Input/Output resources

Table of Contents

---

1. Introduction

Input/Output (IO) resources are defined in HTSJDK-NEXT as the file or combination of files to read input data. This document describes the design for the classes and how could be provided.

1.1. Requirements

- IO resources should be handled independently of the file system in use.
- Linux/MacOS/Windows operating systems should be supported by default.
- IO resources should be able to retrieve companion files.

2. Design

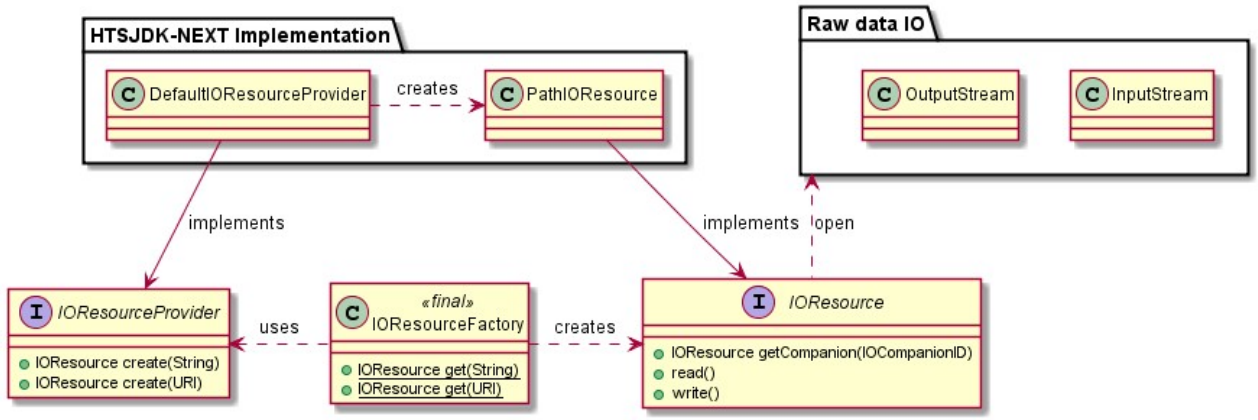
IO resources are implemented in HTSJDK as a Service Provider Interface (SPI). The service should be implemented from the `org.htsjdk.core.spi.IOResourceProvider`.

`IOResource` is the core class in the design. It provides the classes to read/write raw data to the resource. In addition, `IOResource` provides a method to return companion resources by identified by an `IOCompanionID`. `IOCompanion` is just a marker interface that should be recognized by the `IOResource` to provide the extra resource. For example, an `IOIndexCompanion` might provide a method to get the index extension to resolve a sibling file.

`IOResources` are created by the `IOResourceFactory`. `IOResourceFactory` loads the providers through the `java.util.ServiceLoader` (https://docs.oracle.com/javase/8/docs/api/java/util/ServiceLoader.html).

Only the first provider is used and should be specified. If no provider is added into the **META-INF/services**, it delegates into the `DefaultIOResourceProvider` The `DefaultIOResourceProvider` returns an implementation reading the resource with the `java.nio.file.spi.FileSystemProvider` (https://docs.oracle.com/javase/8/docs/api/java/nio/file/spi/FileSystemProvider.html). This allows to use the Java SPI for custom file systems.

2.1. Classes



"Raw data IO" design is not final, as it returns `java.io` raw classes. `IOResource` API for read/write will evolve and probably support random access.

3. Custom implementations

Implementations for the SPI are useful to provide resources out of the scope of the core library. For example, a configuration file could be a resource identifying a file and their companions. Another example is an URI which identifies a resource that should be read in a different way by the high-level interfaces.

Requirements

- Accept all core implementations of `IOCompanionID`.
- Provide the **META-INF/service/org.htsjdk.core.spi.IOResourceProvider** file with the implementation definition.

Optional

- Fall-back to default implementation if not present.

Examples

- Provider for configuration files.
  - If a resource represents a file defining the master file and their companions.
  - If a resource requires some extra-configuration, like encrypted keys for decoding.
- Provider for custom URI schemas.
  - If an URI represents an specific format of the file, with a header that shouldn't be read.
  - If the resource represents a format that should be read in a different way by the high-level interfaces.
  - If an URI should propagate signatures to the companion files.
- Generic provider.
  - If a provider chooses between several loaded implementations depending on the String or URI.