Input/Output

Adam Krechowicz

File class

```
1  File f = new File("/home/adam/abc.txt");
2  System.out.println(f.exists() ? " file exists " : " file does not exists ");
3  System.out.println(f.canRead() ? "there is right to read file " : "we can not ↩
     read file ");
4  System.out.println(f.canWrite() ? "there is right to write file " : "we can not ↩
     write file ");
5  System.out.println(f.canExecute() ? "there is right to execute file " : "we can ↩
     not execute file ");
6  System.out.println(f.getAbsolutePath());
7  System.out.println(f.getName());
```

File class

```
1     System.out.println(f.isDirectory() ? "is directory" : "is not directory");
2     System.out.println(f.isFile() ? "is normal file" : "is not normal file");
3     System.out.println(f.isHidden() ? "file is hidden" : "file is not hidden");
4     System.out.println(new Date(f.lastModified()));
5     System.out.println(f.length());
6     System.out.println(f.separator);
7     System.out.println(f.pathSeparator);
```

Path to files

```
1    File f = new File("/home/adam/abc.txt");
2    //File  f = new File("C:\katalog\test.txt");
3    File f = new File("C:\\katalog\\test.txt");
```

Operations on files

```
1    File f = new File("/home/adam/test.txt");
2    try {
3      f.createNewFile();
4    } catch (IOException e) {
5      e.printStackTrace();
6    }
```

Operations on files

```
1    boolean executable = true;
2    boolean ownerOnly = false;
3    f.setExecutable(executable, ownerOnly);
4    f.setReadable(false, true);
5    f.setWritable(true, false);
6    f.setLastModified(new Date().getTime());
```

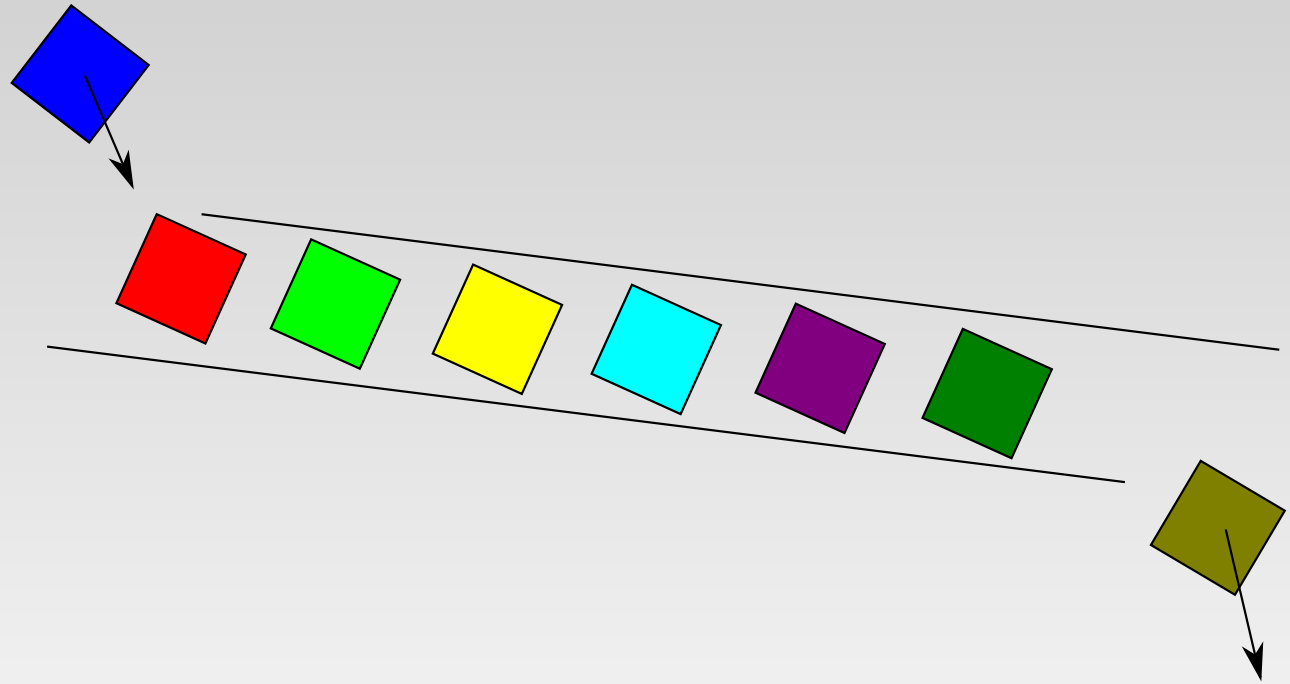
Operations on files

```
1    File f1 = new File("/home/adam/test1.txt");
2    f.renameTo(f1);
3    f1.delete();
4
5    f = new File("/home/adam/katalog");
6    f.mkdir();
```

Directories

```
1     String path = "/home/adam/";
2     File dir = new File(path);
3     File f;
4     for (String s: dir.list()){
5       f = new File(path+s);
6       System.out.println(f.getAbsoluteFile());
7     }
```

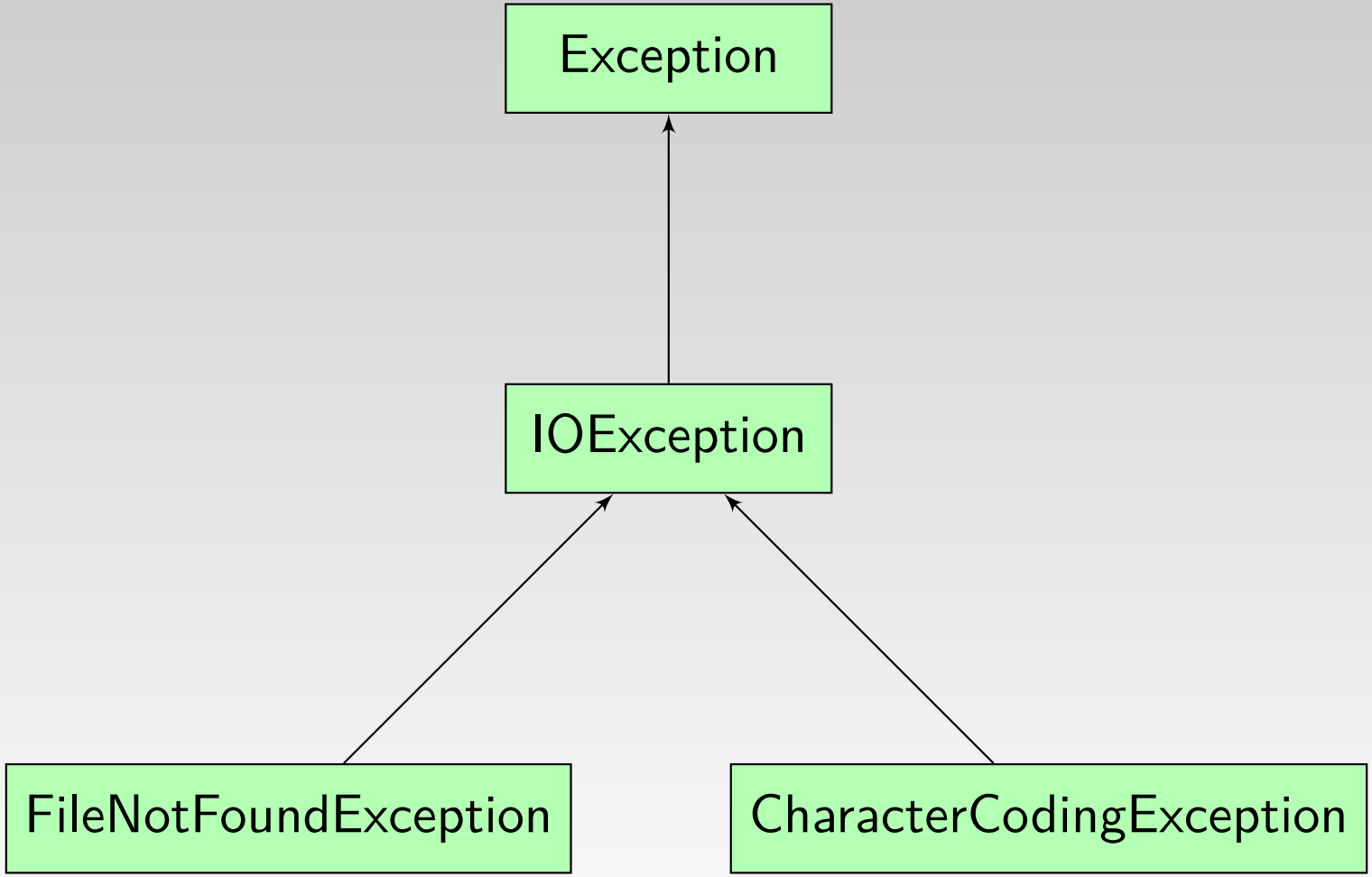
Streams



Writer

```
1   FileWriter fw;
2   int [] buff = {1, 2, 3};
3   try {
4     fw = new FileWriter("/home/adam/abc.txt");
5     fw.write("Hello World");
6     fw.write(97);
7     fw.write(buff);
8     fw.write(buff, 1, 2);
9     fw.write("Hello World", 3, 5);
10    fw.close();
11  } catch (IOException e) {
12    e.printStackTrace();
13  }
```

Exceptios


Exception
IOException
FileNotFoundException
CharacterCodingException

Exceptions

```
1  public static void main(String[] args) {
2    FileWriter fw;
3    try {
4      fw = new FileWriter("/home/adam/abc.txt");
5      if (1==1) throw new IOException();
6      fw.close();
7    } catch (IOException e) {
8      e.printStackTrace();
9    }
10 }
```

Exceptions

```
1  public static void main(String[] args) {
2    FileWriter fw;
3    try {
4      fw = new FileWriter("/home/adam/abc.txt");
5      if (1==1) throw new IOException();
6    } catch (IOException e) {
7      e.printStackTrace();
8    } finally {
9      //fw.close();
10   }
11 }
```

Exceptions

```
1   FileWriter fw;
2   try {
3     fw = new FileWriter("/home/adam/abc.txt");
4     try {
5       if (1 == 1)
6         throw new IOException();
7     } catch (IOException e) {
8       e.printStackTrace();
9     } finally {
10      fw.close();
11    }
12  } catch (Exception e) {
13    e.printStackTrace();
14  }
```

Reader

```
1   FileReader fr;
2   int i;
3   try {
4     fr = new FileReader("/home/adam/abc.txt");
5     try {
6       i = fr.read(); //Typ int
7       while (i != -1) {
8         System.out.print((char)i);
9         i = fr.read();
10      }
11    } catch (IOException e) {
12      e.printStackTrace();
13    } finally {
14      fr.close();
15    }
16  } catch (Exception e) {
17    e.printStackTrace();
18  }
```

Reader

```
1   char[] buffer = new char[512];
2   int result;
3   FileReader fr;
4   try {
5     fr = new FileReader("/home/adam/abc.txt");
6     try {
7       if (fr.ready()) {
8         result = fr.read(buffer, 0, 512);
9         System.out.println(result);
10      }
11    } catch (IOException e) {
12      e.printStackTrace();
13    } finally {
14      fr.close();
15    }
16  } catch (Exception e) {
17    e.printStackTrace();
18  }
```

Reader

```
1   char [] buffer = new char[512];
2   int result;
3   int pos = 0;
4   int size = 512;
5   FileReader fr;
6   try {
7     fr = new FileReader("/home/adam/abc.txt");
8     try {
9       result = fr.read(buffer, 0, size);
10      while (result != -1){
11        result = fr.read(buffer, pos+result, size-result);
12      }
13    } catch (IOException e) {
14      e.printStackTrace();
15    } finally {
16      fr.close();
17    }
18  } catch (Exception e) {
19    e.printStackTrace();
20  }
```

PrintWriter

```
1   PrintWriter pw = null;
2   try {
3     pw = new PrintWriter("/home/adam/abc.txt");
4     int i = 3;
5     pw.print(i);
6     pw.print(new PrintWriterTest());
7     pw.flush();
8     pw.println("Hello World");
9   } catch (IOException e) {
10    e.printStackTrace();
11  } finally {
12    if (pw != null)
13      pw.close();
14  }
```

BufferedReader

```
1   BufferedReader br;
2   String s;
3   try {
4     br = new BufferedReader(new FileReader("/home/adam/abc.txt"));
5     try {
6       br.mark(23);
7       s = br.readLine();
8       System.out.println(s);
9       br.reset();
10      System.out.println(br.readLine());
11      br.reset(); br.skip(2);
12      System.out.println(br.readLine());
13      if (br.readLine() == null)
14        System.out.println("koniec");
15    } finally {
16      br.close();
17    }
18  } catch (IOException ioe) {
19    ioe.printStackTrace();
20  }
```

FileOutputStream

```
1   FileOutputStream fos;
2   try {
3     fos = new FileOutputStream("/home/adam/abc.txt");
4     try {
5       fos.write(97);
6       fos.write("Hello World".getBytes());
7     } catch (IOException e) {
8       e.printStackTrace();
9     } finally {
10      fos.close();
11    }
12  } catch (Exception e) {
13    e.printStackTrace();
14  }
```

FileInputStream

```
1   FileInputStream fis;
2   try {
3     fis = new FileInputStream("/home/adam/abc.txt");
4     try {
5       for (int i = 0; i < fis.available(); i++) {
6         System.out.print((char)fis.read());
7       }
8     } catch (IOException e) {
9       e.printStackTrace();
10    } finally {
11      fis.close();
12    }
13  } catch (Exception e) {
14    e.printStackTrace();
15  }
```

DataOutputStream

```
1   DataOutputStream dos;
2   try {
3     dos = new DataOutputStream(new FileOutputStream("/home/adam/abc.txt"));
4     try {
5       dos.writeInt(10);
6       dos.writeDouble(0.4);
7     } catch (IOException e) {
8       e.printStackTrace();
9     } finally {
10      dos.close();
11    }
12  } catch (Exception e) {
13    e.printStackTrace();
14  }
```

DataInputStream

```
1   DataInputStream dis;
2   try {
3     dis = new DataInputStream(new FileInputStream("/home/adam/abc.txt"));
4     try {
5       System.out.println(dis.readInt());
6       System.out.println(dis.readDouble());
7     } catch (IOException e) {
8       e.printStackTrace();
9     } finally {
10      dis.close();
11    }
12  } catch (Exception e) {
13    e.printStackTrace();
14  }
```

RandomAccessFile

```
1   File f = new File("/home/adam/abc.txt");
2   RandomAccessFile raf;
3   try {
4     raf = new RandomAccessFile(f, "rw");
5     raf.write("Hello World\n".getBytes());
6     System.out.println(raf.getFilePointer());
7     System.out.println(raf.length());
8     raf.seek(0);
9     System.out.println(raf.getFilePointer());
10    System.out.println(raf.readLine());
11  } catch (IOException e) {
12    e.printStackTrace();
13  }
```

Object serialization

```
1  public class SerializableClass implements Serializable {
2    private int poleInteger;
3    private String poleString;
4    public SerializableClass() {
5      super();
6    }
7    public void setPoleInteger(int poleInteger) {
8      this.poleInteger = poleInteger;
9    }
10   public int getPoleInteger() {
11     return poleInteger;
12   }
13   public void setPoleString(String poleString) {
14     this.poleString = poleString;
15   }
16   public String getPoleString() {
17     return poleString;
18   }
19 }
```

ObjectOutputStream

```
1   SerializableClass sc = new SerializableClass();
2   sc.setPoleInteger(4);
3   sc.setPoleString("Hello");
4
5   FileOutputStream fos;
6   ObjectOutputStream oos;
7   try {
8     fos = new FileOutputStream("/home/adam/abc.txt");
9     try {
10      oos = new ObjectOutputStream(fos);
11      oos.writeObject(sc);
12    } catch (IOException e) {
13      e.printStackTrace();
14    } finally {
15      fos.close();
16    }
17  } catch (Exception e) {
18    e.printStackTrace();
19  }
```

ObjectInputStream

```
1   SerializableClass restored;
2   FileInputStream fis;
3   ObjectInputStream ois;
4   try {
5     fis = new FileInputStream("/home/adam/abc.txt");
6     try {
7       ois = new ObjectInputStream(fis);
8       restored = (SerializableClass)ois.readObject();
9       System.out.println(restored.getPoleInteger());
10      System.out.println(restored.getPoleString());
11    } catch (IOException e) {
12      e.printStackTrace();
13    } catch (ClassNotFoundException e) {
14      e.printStackTrace();
15    } finally {
16      fis.close();
17    }
18  } catch (Exception e) {
19    e.printStackTrace();
20  }
```

Transient

```
1  public class HalfSerializableClass implements Serializable {
2
3    private int poleInteger;
4    transient private String poleString;
5
6    public void setPoleInteger(int poleInteger) {
7      this.poleInteger = poleInteger;
8    }
9    public int getPoleInteger() {
10     return poleInteger;
11   }
12   public void setPoleString(String poleString) {
13     this.poleString = poleString;
14   }
15   public String getPoleString() {
16     return poleString;
17   }
18 }
```

Mieszane pliki

```
1   FileOutputStream fos;
2   ObjectOutputStream oos;
3   try {
4     fos = new FileOutputStream("/home/adam/abc.txt");
5     try {
6       oos = new ObjectOutputStream(fos);
7       oos.writeObject(sc);
8       DataOutputStream dos = new DataOutputStream(fos);
9       dos.writeInt(5);;
10    } catch (IOException e) {
11      e.printStackTrace();
12    } finally {
13      fos.close();
14    }
15  } catch (Exception e) {
16    e.printStackTrace();
17  }
```

Multiple content types

```
1   SerializableClass restored;
2   FileInputStream fis;
3   ObjectInputStream ois;
4   try {
5     fis = new FileInputStream("/home/adam/abc.txt");
6     try {
7       ois = new ObjectInputStream(fis);
8       restored = (SerializableClass)ois.readObject();
9       System.out.println(restored.getPoleInteger());
10      System.out.println(restored.getPoleString());
11      DataInputStream dis = new DataInputStream(fis);
12      System.out.println(dis.readInt());
13    } catch (Exception e) {
14      e.printStackTrace();
15    } finally {
16      fis.close();
17    }
18  } catch (Exception e) {
19    e.printStackTrace();
20  }
```

Stream closing

```
1       fis = new FileInputStream("/home/adam/abc.txt");
2       try {
3         ois = new ObjectInputStream(fis);
4         restored = (SerializableClass)ois.readObject();
5         System.out.println(restored.getPoleInteger());
6         System.out.println(restored.getPoleString());
7         DataInputStream dis = new DataInputStream(fis);
8         System.out.println(dis.readInt());
9         dis.close();
10        ois.close();
11        fis.close();
```

ByteArrayOutputStream

```
1   ByteArrayOutputStream baos = new ByteArrayOutputStream();
2
3   try {
4     try {
5       baos.write("Hello World".getBytes());
6     } catch (IOException e) {
7       e.printStackTrace();
8     } finally {
9       baos.close();
10    }
11  } catch (Exception e) {
12    e.printStackTrace();
13  }
14
15  byte[] buffer = baos.toByteArray();
16
17  for (int i = 0; i < buffer.length; i++)
18    System.out.print((char)buffer[i]);
```

ByteArrayInputStream

```
1  ByteArrayInputStream bais = new ByteArrayInputStream(buffer);
2  int i;
3  while ((i = bais.read()) != -1)
4    System.out.print((char)i);
5  bais.read();
```

Properties files

.properties

Files with extension .properties

Example

```
value=123
welcome=Hello
```

Properties files

```
1   Properties prop = new Properties();
2   try {
3     prop.load(new FileInputStream("/home/adam/prop.properties"));
4   } catch (IOException e) {
5     e.printStackTrace();
6   }
7   System.out.println(prop.getProperty("welcome"));
8   prop.setProperty("nazwa", "wartosc");
9   try {
10    prop.store(new FileOutputStream("/home/adam/test"), "comments");
11  } catch (IOException e) {
12    e.printStackTrace();
13  }
```

XML

```
<znacznik>

  <test>
    123
  </test>

  <cos atr="abc">
    Hello
  </cos>

  <closed/>

</znacznik>
```

SAX

SAX

SAX – Simple API for XML

DefaultHandler

- startDocument
- endDocument
- startElement
- endElement
- characters

SAX

```
1  public class XmlTest extends DefaultHandler {
2
3    public void startElement(String uri, String localName,String qName,
4                  Attributes attributes) throws SAXException {
5      System.out.println("Odczytano :" + qName);
6    }
7
8    public static void main(String[] args){
9      SAXParserFactory factory = SAXParserFactory.newInstance();
10     SAXParser saxParser;
11     try {
12       saxParser = factory.newSAXParser();
13       saxParser.parse("/home/adam/test.xml", new XmlTest());
14     } catch (Exception e) {
15       e.printStackTrace();
16     }
17   }
18 }
```

DOM

```
1   DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
2   DocumentBuilder db;
3   try {
4     db = dbf.newDocumentBuilder();
5     Document doc = db.parse(new File("/home/adam/test.xml"));
6     Node n = doc.getDocumentElement();
7     n.normalize();
8     System.out.println(n.getNodeName());
9     NodeList nodes = n.getChildNodes();
10    for (int i = 0; i < nodes.getLength(); i++){
11      System.out.println(nodes.item(i).getNodeName());
12    }
13  } catch (Exception e) {
14    e.printStackTrace();
15  }
```

I/O Redirecting

```
1   PrintStream output;
2   PrintStream error;
3   InputStream input;
4   PrintStream prev = System.out;
5   try {
6     output = new PrintStream("/home/adam/output.txt");
7     error = new PrintStream("/home/adam/error.txt");
8     input = new FileInputStream("/home/adam/input.txt");
9     System.setOut(output);
10    System.setErr(error);
11    System.setIn(input);
12    System.out.println("Hello World");
13    System.err.println("Hello Error");
14    Scanner s = new Scanner(System.in);
15    System.setOut(prev);
16    System.out.println(s.nextLine());
17    throw new FileNotFoundException();
18  } catch (FileNotFoundException e) {
19    e.printStackTrace();
20  }
```

Console class

```
1  public static void main(String[] args) {
2    Console c = System.console();
3    if (c == null) {
4      System.out.println("Please run from console!");
5      System.exit(1);
6    }
7    System.out.println(c.readLine("Wpisz tekst: "));
8    System.out.println(c.readPassword("Wpisz haslo: "));
9  }
```

try-with-resources

```
1  try{
2    try(BufferedReader br =
3        new BufferedReader(new FileReader("/home/adam/abc.txt"))){
4      System.out.println(br.readLine());
5    }
6  } catch (IOException e){
7    e.printStackTrace();
8  }
```

Files

```
1  byte[] buffer;
2  try{
3    buffer = Files.readAllBytes(Paths.get("/home/adam/abc.txt"));
4    System.out.println(new String(buffer));
5    Files.write(Paths.get("/home/adam/nio.txt"), buffer);
6  } catch (IOException e){
7    e.printStackTrace();
8  }
```

NIO – New Input/Output

- Do not use concepts of streams instead of:
  - Channels
  - Buffers
- Better serve in multithreaded programs
- Non blocking operations
- Allows to use selectors

NIO – channels

- Channel may be bidirectional
- Stream have one direction
- Channel is used via buffers
- Channel types
  - FileChannel
  - DatagramChannel
  - SocketChannel
  - ServerSocketChannel

NIO – Buffers

- Classes that wraps memory fragments to serve I/O
- Single buffer may be used to read and write
- Buffer does not need to be prepared
  - flip() – when we want to read after writing
  - clear() – when we still want to read after reading whole buffer
  - compact() – like clear but there is still something to read

Read from channel

```
1  RandomAccessFile raf = new RandomAccessFile("/home/adam/test1", "r");
2  FileChannel channel = raf.getChannel();
3  ByteBuffer bb = ByteBuffer.allocate(64);
4  while (channel.read(bb) != -1) {
5    bb.flip();
6    System.out.println(new String(bb.array()));
7  }
```

Write to channel

```
1 RandomAccessFile raf = new RandomAccessFile("/home/adam/test2", "rw");
2 FileChannel channel = raf.getChannel();
3 ByteBuffer bb = ByteBuffer.allocate(64);
4 bb.put("Hello World".getBytes());
5 bb.flip();
6 channel.write(bb);
```

THE END!

Additional reading:

- https://docs.oracle.com/javase/tutorial/essential/io/streams.html
- https://docs.oracle.com/javase/tutorial/essential/io/fileio.html
- Thinking in Java chapters by Bruce Eckel:
  - I/O