I/O 2

Java Fundamentals

Tõnis Pool

2017, Tallinn

Agenda

1. File Archivers
2. Command Line I/O
3. Logging
4. Process I/O
5. Stopping Processes
6. Value Classes
7. XML APIs
8. JSON APIs
9. Java Serialization

Some Examples

- Github repository http://bit.ly/jf-github
- Checkout the project (or fork)
- Import the **io** or **process** sub project to your IDE

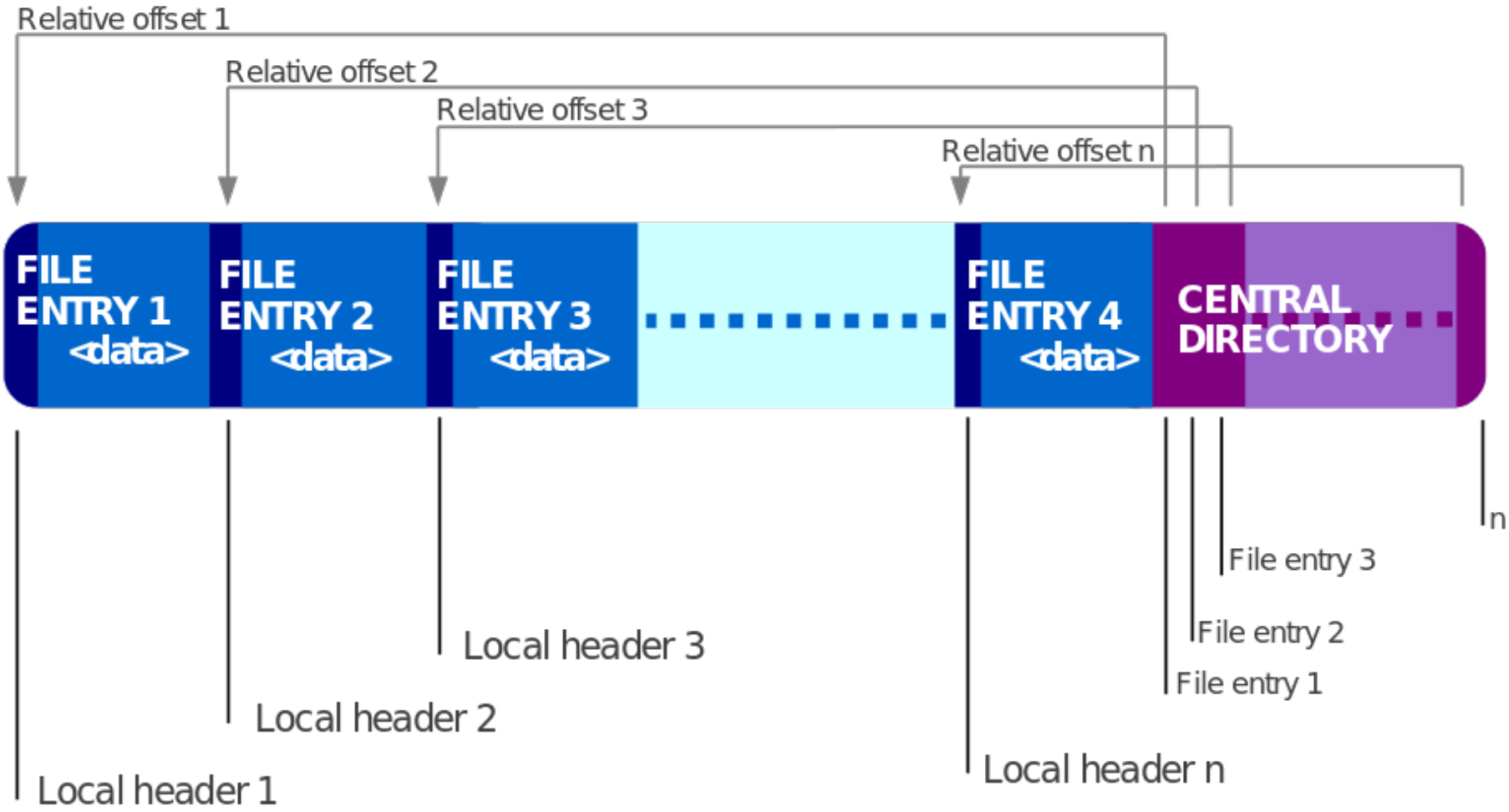
FILE ARCHIVERS

File Archiver

… is a **computer program** that combines a number of **files** together into **one archive** file, or a series of archive files, for easier transportation or storage.

File archivers may employ **lossless data compression** in their **archive formats** to reduce the size of the archive.

ZIP structure



java.util.zip

- **ZipFile** – extracts files from ZIP archive also using the central directory (it sees entire file)
- **ZipInputStream** – extracts files from ZIP archive without the central directory
- **ZipOutputStream** – creates a new ZIP archive
- **ZipEntry** – metadata about a single file in the ZIP archive

Archiving a Directory

```
try (OutputStream out = new
BufferedOutputStream(Files.newOutputStream(dest));
     ZipOutputStream zo = new ZipOutputStream(out);
     Stream<Path> s = Files.walk(src)) {
  s.filter(p -> !p.equals(src))
   .forEach(path -> packEntry(src, zo, path));
}
```

Archiving a Directory Entry

```
void packEntry(Path src, ZipOutputStream zo, Path path)
  throws IOException {
    String name = src.relativize(path)
        .toString().replace('\\', '/');
    boolean isDir = Files.isDirectory(path);
    if (isDir) {
        name += "/";
    }
    log.debug("Packing {}", name);
    ZipEntry e = new ZipEntry(name);
    zo.putNextEntry(e);
    if (!isDir) {
        Files.copy(path, zo);
    }
    zo.closeEntry();
}
```

Extracting an Archive (ZipFile)

```
try (ZipFile zf = new ZipFile(src.toFile())) {
  Enumeration<? extends ZipEntry> en = zf.entries();
  while (en.hasMoreElements()) {
    ZipEntry entry = en.nextElement();
    Path p = dest.resolve(entry.getName());
    Files.createDirectories(p.getParent());
    if (!entry.isDirectory()) {
      InputStream in = zf.getInputStream(entry);
      Files.copy(in, p,
            StandardCopyOption.REPLACE_EXISTING);
    }
  }
}
```

Extracting an Archive (Stream)

```
try (ZipInputStream in = new ZipInputStream(
      new BufferedInputStream(Files.newInputStream(src)))) {
  ZipEntry entry;
  while ((entry = in.getNextEntry()) != null) {
    Path p = dest.resolve(entry.getName());
    Files.createDirectories(p.getParent());
    if (!entry.isDirectory()) {
      Files.copy(in, p,
        StandardCopyOption.REPLACE_EXISTING);
    }
  }
}
```

Apache Commons Compress

https://commons.apache.org/proper/commons-compress/

- Supported archive file formats:
  ar, cpio, Unix dump, tar, zip, gzip, XZ, Pack200, bzip2, 7z, arj, lzma, snappy, DEFLATE and **Z**
- Supports UNIX file permissions for ZIP archives
- API is similar to **java.util.zip**

zt-zip

https://github.com/zeroturnaround/zt-zip

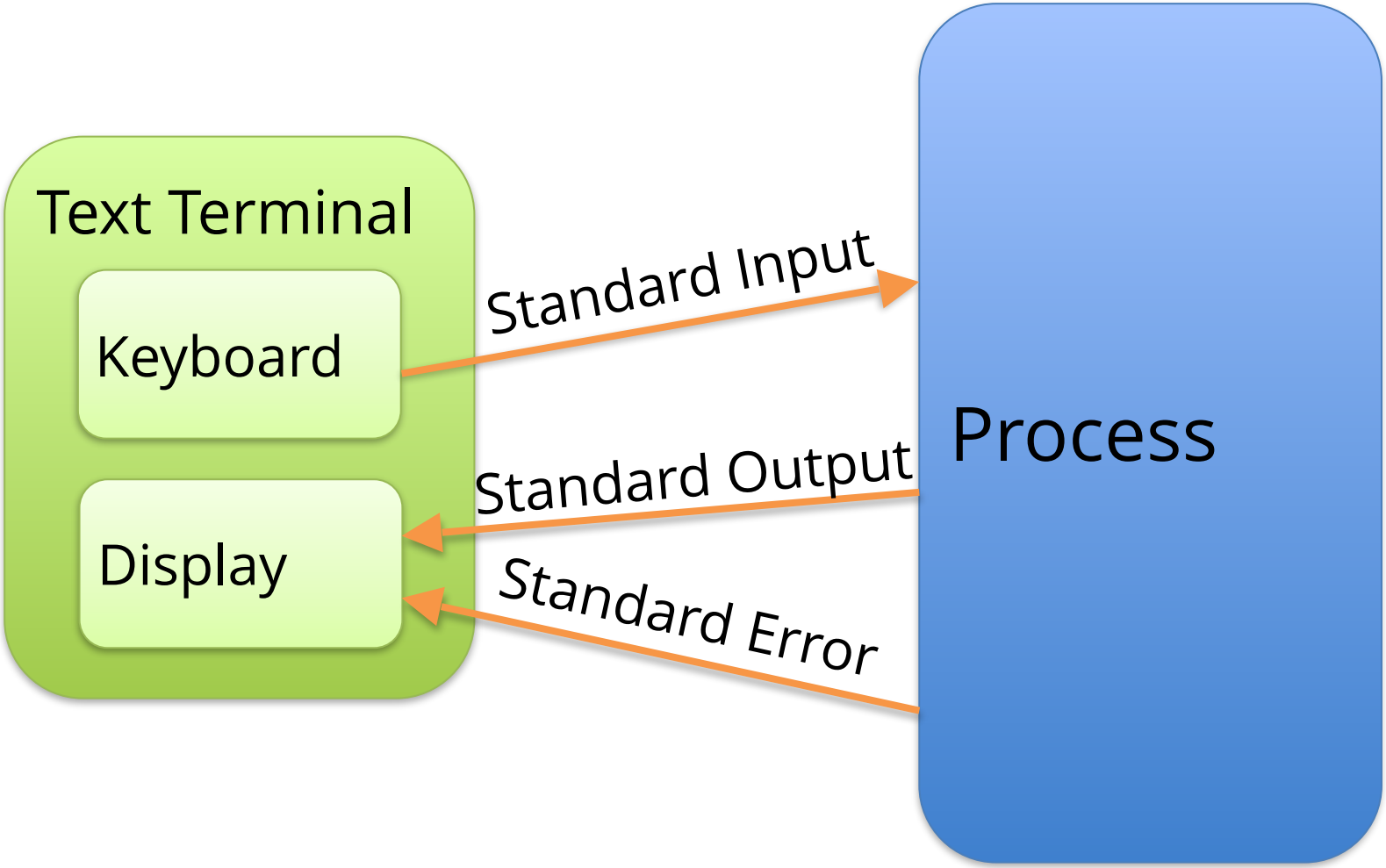
- Shortcut methods for common tasks
- Also supports UNIX file permissions

Examples

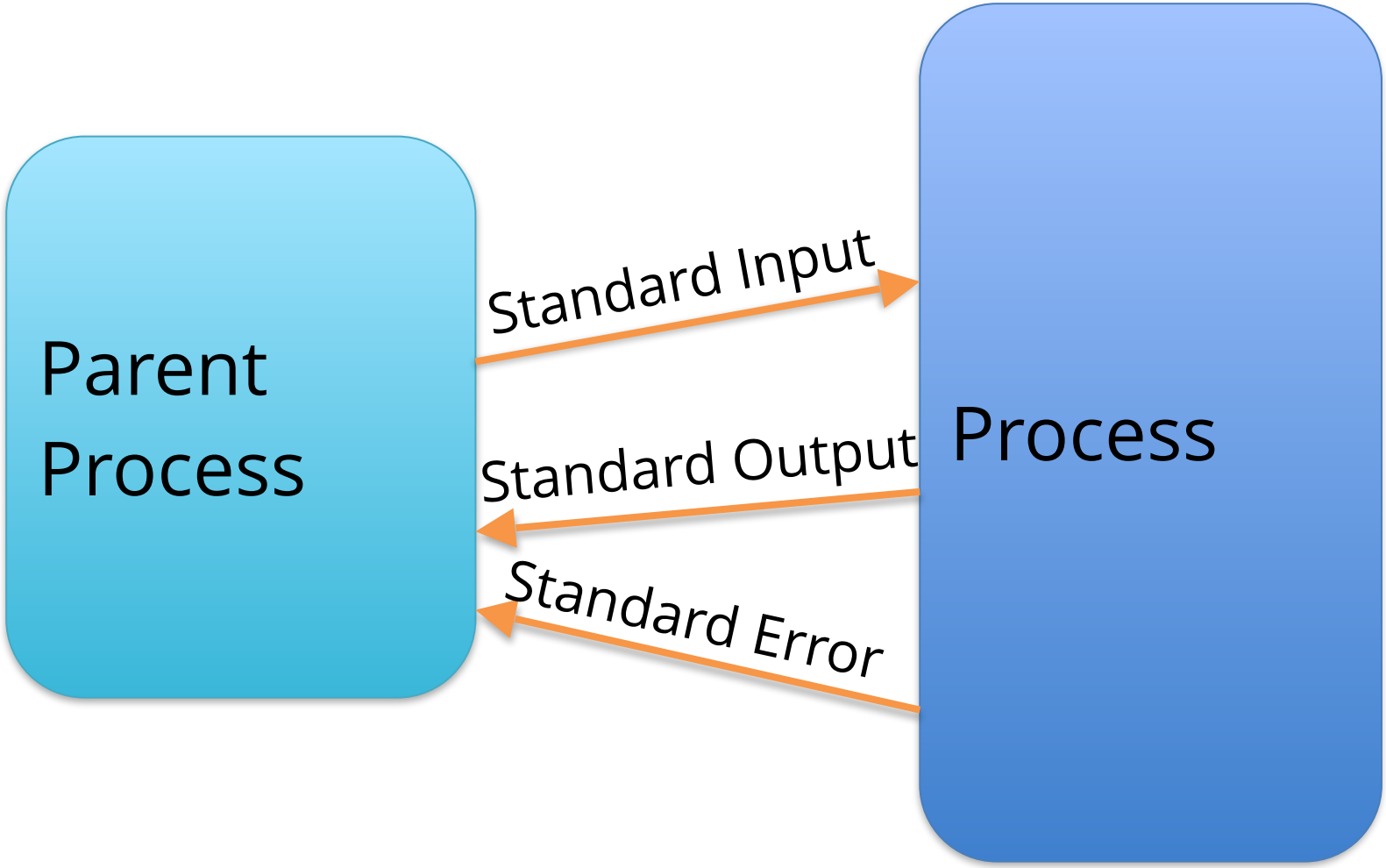
```
// Create an archive
ZipUtil.pack(dir.toFile(), archive.toFile());

// Extract an archive
ZipUtil.unpack(archive.toFile(), dir.toFile());

// Print all entry names
ZipUtil.iterate(archive.toFile(), System.out::println);
```

COMMAND LINE I/O

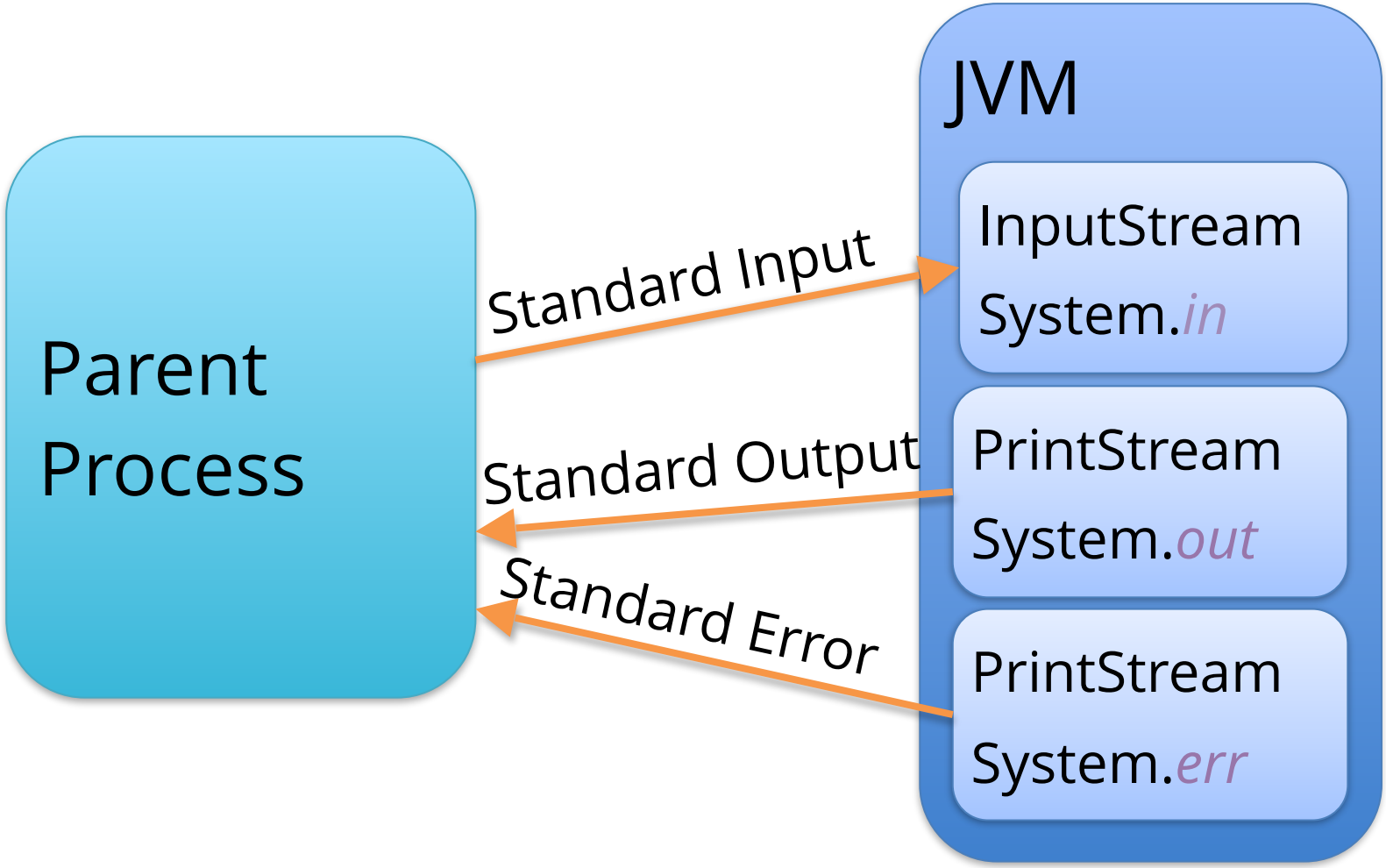
Standard Streams



Standard Streams in General



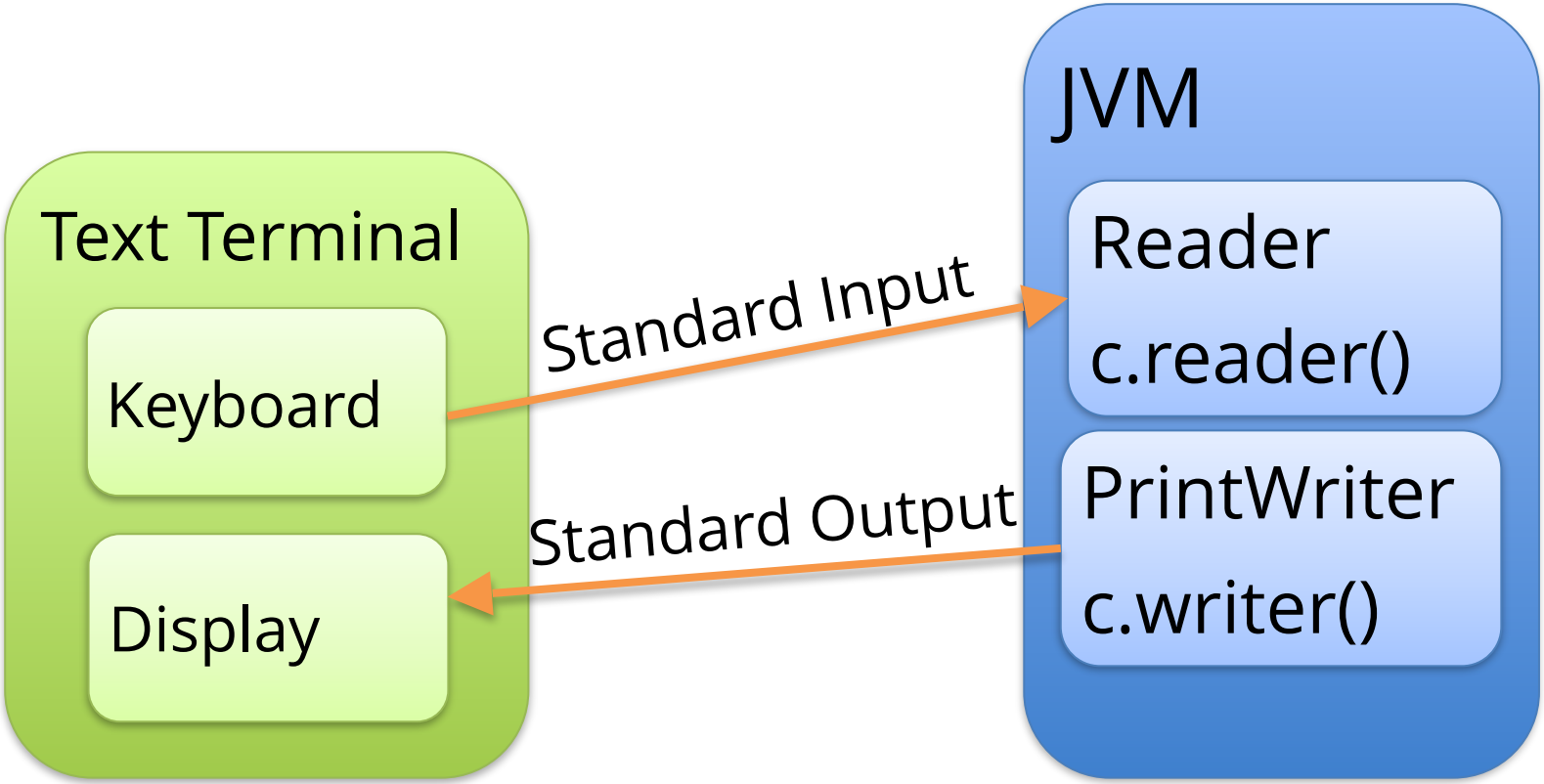
Standard Streams of a JVM



Reading Standard Input

```
BufferedReader in = new BufferedReader(
  new InputStreamReader(System.in));
System.out.println("Enter your name: ");
String name = in.readLine();
System.out.format("Hello, %s!%n", name);
```

Console Streams of a JVM



Console c = System.*console*();

Reading Console Input

```
Console console = System.console();
if (console == null)
  throw new IllegalStateException("No console");
String name = console.readLine("Enter your name: ");
console.format("Hello, %s!%n", name);
```

Reading Password

```
Console console = System.console();
if (console == null)
throw new IllegalStateException("No console.");
char[] expected = {'1','2','3','4','5','6'};
char[] actual = console.readPassword("Enter password: ");
if (Arrays.equals(actual, expected))
  console.format("Permission granted!%n");
else
  console.format("Permission denied!%n");
```

Reading Tokens from Standard Input

```
Scanner scanner = new Scanner(System.in);
int a = scanner.nextInt();
int b = scanner.nextInt();
int s = a + b;
System.out.println(s);
```

Reading Tokens from a File

```
try (Scanner scanner = new Scanner(
    Files.newBufferedReader(
    Paths.get("numbers.txt")))) {
  int a = scanner.nextInt();
  int b = scanner.nextInt();
  int s = a + b;
  System.out.println(s);
}
```

LOGGING

Example

```
public class Hello {
  private static final Logger log =
    LoggerFactory.getLogger(Hello.class);

  public static void main(String[] args) {
    log.info("Hello");
  }
}
```

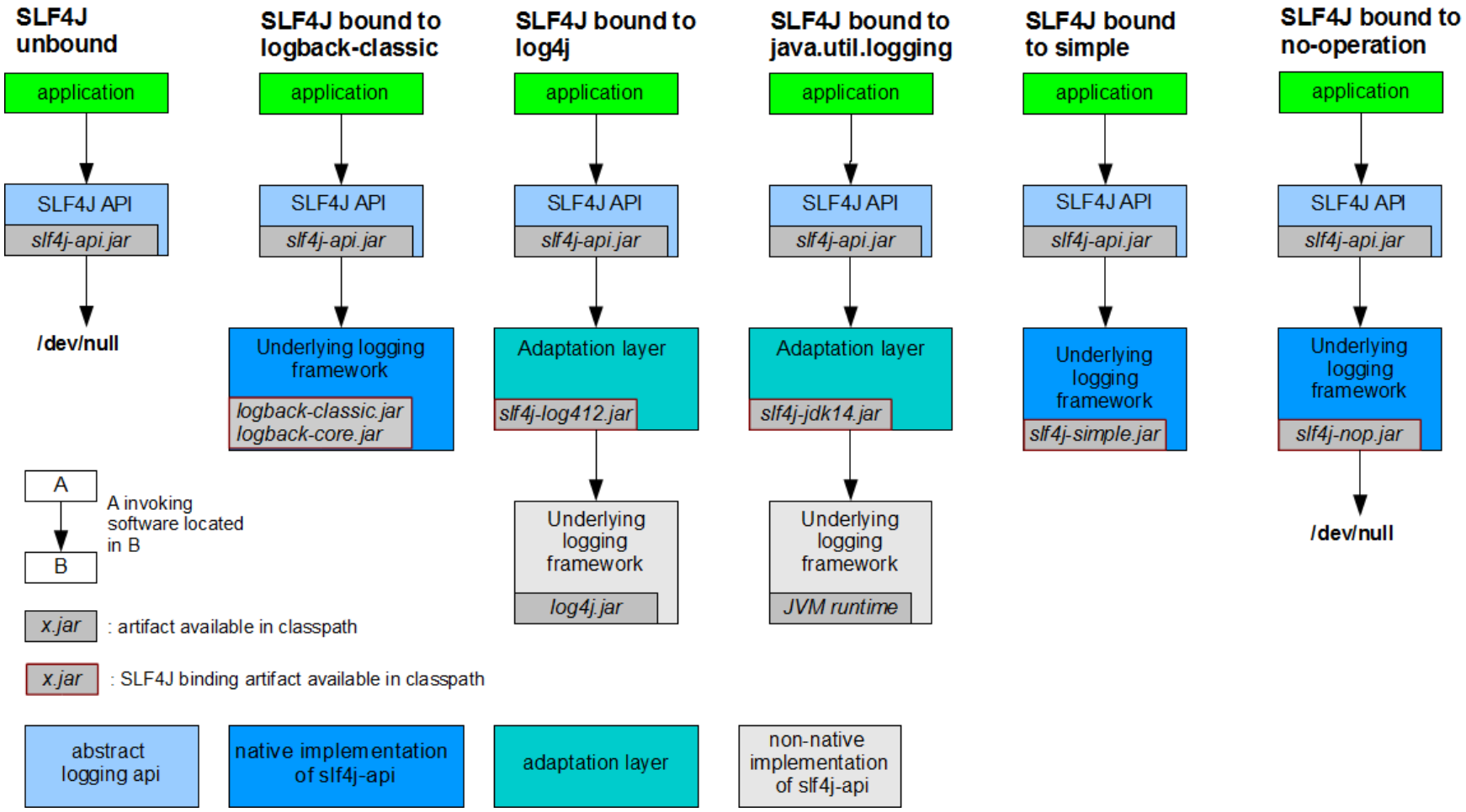
Log Frameworks

- Log4j (2001…2015 discontinued)
- java.util.logging (2002…)
- Apache Commons Logging (2002…)
- Simple Logging Facade for Java (SLF4J) (2006…)
- Logback (2011…)
- Apache Log4j 2 (2012…)

Log Frameworks

- **Log4j (2001…2015 discontinued)**
- java.util.logging (2002…)
- Apache Commons Logging (2002…)
- **Simple Logging Facade for Java (SLF4J) (2006…)**
- **Logback (2011…)**
- Apache Log4j 2 (2012…)

SLF4J


SLF4J unbound
application
SLF4J API
slf4j-api.jar
/dev/null
SLF4J bound to logback-classic
application
SLF4J API
slf4j-api.jar
Underlying logging framework
logback-classic.jar
logback-core.jar
SLF4J bound to log4j
application
SLF4J API
slf4j-api.jar
Adaptation layer
slf4j-log412.jar
Underlying logging framework
log4j.jar
SLF4J bound to java.util.logging
application
SLF4J API
slf4j-api.jar
Adaptation layer
slf4j-jdk14.jar
Underlying logging framework
JVM runtime
SLF4J bound to simple
application
SLF4J API
slf4j-api.jar
Underlying logging framework
slf4j-simple.jar
SLF4J bound to no-operation
application
SLF4J API
slf4j-api.jar
Underlying logging framework
slf4j-nop.jar
/dev/null
A
B
A invoking software located in B
x.jar : artifact available in classpath
x.jar : SLF4J binding artifact available in classpath
abstract logging api
native implementation of slf4j-api
adaptation layer
non-native implementation of slf4j-api

Recommended

- Depend on **SLF4J API**
- Use **Logback** as the logging implementation
- For Maven projects
  - Depend on **logback-classic** in **pom.xml**
  - Add **logback.xml** under src/main/resources
  - Add **logback-test.xml** under src/test/resources

Placeholders

```
Path dir = Paths.get(".");
try (Stream<Path> walk = Files.walk(dir)) {
   List<Path> paths = walk.collect(Collectors.toList());
   log.info("Found {} paths.", paths.size());
   log.trace("All paths " + paths);
   // paths.toString() is always invoked
   log.trace("All paths: {}", paths);
   // paths.toString() is only invoked if needed
}
```

Exceptions

```
boolean contentEquals(Path f1, Path f2) {
  log.debug("Comparing {} and {}", f1, f2);
  try {
    return FileUtils.contentEquals(f1.toFile(),
        f2.toFile());
  }
  catch (IOException e) {
    log.debug("Could not compare {} and {}", f1, f2, e);
    return false;
  }
}
```

PROCESS I/O

Process

- A **process** is an instance of a computer program that is being executed.
- It contains the program code and its current activity.

Process Tools

- Windows GUI: **Task Manager**, **Process Explorer**
- Windows CLI: **tasklist, wmic**
- UNIX GUI: **System Monitor**
- UNIX CLI: **ps, top**
- Mac OS X: **Activity Monitor**
- Java: **jps**

Java Process APIs

- ProcessBuilder
  - Since Java 5
  - Improved in Java 7 & 9
- Runtime.exec()
  - Since Java 1.0
  - Error-prone and inconvenient
  - **Don’t use it**

Starting Process

On Windows:

```
ProcessBuilder builder = new ProcessBuilder(
"C:/Program Files (x86)/Google/Chrome/Application/chrome.exe",
"http://jf.0t.ee");
builder.start();
```

On Mac OS X:

```
ProcessBuilder builder = new ProcessBuilder(
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome",
"http://jf.0t.ee");
builder.start();
```

Opening Browser

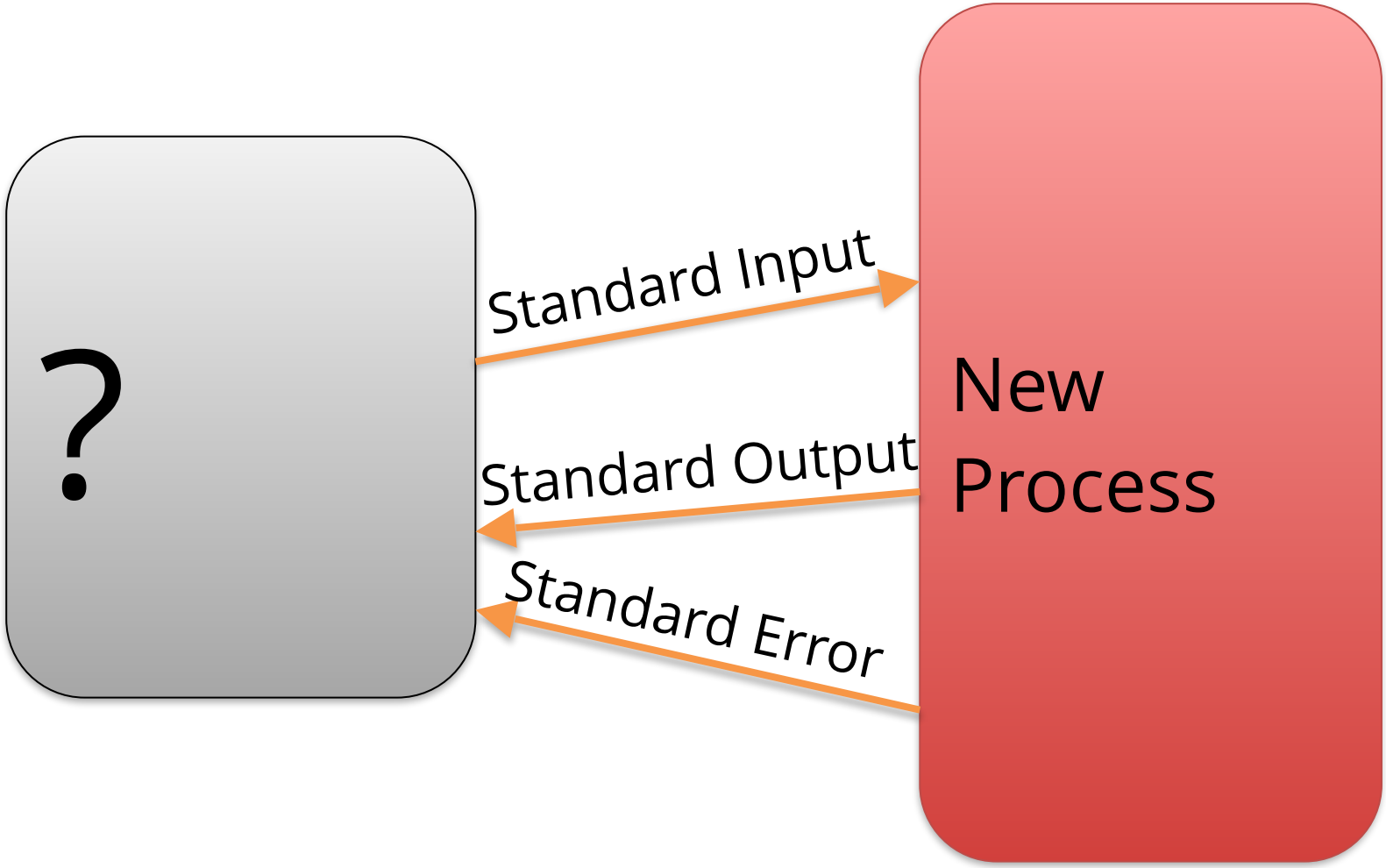
```
Desktop d = Desktop.getDesktop();
// May throw UnsupportedOperationException
// or HeadlessException

if (d.isSupported(Desktop.Action.BROWSE)) {
  URI uri = URI.create("http://jf.0t.ee");
  d.browse(uri); // May throw IOException
}
```

Starting Process with Output

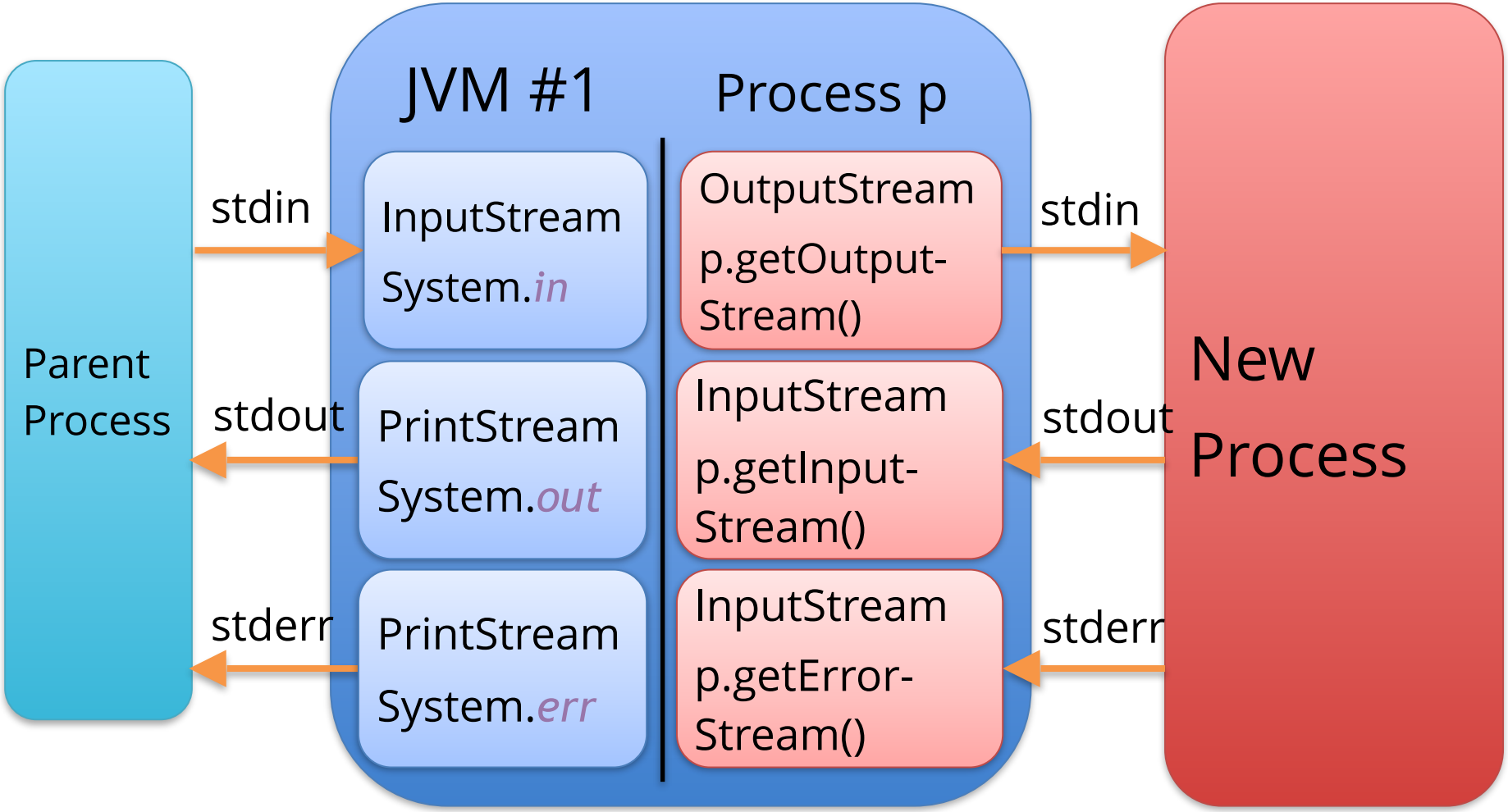
```
ProcessBuilder builder = new ProcessBuilder("ps");
builder.start();
```

Where does the output go?

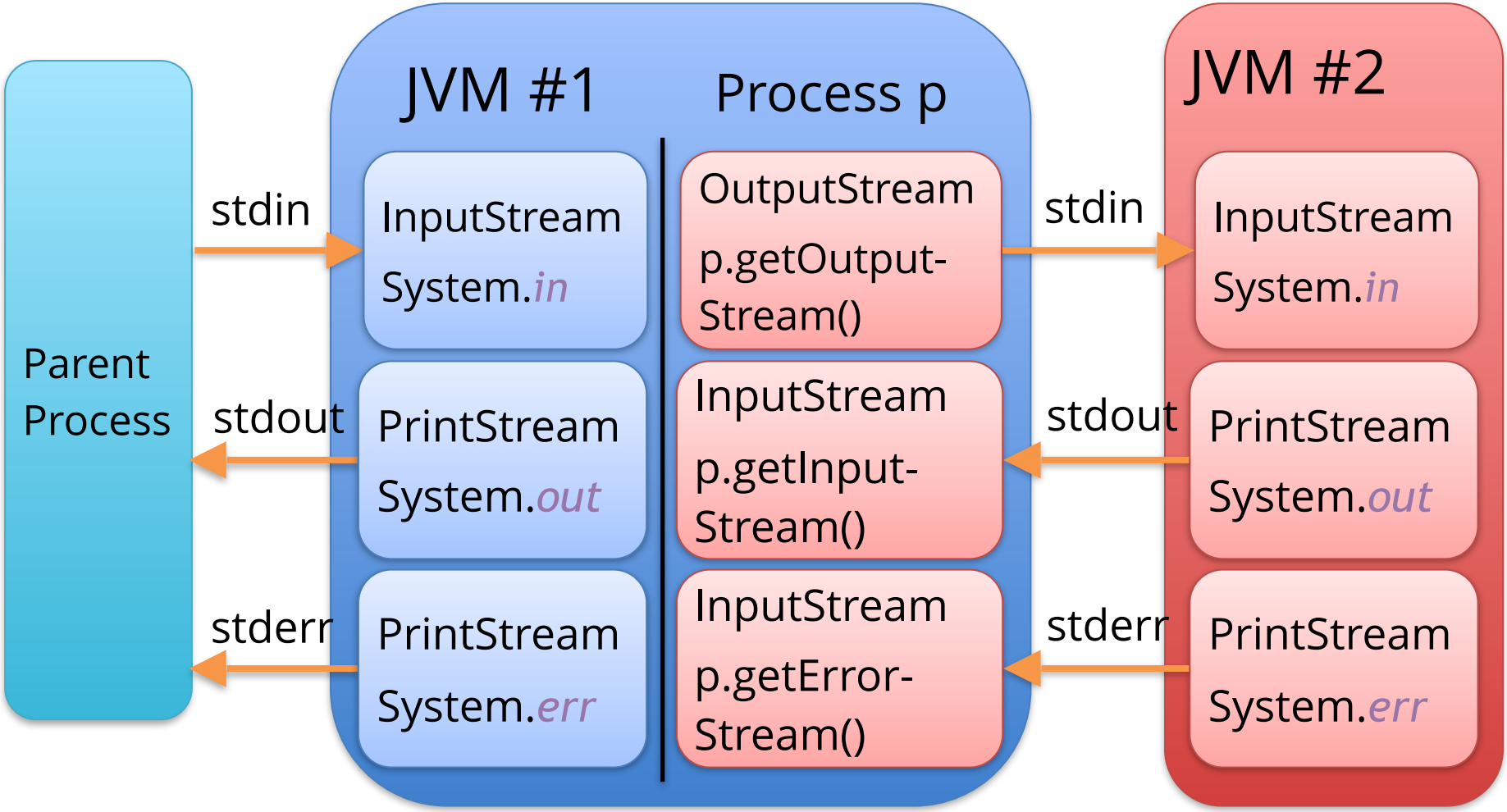
Standard Streams of New Process



No One is Handling Streams By Default



No One is Handling Streams By Default

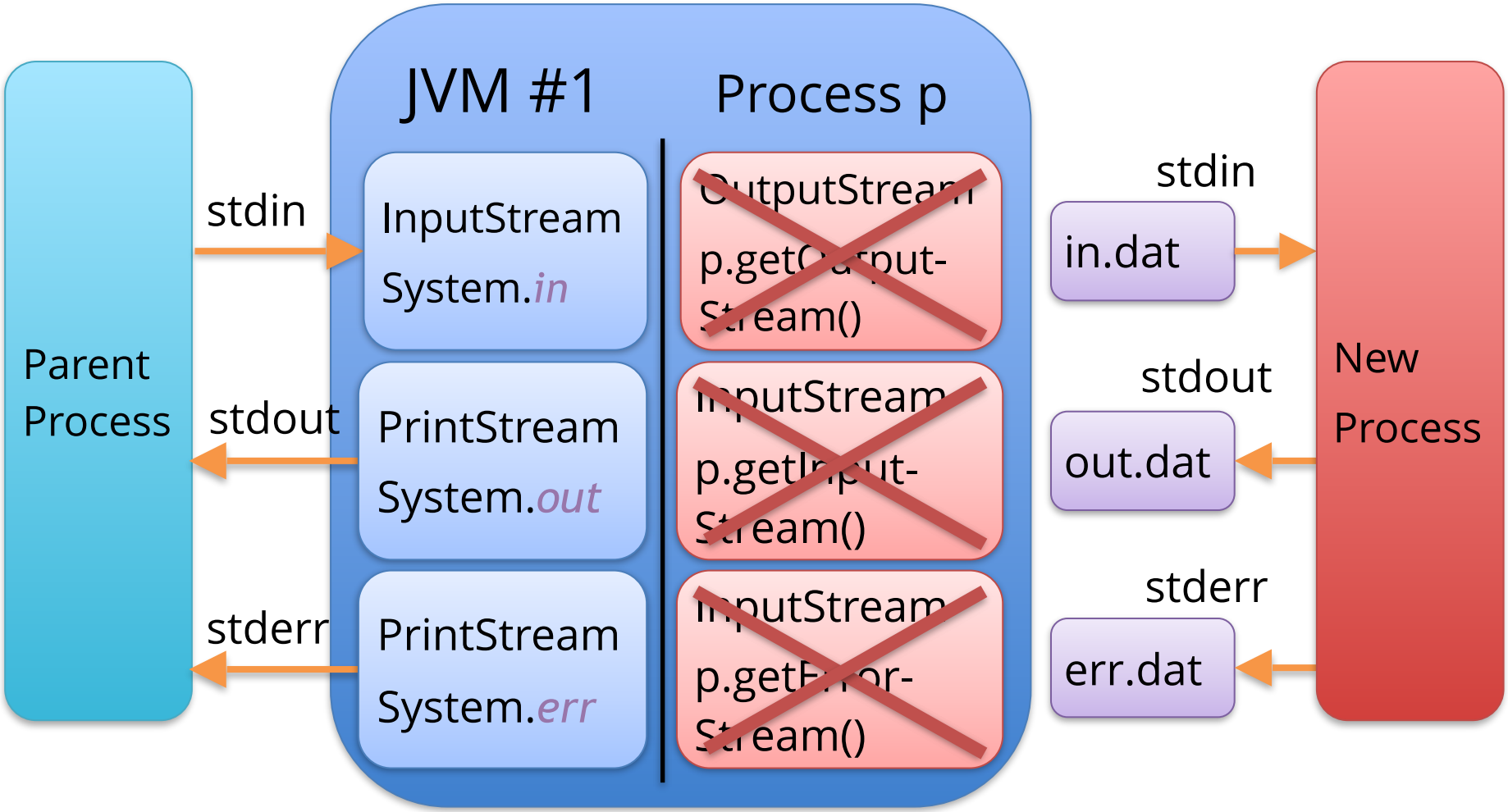


Pipe Buffer

- The standard streams of the new process are **piped** to its parent process
- Each pipe has its own buffer (e.g. 64kb)
- If the pipe is only written into but not read it may become full thus causing the new process to block forever

Redirecting Streams to/from Files

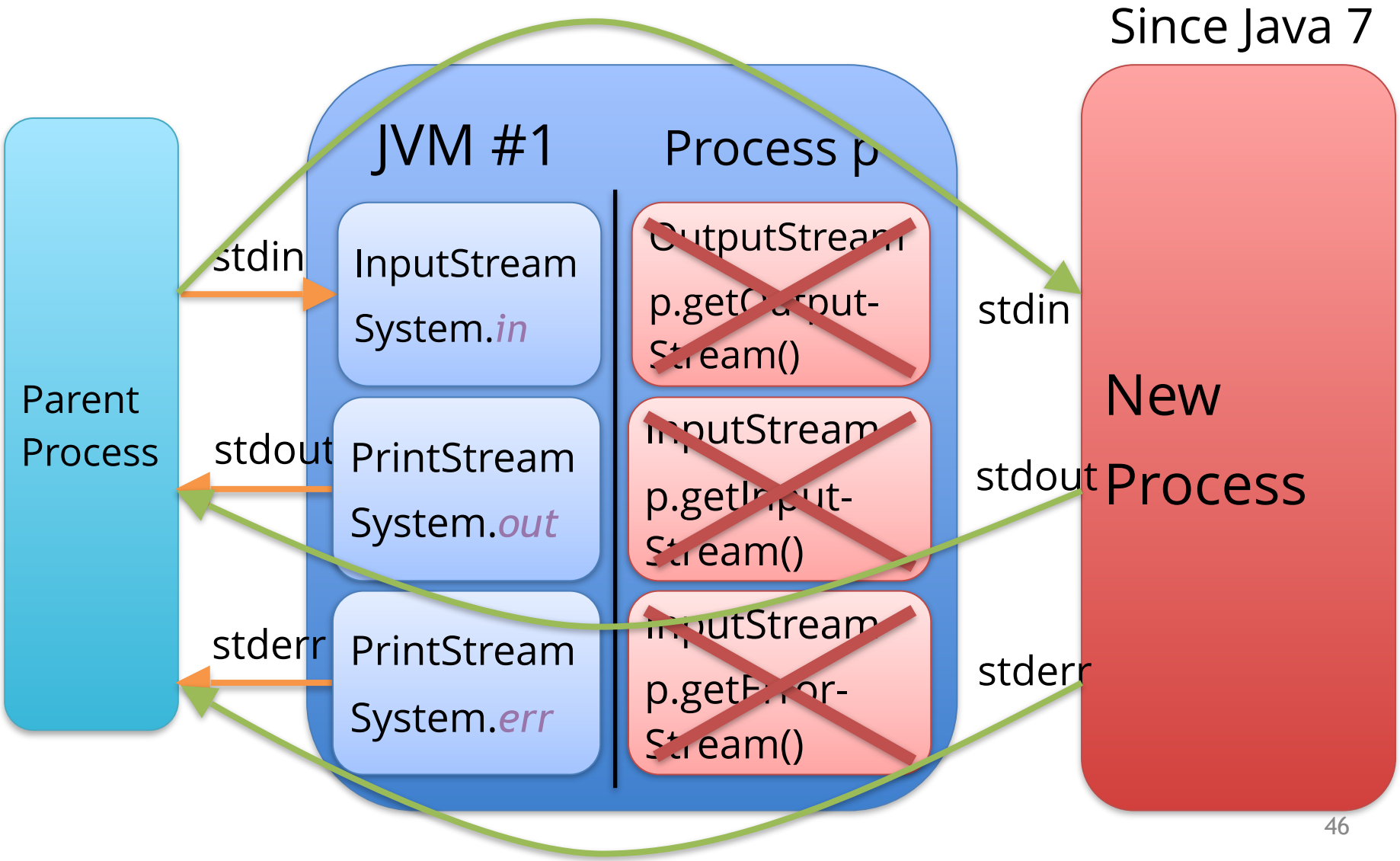
Since Java 7



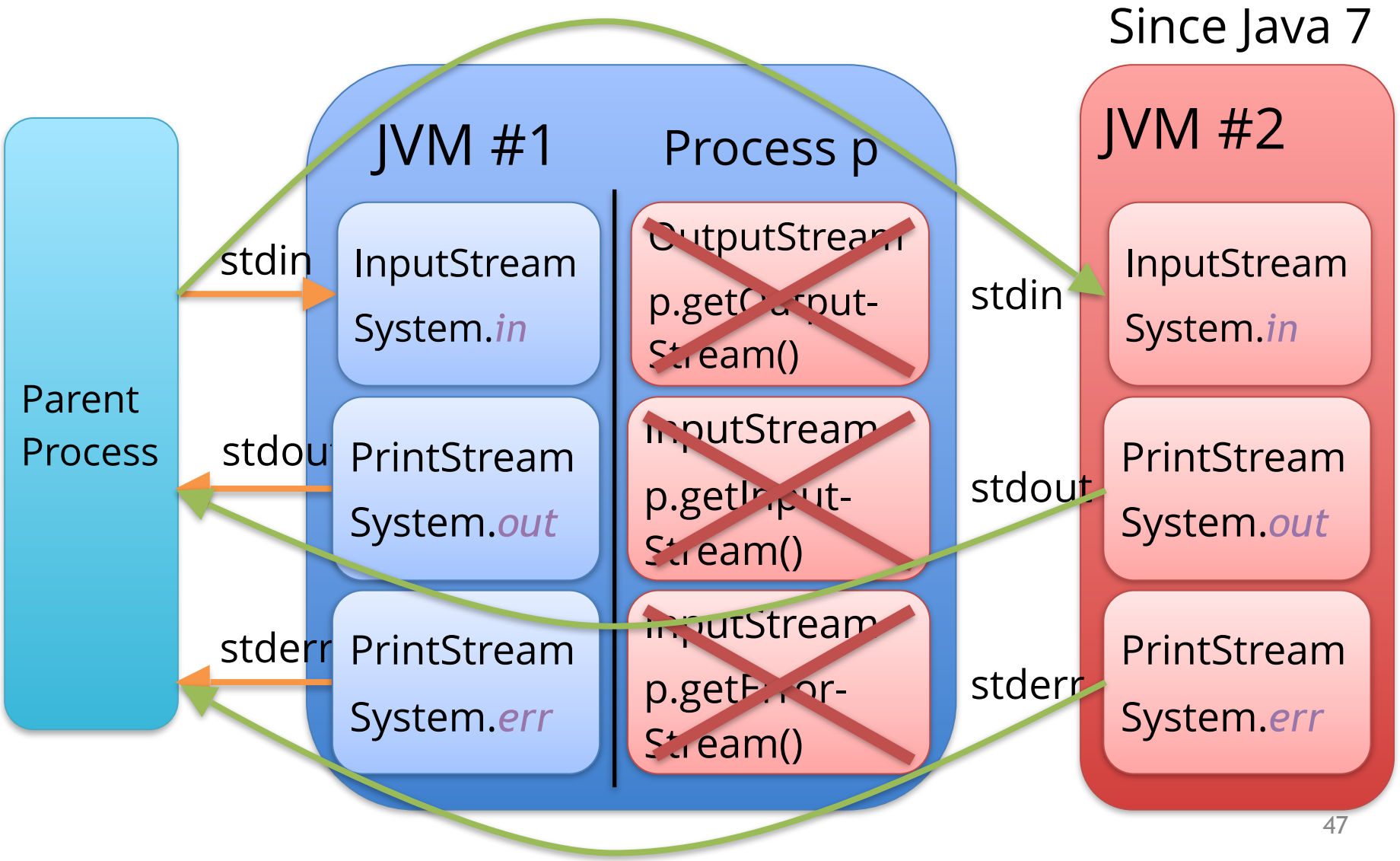
Redirecting Output to File

```
ProcessBuilder pb = new ProcessBuilder("ps");
Path file = Paths.get("out.txt");
pb.redirectOutput(Redirect.to(file.toFile()));
Process process = pb.start();
process.waitFor();
// Process has finished writing the file
try (Stream<String> lines = Files.lines(file)) {
   lines.forEach(System.out::println);
}
```

Inherit Streams



Inherit Streams for New JVM



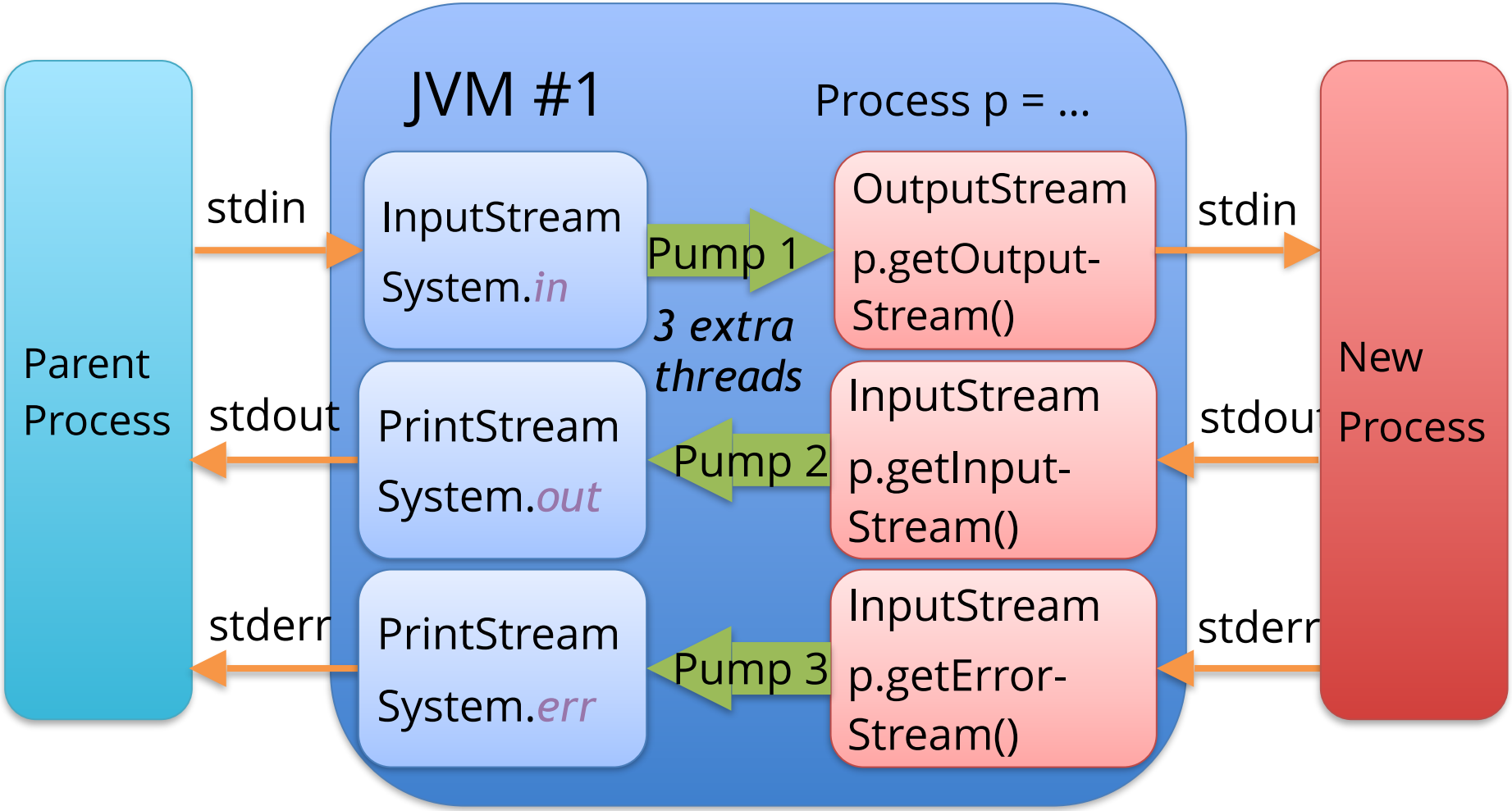
Inherit All Streams

```
ProcessBuilder builder = new ProcessBuilder("head", "-n", "3");
builder.inheritIO();
Process process = builder.start();
process.waitFor();
```

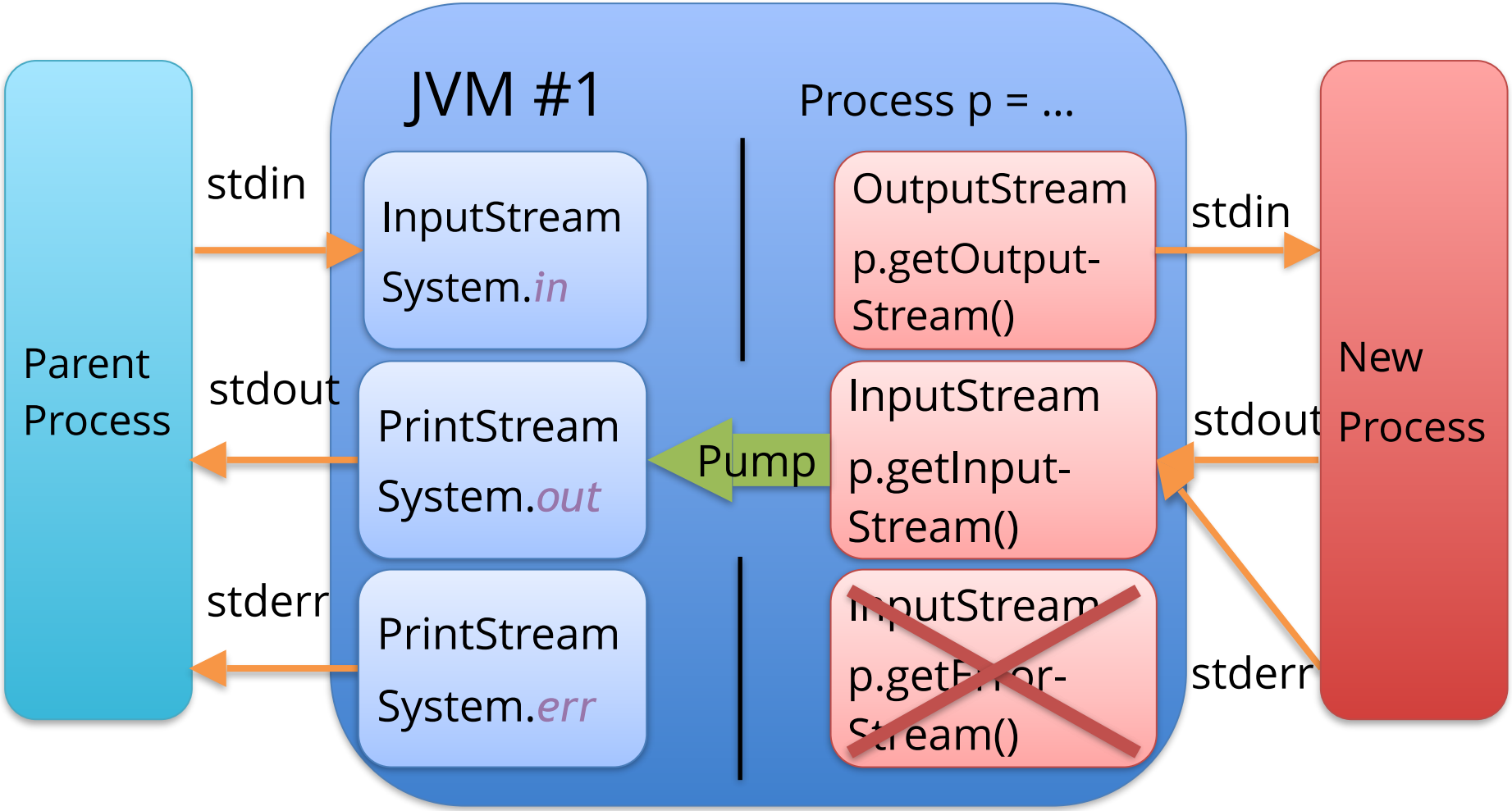
Inherit Only Output Stream

```
ProcessBuilder builder = new ProcessBuilder("head", "-n", "3");
builder.redirectOutput(Redirect.INHERIT);
Process process = builder.start();
process.waitFor();
```

Pumping Streams to/from Parent's



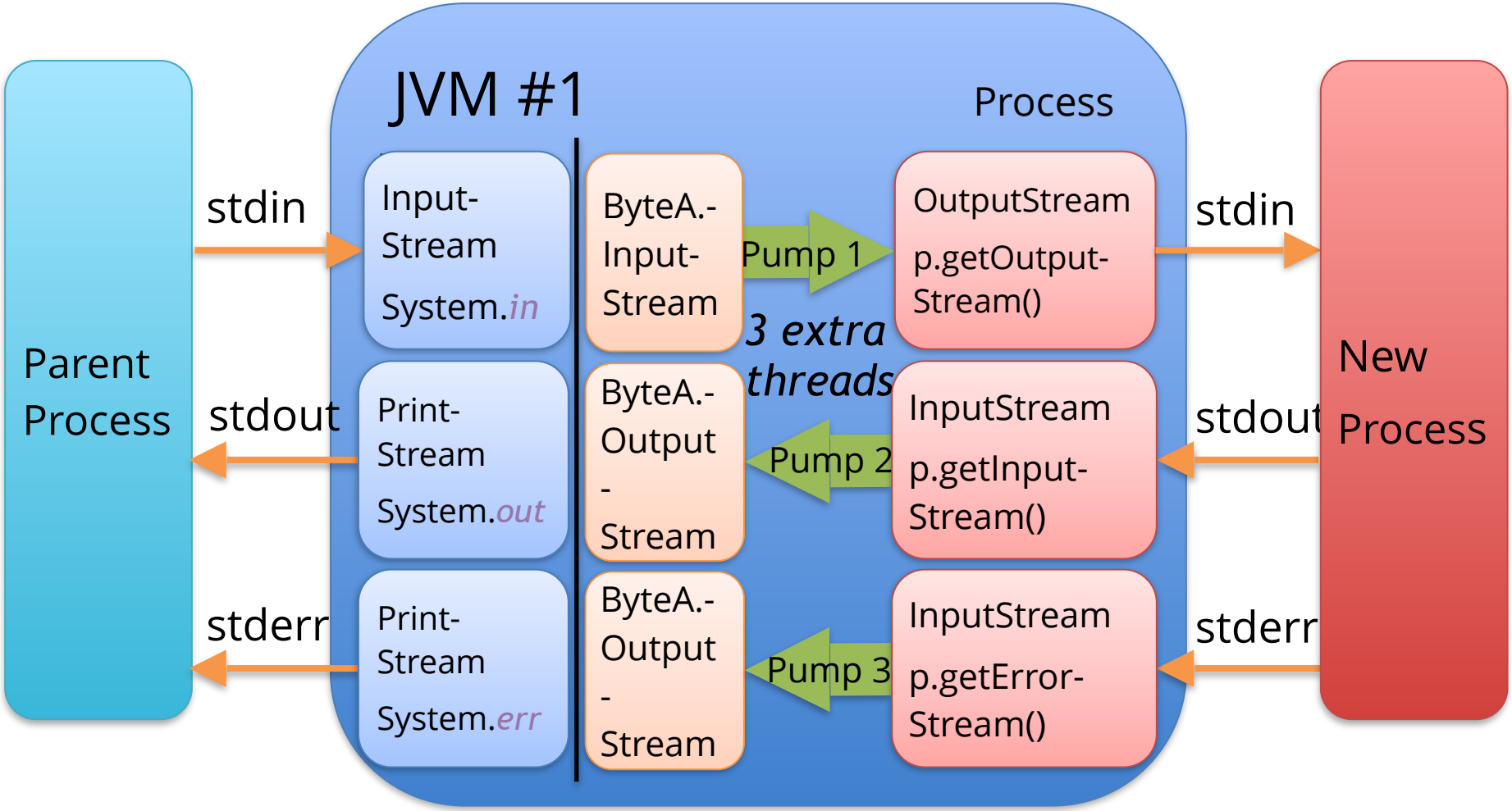
Pumping Output and Error to Parent



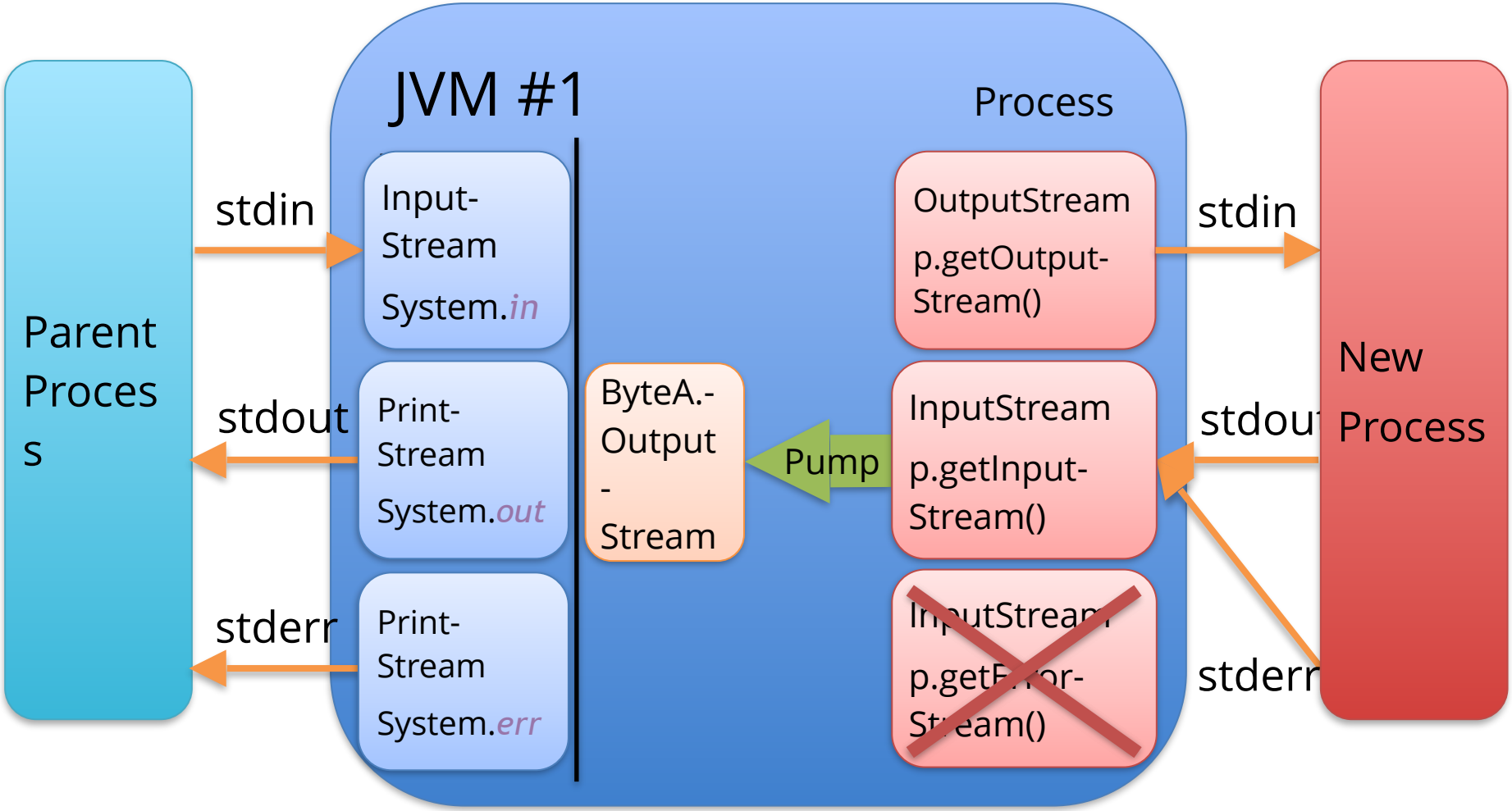
Read Process Output and Error

```
ProcessBuilder pb = new ProcessBuilder("ps");
pb.redirectErrorStream(true);
Process process = pb.start();
InputStream out = process.getInputStream();
IOUtils.copy(out, System.out);
```

Pumping Streams to/from Memory



Pumping Output and Error into Memory



Read Output and Error to Memory

```
ProcessBuilder pb = new ProcessBuilder("ps");
pb.redirectErrorStream(true);
Process process = pb.start();

InputStream stdout = process.getInputStream();
Charset cs = StandardCharsets.UTF_8;
String output = IOUtils.toString(stdout, cs);
System.out.println(output);
```

Setting Working Directory

```
ProcessBuilder builder = new ProcessBuilder("ls");

builder.directory(new File("/"));

builder.redirectOutput(Redirect.INHERIT);
Process process = builder.start();
process.waitFor();
```

Setting Environment

```
ProcessBuilder builder = new ProcessBuilder("env");
builder.inheritIO();

Map<String, String> env = builder.environment();
env.put("foo", "bar");
env.put("PATH", "/myBin" + File.pathSeparator
  + env.get("PATH"));

builder.start();
```

Check Process Exit Status

```
ProcessBuilder pb = new ProcessBuilder(
  "cat", "out.txt");
pb.inheritIO();
Process process = pb.start();
int exit = process.waitFor();
if (exit != 0)
  throw new IllegalStateException("Exit: " + exit);
// 0 - success
```

Stopping a Process

```
ProcessBuilder builder =
  new ProcessBuilder("ping", "ut.ee");
builder.redirectOutput(Redirect.INHERIT);
Process process = builder.start();
try {
  Thread.sleep(1000);
}
finally {
  process.destroy();
}
```

Apache Commons Exec

- http://commons.apache.org/exec/
- Enables to
  - Pump process streams
  - Destroy a process on timeout
  - Expect certain exit values
  - Handle completion or failure using a callback
  - Substitute variables in command line
- JDK 1.3+ compatible

zt-exec

- https://github.com/zeroturnaround/zt-exec
- Even better than commons-exec!
  - Improved API
    - One-liners for common but complex use-cases
    - One-liners for reading process output to string
    - Async processes (java.util.concurrent.Future)
  - Improved stream, timeout and exit code handling
- JDK 1.5+ compatible
- Used by MongoDB, documents4j

Stop Process on Timeout (zt-exec)

```
new ProcessExecutor().command(
    "/Applications/Calculator.app"
        + "/Contents/MacOS/Calculator")
    .timeout(10, TimeUnit.SECONDS).execute();
```

STOPPING PROCESSES

Shutdown Hooks

```
Runtime.getRuntime().addShutdownHook(
    new Thread(() -> System.out.println("JVM shutting down!”)
));
```

- Not **guaranteed** to run
- Can’t be stopped midway
- Run concurrently
- But still useful

(UNIX) Process Signals

Name	Number	Description
HUP	1	Hang up (log out)
INT	2	Interrupt (Ctrl+C)
QUIT	3	Quit (Ctrl+Break)
KILL	9	Non-catchable, non-ignorable kill
TERM	15	Software termination signal

Java Thread Dump Tools

- Ctrl+Break
- **jstack <pid>**
- **kill -QUIT <pid>**
- **sendsignal.exe <pid>**

Stopping a Process

Stopping Gracefully
(process decides what to do)

- Close Window
- Ctrl+C
- **kill <pid> (kill -TERM <pid>)**
- **taskkill /PID <pid>** (may be unsupported)
- Process.**destroy**() on UNIX

Stopping Forcibly
(process has no choice)

- **kill -KILL <pid>**
- **taskkill /F /PID <pid>**
- Process.**destroy**() on Win
- Process.**destroyForcibly**() (since Java 8)

JDK 9 Improvements

- Introduces **`java.lang.ProcessHandle`**
  - `current()`
  - `of(long pid)`
  - `getPid()`
  - `allProcesses()`
- `java.lang.Process#`**`toHandle()`**

Limitations

- Before Java 8
  - There's no guarantee for stopping a process: On UNIX Process.**destroy**() is graceful
- Before Java 9
  - There's no API for stopping processes that are not started by the current JVM (by their PID)
  - There's no API for getting PID of current JVM

zt-process-killer

- https://github.com/zeroturnaround/zt-process-killer
- Abstraction of processes regardless they are started from Java or not (use process ID)
- Methods:
  - Check whether a process is alive
  - Wait until a process has finished
  - Terminate a process gracefully
  - Terminate a process forcibly
  - Get the process ID (PID) of running JVM

Stopping a PID

```
int pid = Integer.parseInt(
  FileUtils.readFileToString(new File("pidfile")));

PidProcess p = Processes.newPidProcess(pid);

ProcessUtil.destroyGracefullyOrForcefullyAndWait(p,
    30, TimeUnit.SECONDS, // graceful timeout
    10, TimeUnit.SECONDS); // forceful timeout
```

VALUE CLASSES

*”Classes should be **immutable** unless there’s a very good reason to make them mutable....If a class cannot be made immutable, **limit its mutability** as much as possible.”*

— Joshua Bloch

Effective Java

Value Classes

```
public final class Movie {
  private final String title;
  private final int year;

  public Movie(String title, int year) {
    this.title = title;
    this.year = year;
  }
}
```

- Compound data structure
- We **only** care about the value of fields not about a specific instance
- Can still contain methods

JDK 8 Value-based Classes

- **Final** and **immutable**
- Implementations of **`equals`**, **`hashCode`**, and **`toString`** which are computed **solely** from the instance's state
- Make no use of identity-sensitive operations such as reference equality (**==**) between instances
- Are considered equal solely based on **equals()**, not based on reference equality (==)
- Are instead instantiated through **factory methods** which make **no committment** as to the identity of returned instances
- **Are freely substitutable**.

https://docs.oracle.com/javase/8/docs/api/java/lang/doc-files/ValueBased.html

Project Valhalla

- ***"Codes like a class, works like an int!"***
- No identity
- Can be flattened
- JDK 10 ?

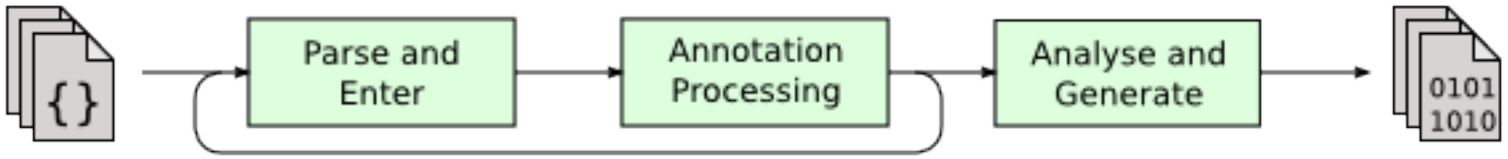
http://openjdk.java.net/jeps/169

No JDK 10?

- AutoValue
  https://github.com/google/auto/blob/master/value/userguide/index.md
- Immutables
  https://immutables.github.io/
- Lombok
  http://jnb.ociweb.com/jnb/jnbJan2010.html

Based on `@Annotation`-s **generate** additional classes to avoid boilerplate

Annotation Processing



AutoValue Example

```
@AutoValue  // concrete class will be generated by AutoValue
abstract class Person {
   /**
    * Create instance of Person.
    */
   static Person create(String lastName, String firstName, long birthYear) {
      return new AutoValue_Person(lastName, firstName, birthYear);
   }

   abstract String lastName();
   abstract String firstName();
   abstract long birthYear();
}
```

AutoValue Example (2)

```
@Generated("com.google.auto.value.processor.AutoValueProcessor")
 final class AutoValue_Person extends Person {

  private final String lastName;
  private final String firstName;
  private final long birthYear;

  AutoValue_Person(
      String lastName,
      String firstName,
      long birthYear) {
    if (lastName == null) {
      throw new NullPointerException("Null lastName");
    }
    this.lastName = lastName;
    if (firstName == null) {
      throw new NullPointerException("Null firstName");
    }
    this.firstName = firstName;
    this.birthYear = birthYear;
  }

  @Override
  String lastName() {
    return lastName;
  }

  @Override
  String firstName() {
    return firstName;
  }

  @Override
  long birthYear() {
    return birthYear;
  }

  @Override
  public String toString() {
    return "Person{"
        + "lastName=" + lastName + ", "
        + "firstName=" + firstName + ", "
        + "birthYear=" + birthYear
        + "}";
  }

  @Override
  public boolean equals(Object o) {
    if (o == this) {
      return true;
    }
    if (o instanceof Person) {
      Person that = (Person) o;
      return (this.lastName.equals(that.lastName()))
           && (this.firstName.equals(that.firstName()))
           && (this.birthYear == that.birthYear());
    }
    return false;
  }

  @Override
  public int hashCode() {
    int h = 1;
    h *= 1000003;
    h ^= this.lastName.hashCode();
    h *= 1000003;
    h ^= this.firstName.hashCode();
    h *= 1000003;
    h ^= (this.birthYear >>> 32) ^ this.birthYear;
    return h;
  }
```

XML APIS

Java Tutorials

https://docs.oracle.com/javase/tutorial/jaxp/

Java API for XML Processing (JAXP)

- Document Object Model (DOM)
  - XPath
- Simple API for XML (SAX)
- Streaming API for XML (StAX)

XML API Comparison

	DOM	SAX	StAX
API Type	In memory tree	Push, streaming	Pull, streaming
Ease of Use	High	Medium	High
CPU and Memory Efficency	Varies	Good	Good
Forward Only	No	Yes	Yes
Read XML	Yes	Yes	Yes
Validate XML	Yes	Yes	No
Write XML	Yes	No	Yes
Create, Read, Update, Delete	Yes	No	No

See https://docs.oracle.com/javase/tutorial/jaxp/stax/why.html

XML Example

```
<?xml version="1.0" ?>
<movies>
  <movie title="Interstellar" year="2014"/>
</movies>
```

Creating XML with DOM

```
DocumentBuilder builder =
    DocumentBuilderFactory.newInstance()
    .newDocumentBuilder();
Document doc = builder.newDocument();
Element movies = doc.createElement("movies");

Element movie = doc.createElement("movie");
movie.setAttribute("title", "Interstellar");
movie.setAttribute("year", "2014");
movies.appendChild(movie);

doc.appendChild(movies);
```

Writing DOM onto Disk

```
Transformer transformer =
    TransformerFactory.newInstance()
    .newTransformer();
DOMSource source = new DOMSource(doc);
StreamResult result = new StreamResult(dest.toFile());
transformer.transform(source, result);
```

Reading DOM from Disk

```
DocumentBuilder builder =
    DocumentBuilderFactory.newInstance()
    .newDocumentBuilder();
Document doc = builder.parse(src.toFile());
```

Reading Values from DOM

```
NodeList movies =
    doc.getElementsByTagName("movie");
for (int i = 0; i < movies.getLength(); i++) {
  Element movie = (Element) movies.item(i);
  String title = movie.getAttribute("title");
  String year = movie.getAttribute("year");
  log.info("Found movie {} ({})", title, year);
}
```

Reading Values via XPath

```
XPath xpath = XPathFactory.newInstance().newXPath();

String title = (String) xpath.compile(“/movies/movie/@title”)
    .evaluate(doc, XPathConstants.STRING);

String year = (String) xpath.compile(“/movies/movie/@year")
    .evaluate(doc, XPathConstants.STRING);

log.info("Found movie {} ({})", title, year);
```

Reading XML with SAX

```
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setNamespaceAware(true);
SAXParser parser = factory.newSAXParser();
parser.parse(new InputSource(src.toUri().toString()),
    new MovieHandler());
```

Handling SAX Events

```
class MovieHandler extends DefaultHandler {
  public void startElement(String uri,
    String localName, String qName,
    Attributes attributes) {
    if (localName.equals("movie")) {
      String title = attributes.getValue("title");
      String year = attributes.getValue("year");
      log.info("Found movie {} ({})", title, year);
    }
  }
}
```

Creating XML via StAX

```
XMLOutputFactory output = XMLOutputFactory.newInstance();
try (OutputStream out =
    new BufferedOutputStream(Files.newOutputStream(dest))) {
  XMLStreamWriter writer =
      output.createXMLStreamWriter(out);
  writer.writeStartDocument();
  writer.writeStartElement("movies");
  writer.writeEmptyElement("movie");
  writer.writeAttribute("title", "Interstellar");
  writer.writeAttribute("year", "2014");
  writer.writeEndElement();
  writer.writeEndDocument();
  writer.close();
}
```

Escaping Characters

- Commons Lang StringEscapeUtils
  - escapeXml10()
  - escapeXml11()
  - unescapeXml()

Reading XML with StAX

```
XMLInputFactory f = XMLInputFactory.newInstance();
try (InputStream in = new
BufferedInputStream(Files.newInputStream(src))) {
  XMLEventReader reader = f.createXMLEventReader(in);
  while (reader.hasNext()) {
    XMLEvent event = reader.nextEvent();
    if (event.isStartElement()) {
      StartElement se = (StartElement) event;
      if (se.getName().getLocalPart().equals("movie")) {
        String title = se.getAttributeByName(
          new QName("title")).getValue();
        String year = se.getAttributeByName(
          new QName("year")).getValue();
        log.info("Found movie {} ({})", title, year);
      }
    }
  }
}
```

XML API Comparison

	DOM	SAX	StAX
API Type	In memory tree	Push, streaming	Pull, streaming
Ease of Use	High	Medium	High
CPU and Memory Efficency	Varies	Good	Good
Forward Only	No	Yes	Yes
Read XML	Yes	Yes	Yes
Validate XML	Yes	Yes	No
Write XML	Yes	No	Yes
Create, Read, Update, Delete	Yes	No	No

See https://docs.oracle.com/javase/tutorial/jaxp/stax/why.html

Java API for XML Binding (JAXB)

- Originally to generate Java classes from XML schema
- Later easy marshaling of Java classes

Creating XML via JAXB

```
List<Movie> movies = new ArrayList<>();
movies.add(new Movie("Interstellar", 2014));
Data data = new Data(movies);

JAXBContext jaxbContext = JAXBContext.newInstance(Data.class);
Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

// output pretty printed
jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

try (Writer writer = Files.newBufferedWriter(dest)) {
    jaxbMarshaller.marshal(data, writer);
}
```

Creating XML via JAXB (2)

```
@XmlRootElement(name = "movies")
public class Data {

  @XmlElement(name = "movie")
  private final List<Movie> movies;

  @SuppressWarnings("unused")
  public Data() {
    this.movies = null;
  }

  public Data(List<Movie> movies) {
    this.movies = Collections.unmodifiableList(movies);
  }
}
```

Creating XML via JAXB (3)

```
@XmlRootElement
public final class Movie {

  @XmlAttribute
  private final String title;

  @XmlAttribute
  private final int year;

  @SuppressWarnings("unused")
  public Movie() {
    this(null, 0);
  }
}
```

JSON APIS

JSON Example

```
{
   "movies":[
      {
         "title":"Interstellar",
         "year":2014
      }
   ]
}
```

JSON APIs

- **org.json:** https://github.com/stleary/JSON-java
- **google-gson:** https://github.com/google/gson
- **Jackson:** https://github.com/FasterXML/jackson

Plain Java Classes

```
class Data {
  List<Movie> movies;
}

class Movie {
  String title;
  int year;
}
```

Writing Object into JSON (GSON)

```
List<Movie> movies = new ArrayList<>();
movies.add(new Movie("Interstellar", 2014));
Data data = new Data(movies);
try (Writer writer = Files.newBufferedWriter(dest)) {
    new Gson().toJson(data, writer);
}
```

Reading Object from JSON (GSON)

```
Data data;
try (Reader reader = Files.newBufferedReader(src)) {
   data = new Gson().fromJson(reader, Data.class);
}

Movie movie = data.getMovies().get(0);
log.info("Found movie {} ({})", movie.getTitle(),
movie.getYear());
```

Writing Object into JSON (Jackson)

```
List<Movie> movies = new ArrayList<>();
movies.add(new Movie("Interstellar", 2014));
Data data = new Data(movies);

ObjectMapper mapper = new ObjectMapper();
mapper.enable(SerializationFeature.INDENT_OUTPUT);
try (Writer writer = Files.newBufferedWriter(dest)) {
   mapper.writeValue(writer, data);
}
```

Reading Object from JSON (Jackson)

```
Data data;
ObjectMapper mapper = new ObjectMapper();
try (Reader reader = Files.newBufferedReader(src)) {
    data = mapper.readValue(reader, Data.class);
}

Movie movie = data.getMovies().get(0);
log.info("Found movie {} ({})", movie.getTitle(),
movie.getYear());
```

JAVA SERIALIZATION

Serializable Classes

```
class Data implements Serializable {
  List<Movie> movies;
}

class Movie implements Serializable {
  String title;
  int year;
}
```

Writing Object

```
List<Movie> movies = new ArrayList<>();
movies.add(new Movie("Interstellar", 2014));
Data data = new Data(movies);
try (ObjectOutputStream out = new ObjectOutputStream(
      new BufferedOutputStream(
        Files.newOutputStream(dest)))) {
   out.writeObject(data);
}
```

Reading Object

```
Data data;
try (ObjectInputStream in = new ObjectInputStream(
     new BufferedInputStream(
       Files.newInputStream(src)))) {
   data = (Data) in.readObject();
}
Movie movie = data.getMovies().get(0);
log.info("Found movie {} ({})", movie.getTitle(),
movie.getYear());
```

Alternatives

- **fast-serialization**: https://github.com/RuedigerMoeller/fast-serialization
  up to 10 times faster 100% JDK Serialization compatible drop-in replacement
- **kryo**: https://github.com/EsotericSoftware/kryo
  also very fast but is incompatible with JDK Serialization

Summary

- For handling ZIP files use zt-zip,
  for other archives use Commons Compress
- For reading passwords use
  Console.readPassword()
- For logging use SLF4J and Logback
- For handling XML files use JAXB or JAXP
  according to your needs
- For handling JSON files use GSON or Jakcson

Summary (2)

- Use **ProcessBuilder** or **zt-exec** to start sub processes
- Handle sub process I/O correctly, either
  - Inherit I/O or redirect to files (since Java 7)
  - Read the output
  - Use zt-exec
- Remember to stop your sub process
- If stopping processes is critical to your application use **Java 9** or **zt-process-killer**

QUESTIONS?

Homework #5

https://github.com/JavaFundamentalsZT/jf-hw-sysdump

Your Task

Write a Java program which outputs system info as XML and JSON files. Implement the following interface:

```
interface SystemDump {
  Info newInfo();
  void writeXml(Info src, Path dest);
  void writeJson(Info src, Path dest);
}
```

System Info

```
interface Info {
  // sorted map of all environment variables.
  Map<String, String> getSystemEnvironment();
  // sorted map of all system properties.
  Map<String, String> getSystemProperties();
  // output of a system-dependent command line tool
  //  On Windows: cmd /c ver
  //  On others: uname -a
  String getSystemVersion();
}
```

Requirements

1. Both Maps have to be sorted A-Z by keys.
2. Execute a proper command line tool depending on the OS and gather its output.
3. Use XML and JSON APIs (don't write a String directly).
4. Buffer data for better performance.
5. Close all resources properly.

Example JSON


```
{
  "systemEnvironment":{
    "HOME":"/Users/mati",
    "MAVEN_OPTS":"-Xmx1024M"
  },
  "systemProperties":{
    "java.specification.name":"Java Platform API Specification",
    "path.separator":":"
  },
  "systemVersion":"Darwin Stalker.local 15.6.0 Darwin Kernel
Version 15.6.0: Mon Aug 29 20:21:34 PDT 2016;
root:xnu-3248.60.11~1/RELEASE_X86_64 x86_64\\n"
}
```

Example XML

```
<systemDump>
  <systemEnvironment>
    <entry key="HOME">/Users/mati</entry>
    <entry key="MAVEN_OPTS">-Xmx1024M</entry>
  </systemEnvironment>
  <systemProperties>
    <entry key="java.specification.name">Java Platform API Specification</
entry>
    <entry key="path.separator">:</entry>
  </systemProperties>
  <systemVersion>
    Darwin Stalker.local 15.6.0 Darwin Kernel Version 15.6.0: Mon Aug 29
20:21:34 PDT 2016; root:xnu-3248.60.11~1/RELEASE_X86_64 x86_64\n
  </systemVersion>
</systemDump>
```

Unit Tests

- The project has a set of unit tests.
- To get the maximum points all those tests must pass (run `mvn clean test`).
- Existing tests cannot be altered.