example
    - открыть DatagramChannel и привязывается bind() к конкретному PORT
    - задать ОБЯЗАТЕЛЬНО опции INET, SO_REUSEADDR , IP_MULTICAST_IF и non blocking
    - рассылка теперь канал может рассылать группе send(groupAddress, NetworkInterface)
    - прием канал можетподписаться на другие группы join(groupAddress, NetworkInterface)
- Важные моменты
    - в примере канал создает группу на передачу 239.255.0.1 и подписывается на другие группы
    - НЕЛЬЗЯ подписываться на свою же группу передачи иначе зацикливается канал
- ВНИМАНИЕ. НЕЛЬЗЯ подписываться на свою же группу передачи иначе зацикливается канал
- Пример. реализация рассылки Datagram по MilticastChannel Server **java8_nio/nio2/network/MultiCastServer**
- ```
  public class MultiCastServer {
      private static final String HOST = "localhost";
      private static final int PORT = 9990;
      private static final String GROUP_LOCAL = "239.255.0.1";
      private static final String GROUP_MASK = "239.255.0.";
      private static final Charset UTF_CHARSET = Charset.forName("UTF-8");

      public static void main(String[] args) {
          NetworkInterface ni = null;
          DatagramChannel dc = null;
          ByteBuffer b = ByteBuffer.allocate(50);
          try {
              ni = NetworkInterface.getByInetAddress(InetAddress.getByName(HOST));
              dc = DatagramChannel.open(StandardProtocolFamily.INET)
                      .setOption(StandardSocketOptions.SO_REUSEADDR, true)
                      .bind(new InetSocketAddress(PORT))
                      .setOption(StandardSocketOptions.IP_MULTICAST_IF, ni);
              dc.configureBlocking(false);
              for (int i = 0; i < 10; i++) {
                  String s = GROUP_MASK + i;
                  if(s.equals(GROUP_LOCAL)) continue;
                  dc.join(InetAddress.getByName(s), ni);  // группа ввода
              }
              InetAddress group = InetAddress.getByName(GROUP_LOCAL); // группа на вывод
              System.out.printf("server started at:%s group:%s%n",
                                                      dc.getLocalAddress(), group.getHostAddress());
              int i = 0;
              while (true) {
                  b.clear();
                  b.put(("line" + i).getBytes(Charset.forName("UTF-8")));
                  b.flip();
                  dc.send(b, new InetSocketAddress(group, PORT));
                  i++;
                  if (i == 1050) break;
  // check
                  b.clear();
                  SocketAddress sa = dc.receive(b);
                  if(b.position() > 0) {
                      b.flip();
                      System.out.printf("%n%s:%s%n",sa,new String(b.array(),0,b.limit(),UTF_CHARSET));
                  }
                  Thread.sleep(400);
                  System.out.print(".");
              }
              System.out.println();
  // send exit
              b = ByteBuffer.wrap((GROUP_MASK + ":exit").getBytes(UTF_CHARSET));
              dc.send(b, new InetSocketAddress(group, PORT)); // exit from the group
              b.clear();
          } catch (IOException | InterruptedException e) {
              e.printStackTrace();
          } finally {
              IOUtils.close(dc);
          }
          System.out.println("server stopped...");
      }
  }
  ```
  ```

MultiCastChannel Client Example

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- MultiCastChannel Client Example
    - открыть DatagramChannel и привязывается bind() к конкретному PORT
    - задать ОБЯЗАТЕЛЬНО опции INET, SO_REUSEADDR , IP_MULTICAST_IF и non blocking
    - прием канал подписывается на группы сервера join(groupAddress, NetworkInterface)
    - рассылка канал может создать группу рассылки send(groupAddress, NetworkInterface)
    -
- Важные моменты
    - в примере канал создает группу на передачу 239.255.0.2 и подписывается на другие группы
    - НЕЛЬЗЯ подписываться на свою же группу передачи иначе зацикливается канал
- ВНИМАНИЕ. НЕЛЬЗЯ подписываться на свою же группу передачи иначе зацикливается канал
- Пример. реализация рассылки Datagram по MilticastChannel Client **java8_nio/nio2/network/MultiCastClient**
- ```
  public class MultiCastClient {
      private static final String HOST = "localhost";
      private static final int PORT = 9990;
      private static final String GROUP_LOCAL = "239.255.0.2";
      private static final String GROUP_MASK = "239.255.0.";
      private static final Charset UTF_CHARSET = Charset.forName("UTF-8");


      public static void main(String[] args) {

          NetworkInterface ni = null;
          DatagramChannel dc = null;
          MembershipKey key = null;
          ByteBuffer b = ByteBuffer.allocate(100);
          try {
              ni = NetworkInterface.getByInetAddress(InetAddress.getByName(HOST));
              dc = DatagramChannel.open(StandardProtocolFamily.INET)
                      .setOption(StandardSocketOptions.SO_REUSEADDR, true)
                      .bind(new InetSocketAddress(PORT))
                      .setOption(StandardSocketOptions.IP_MULTICAST_IF, ni);

              for (int i = 0; i < 10; i++) {
                  String s = GROUP_MASK + i;
                  if(s.equals(GROUP_LOCAL)) continue;
                  dc.join(InetAddress.getByName(s), ni);  // группа ввода
              }
              InetAddress group = InetAddress.getByName(GROUP_LOCAL); // группа вывода

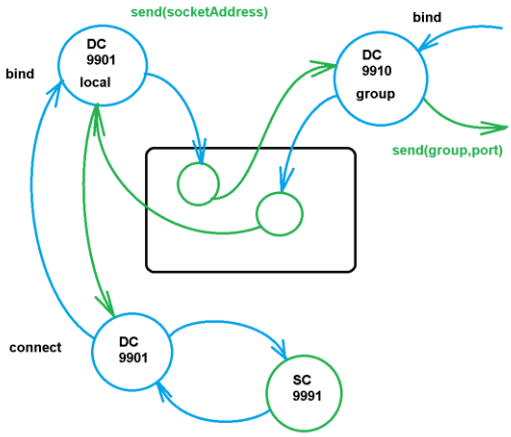
              dc.configureBlocking(false);
              while (true) {
                  dc.receive(b);

                  if (b.position() == 0) continue;
                  b.flip();
                  String s = new String(b.array(), 0, b.limit(), UTF_CHARSET);

                  if (s.equals(GROUP_MASK + ":exit")) {
                      break;
                  }
                  System.out.println(s);
                  b.clear();
                  ByteBuffer c = ByteBuffer.allocate(100);
                  c.put(String.format("from client :%s ping",GROUP_LOCAL).getBytes(UTF_CHARSET));
                  c.flip();
                  dc.send(c,new InetSocketAddress(group, PORT));  // отправлять в свою группу
                  Thread.sleep(500);
              }
          } catch (IOException |InterruptedException e) {
              e.printStackTrace();
          } finally {
              IOUtils.close(dc);
          }
          System.out.println("client closed...");
      }
  }
  ```
  ```

UDPMultiCastChatServer

- Java NIO Java NIO1 Update Java NIO2 Components Java NIO2 продолжение Files Static Methods
- UDPMultiCastChatServer
    - полноценный ChatServer на базе DatagramChannel использует MultiCastChannel для рассылки
- Важные моменты
    - организация каждый канал это автономный сервер и клиент одновременно
- Сервер открывает два DatagramChannel канала и запускаетв потоке InputServer()
    - inputServer() работает со своим терминалом, Putty или Telnet
    - канал сервера DatagramChannel сервера присоединяется к группам 239.255.0.1-239.255.0.10
    - при этом свою группу он не прослушивает, а использует для рассылки
    - канал сервера получает сообщения от других групп и пересылает их клиенту
    - используя DatagramChannel клиента как передатчик к InputServer
    - канал клиента DatagramChannel клиента открывает bind() на отдельный уникальный PORT
    - DatagramChannel inputServer присоединяется connect() к этому же порту
    - канал клиента получает сообщения от InputServer и пересылает их серверу
    - используя DatagramChannel сервера как передатчик к группам
    - Selector для прослушивания каналов сервера и клиента используется Selector
- InputServer регистрирует DatagramChannel и SocketChannel на уникальные порты
    - порты InputServer и порт клиента сервера должны быть уникальны
    - DatagramChannel принимает сообщения от сервера и отсылает SocketChannel
    - SocketChannel принимает сообщения от терминала отсылает DatagramChannel
    - charset Charset отслеживается и используется следующим образом
    - InputServer принимает и передает SocketChannel в локальном своем Charset
    - принимает и передает серверу в едином Charset.forName("UTF-8")
    - Telnet input Telnet передает по одному символу, поэтому InputServer анализирует ввод и
    - не передает сообщение, пока не получен хотя бы один символ перевода строки
    - на сервер сообщения передаются по строкам независимо от типа терминала
    - уникальные порты и адреса
    - group 239.255.0.1-10 уникальный адрес для каждого сервера клиента чата
    - groupPort 9910 единый адрес для всех сервер клиентов чата
    - udpPort 9901 уникальный порт для каждого сервер клиента чата
    - socketPort 9991 уникальный порт для каждого сервер клиента чата
- Диаграмма соединений



-

- Пример. реализация UDPMultiCastChat **java8_nio/nio2/network/UDPMultiCastChat**

```
public class UDPMultiCastChat {
    private static final Charset TELNET_CHARSET = Charset.forName("CP866");
    private static final Charset UTF_CHARSET = Charset.forName("UTF-8");

    private static final String HOST = "localhost";
    private static final int PORT = 1;
    private static final int SOCKET_MASK = 9990;
    private static final int GROUP_PORT = 9910;
    private static final int UDP_MASK = 9900;

    private static final String GROUP_MASK = "239.255.0.";
    private static final int GROUP_INDEX = -1;


    private static final String[] PUTTY_WELCOME = {
            "Welcome to udp server!",
            "To close Putty without warning message:",
            "Change Putty>>Window>>Behaviour Settings then Save Default Session",
            "Type any text<Enter> (closewindow to exit):"
    };

    // parsers
    private static int parsePort(String[] args, int port) throws NumberFormatException {
        if (args.length > 0) {
            try {
                port = Integer.parseInt(args[0].replaceAll("\\p{Punct}", ""));
            } catch (NumberFormatException e) {
                e.printStackTrace();
            }
        }
        return port;
    }

    private static boolean parsePutty(String[] args, boolean isPutty) {
        if (args.length > 1) {
            return args[1].replaceAll("\\p{Punct}", "")
                    .toLowerCase()
                    .contains("putty");
        }
        return isPutty;
    }

    // runners
    public static void runTelnet(String host, int port) throws IOException {
        Runtime.getRuntime().exec("cmd /c start telnet " + host + " " + port);
    }

    private static void runPutty(String host, int port) throws IOException {
        Runtime.getRuntime().exec("./_lib/putty -raw " + host + " " + port);
    }

    public static void run(int port, boolean isPutty) throws IOException {
        String cp = "out/production/java_nio";
        String name = "nio2.network.UDPMultiCastChat";
        String putty = isPutty ? "putty" : "telnet";
        String cmd = String.format("cmd /c start java -cp %s %s %d %s", cp, name, port, putty);
        Runtime.getRuntime().exec(cmd);
    }

    private static void registerDatagramChannel(Selector selector, String host, int port) throws IOException
{
        DatagramChannel dc = DatagramChannel.open();
        dc.bind(new InetSocketAddress(host, port)); // listening port
        dc.configureBlocking(false);
        dc.register(selector, SelectionKey.OP_READ);

    }
```

- Пример. реализация UDPMultiCastChat **java8_nio/nio2/network/UDPMultiCastChat**

```
    public static void main(String[] args) {
//        args = new String[]{"" + PORT};

        System.out.printf("udp server started...%n");
        Selector selector = null;
        ByteBuffer b = ByteBuffer.allocate(1024);
        SocketAddress socketAddress = null;
        DatagramChannel dcGroup = null;
        DatagramChannel dcKey = null;
        NetworkInterface ni = null;


        int port = parsePort(args, -1);
        boolean isPutty = parsePutty(args, false);

        if (port < 0) {
            System.out.println("Usage: UDPClient 1-10");
            port = 9;
        }
        final int socketPort = SOCKET_MASK + port;
        final int udpPort = UDP_MASK + port;
        final String groupAddr = GROUP_MASK + port;
        final Charset consoleCharset = isPutty ? UTF_CHARSET : TELNET_CHARSET;
        final Charset groupCharset = UTF_CHARSET;

        try {
            ExecutorService exec = Executors.newCachedThreadPool();

            if (isPutty) runPutty(HOST, socketPort);
            else runTelnet(HOST, socketPort);

            InputServer inputServer = new InputServer(socketPort, udpPort, groupAddr);
            exec.execute(inputServer);  // for putty
            exec.shutdown();
// selector
            selector = Selector.open();
// udp server
            ni = NetworkInterface.getByInetAddress(InetAddress.getByName(HOST));
            dcGroup = DatagramChannel.open(StandardProtocolFamily.INET)
                    .setOption(StandardSocketOptions.SO_REUSEADDR, true)
                    .bind(new InetSocketAddress(GROUP_PORT))
                    .setOption(StandardSocketOptions.IP_MULTICAST_IF, ni);
            for (int i = 0; i < 10; i++) {
                String s = GROUP_MASK + i;
                if (s.equals(groupAddr)) continue;
                dcGroup.join(InetAddress.getByName(s), ni);  // группа ввода
            }
            dcGroup.configureBlocking(false)
                    .register(selector, SelectionKey.OP_READ, "group");

            dcKey = DatagramChannel.open(StandardProtocolFamily.INET)
                    .bind(new InetSocketAddress(udpPort));
            dcKey.configureBlocking(false)
                    .register(selector, SelectionKey.OP_READ, "udp");

            InetAddress group = InetAddress.getByName(groupAddr); // группа вывода
```

- Пример. реализация UDPMultiCastChat java8_nio/nio2/network/UDPMultiCastChat

```
            while (true) {
                if (exec.isTerminated()) {
                    break;
                }
                int n = selector.select(100);
                if (n == 0) continue;
                Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                while (it.hasNext()) {
                    SelectionKey key = it.next();
                    it.remove();
                    if (key.isReadable()) {
                        DatagramChannel dc = (DatagramChannel) key.channel();
                        b.clear();
                        socketAddress = dc.receive(b); // accumulate in buffer
                        if (socketAddress == null) continue;

                        b.flip();
                        if (dc.equals(dcGroup)) {
                            String s = new String(b.array(), 0, b.limit(), groupCharset);
                            System.out.print(s);
                            b.clear().put(s.getBytes(consoleCharset)).flip();
                            dcKey.send(b, inputServer.getUdpSocketAddress());          // UDP: to socket
                        } else {
                            String s = new String(b.array(), 0, b.limit(),consoleCharset);
                            b.clear().put(s.getBytes(groupCharset)).flip();
                            dcGroup.send(b, new InetSocketAddress(group, GROUP_PORT)); // KEY: to group
                        }
                        b.clear();

                    }
                }
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            IOUtils.close(selector);
        }
        System.out.printf("udp server closed...%n");
    }

    private static class InputServer implements Runnable {
        private BufferedReader br;
        private PrintWriter pw;
        private int port;
        private int udpPort;
        private SocketAddress socketAddress;
        private byte[] header;

        public InputServer(int port, int udpPort, String groupAddr) throws IOException {
            this.port = port;
            this.udpPort = udpPort;
            this.socketAddress = null;
            this.header = String.format("%s:", groupAddr).getBytes(UTF_CHARSET);
        }
        synchronized public void setUDPSocketAddress(DatagramChannel dc) throws IOException {
            socketAddress = dc.getLocalAddress();
        }
        synchronized public SocketAddress getUdpSocketAddress() {
            return socketAddress;
        }
        @Override
        public void run() {
// socket
            ServerSocketChannel ssc = null;
            Selector selector = null;
            System.out.printf("socket server at s:%d u:%d started...%n", port, udpPort);
// udp
            ByteBuffer bSC = ByteBuffer.allocate(1024);
            ByteBuffer bDC = ByteBuffer.allocate(1024);
            InetSocketAddress udpAddress = new InetSocketAddress(HOST, udpPort);
            ByteBuffer bF = ByteBuffer.allocate(1024);
```

- Пример. реализация UDPMultiCastChat java8_nio/nio2/network/UDPMultiCastChat

```
            try {
                selector = Selector.open();
                ssc = ServerSocketChannel.open();
                ssc.bind(new InetSocketAddress(HOST, port));
                SocketChannel sc = ssc.accept();
                sc.configureBlocking(false);
                sc.register(selector, SelectionKey.OP_READ | SelectionKey.OP_WRITE, "socket");
                for (String s : PUTTY_WELCOME) {
                    bDC.put(String.format("%s%n", s).getBytes(UTF_CHARSET));
                }
                DatagramChannel dc = DatagramChannel.open();
                dc.connect(udpAddress);
                dc.configureBlocking(false);
                dc.register(selector, SelectionKey.OP_READ | SelectionKey.OP_WRITE, "datagram");
                setUDPSocketAddress(dc);
                while (true) {
                    int n = selector.select(100);
                    if (n == 0) break;
                    Iterator<SelectionKey> it = selector.selectedKeys().iterator();
                    while (it.hasNext()) {
                        SelectionKey key = it.next();
                        it.remove();
                        if (key.isReadable()) {
                            if (key.attachment().equals("socket")) {
                                sc = (SocketChannel) key.channel();
                                if (sc.read(bSC) == -1) {
                                    sc.close();
                                    dc.close();
                                    it.forEachRemaining(SelectionKey::cancel);
                                }
                            } else {
                                dc = (DatagramChannel) key.channel();
                                if (dc.read(bDC) == -1) {
                                    dc.close();
                                }
                            }
                        } else if (key.isWritable()) {
                            if (key.attachment().equals("socket")) {
                                sc = (SocketChannel) key.channel();
                                bDC.flip();
                                while (bDC.hasRemaining()) {
                                    sc.write(bDC);
                                }
                                bDC.clear();
                            } else {
                                dc = (DatagramChannel) key.channel();
                                byte[] b = bSC.array();
                                boolean isFound = false;
                                for (int i = 0; i < bSC.position(); i++) {
                                    if (b[i] == '\r' || b[i] == '\n') {
                                        isFound = true;
                                        break;
                                    }
                                }
                                if (!isFound) continue;
                                bSC.flip();
                                bF.clear().put(header).put(bSC).flip();
                                while (bF.hasRemaining()) {
                                    SocketAddress socketAddress = dc.getRemoteAddress();
                                    dc.send(bF, socketAddress);
                                }
                                bSC.clear();
                            }
                        }
                    }
                }

            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                IOUtils.close(ssc, selector);
            }
            System.out.printf("socket server at s:%d u:%d closed...%n", port, udpPort);
        }
    }
}
```

Java Net

Java Net

- Java Net
    - свойства
    - позволяет
- Пример. реализация
-

Java Net Classes

- Java Net Interfaces
    - ContentHandlerFactory
    - FileNameMap
    - ProtocolFamily
    - SocketImplFactory
- Java Net Classes
    - Authenticator The class Authenticator represents an object that knows how to obtain
    - CacheRequest Represents channels for storing resources in the ResponseCache.
    - CacheResponse Represent channels for retrieving resources from the ResponseCache.
    - ContentHandler The abstract class `ContentHandler` is the superclass of all
    - classes that read an `Object` from a `URLConnection`
    - CookieHandler A CookieHandler object provides a callback mechanism to hook up a HTTP state management policy implementation into the HTTP protocol handler
    - CookieManager CookieManager provides a concrete implementation of `CookieHandler`, which separates the storage of cookies from the policy surrounding accepting and rejecting cookies.
    - DatagramPacket This class represents a datagram packet.
    - DatagramSocket This class represents a socket for sending and receiving datagram packets.
    - DatagramSocketImpl Abstract datagram and multicast socket implementation base class.
    - HttpCookie An HttpCookie object represents an HTTP cookie, which carries state information between server and user agent.
    - HttpURLConnection A URLConnection with support for HTTP-specific features.
    - IDN Provides methods to convert internationalized domain names (IDNs) between a normal Unicode representation and an ASCII Compatible Encoding (ACE) representation.
    - Inet4Address This class represents an Internet Protocol version 4 (IPv4) address.
    - Inet6Address This class represents an Internet Protocol version 6 (IPv6) address.
    - InetAddress This class represents an Internet Protocol (IP) address.
    - InetSocketAddress This class implements an IP Socket Address (IP address + port number) It can also be a pair (hostname + port number), in which case an attempt will be made to resolve the hostname.

- Java Net Classes
  - InterfaceAddress This class represents a Network Interface address.
  - JarURLConnection A URL Connection to a Java ARchive (JAR) file or an entry in a JAR file.
  - MulticastSocket The multicast datagram socket class is useful for sending and receiving IP multicast packets.
  - NetPermission This class is for various network permissions.
  - NetworkInterface This class represents a Network Interface made up of a name, and a list of IP addresses assigned to this interface.
  - PasswordAuthentication The class PasswordAuthentication is a data holder that is used by Authenticator.
  - Proxy This class represents a proxy setting, typically a type (http, socks) and a socket address.
  - ProxySelector Selects the proxy server to use, if any, when connecting to the network resource referenced by a URL.
  - ResponseCache Represents implementations of URLConnection caches.
  - SecureCacheResponse Represents a cache response originally retrieved through secure means, such as TLS.
  - ServerSocket This class implements server sockets.
  - Socket This class implements client sockets (also called just "sockets").
  - SocketAddress This class represents a Socket Address with no protocol attachment.
  - SocketImpl The abstract class `SocketImpl` is a common superclass of all classes that actually implement sockets.
  - SocketPermission This class represents access to a network via sockets.
  - StandardSocketOptions Defines the *standard* socket options.
  - URI Represents a Uniform Resource Identifier (URI) reference.
  - URL Class `URL` represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.
  - URLClassLoader This class loader is used to load classes and resources from a search path of URLs referring to both JAR files and directories.
  - URLConnection The abstract class `URLConnection` is the superclass of all classes that represent a communications link between the application and a URL.
  - URLDecoder
  - Utility class for HTML form decoding.
  - URLEncoder
  - Utility class for HTML form encoding.
  - URLPermission
  - Represents permission to access a resource or set of resources defined by a given url, and for a given set of user-settable request methods and request headers.
  - URLStreamHandler The abstract class `URLStreamHandler` is the common superclass for all stream protocol handlerf
  -
- Пример. реализация **java8_nio/nio/Main05R**

Java Net

- Java Net
    - свойства
    - позволяет
- Пример. реализация **java8_nio/nio/Main05R**
-
-

TRICKS

GIT

- GIT
  - клонировать с github — git clone https://github.com/v777779/jup
  - удалить папку — git rm –r stage_one_temp
  - git commit –m "removed stage_one_temp folder"
  - git push
  - удалить физически — rmdir /s /q stage_one_temp
  - добавить файл — git add stage_one/_exam_code/*
  - git commit -m "added stage_one final code"
  - git push
  - восстановление всего — git checkout *
  - git checkout stage_one/v3
- Если не добавляет файл *.jar
  - значит работает .gitignore — git add * -f
  - форсировать добавление
  -