Command Parser-1.1

User Manual

Liubin Zhang, Yangyang Yuan, Wenjie Peng

Table of Contents

Introduction

Usage Guide

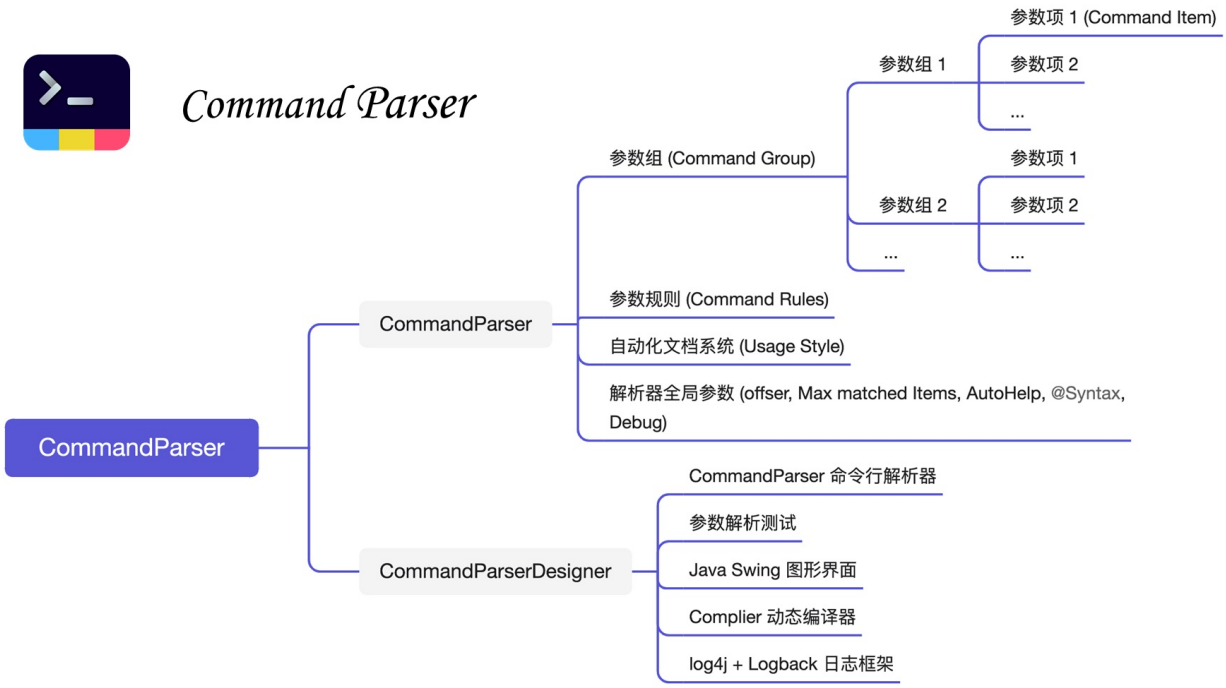
Example

About Command Parser

CommandParser is a lightweight Java platform for quickly developing, parsing, and managing command-line arguments. It provides a basic graphical interface (CommandParserDesigner) for managing and editing command items visually.

CommandParser has the following advantages:

- **Cross-platform**: It is developed based on Java8, and all software libraries are standardized, which can be run in a wide range of devices and JDK versions.
- **GUI-platform**: Commandparserdesigner-1.1. jar provides user-friendly graphical design and management functions to help developers easily preview and manage parsers with hundreds or thousands of parameters. Graphically designed parsers can directly export Java script files to run.
- **Script designer**: CommandParser supports pure script development, and parameter registration can be completed by invocation chaining. Script files can also be imported into GUI platform for visually editing and management.
- **Lightweight**: The commandParser-1.1.jar only occupies 400+ KB with no external dependencies. The core program package is separated from the graphical interface program to reduce the package size.
- **Single line parse**: It completes parsing of string arrays or files by `parser.parse(...)`.
- **Automated documents and highly customizable**: automated documents meet almost all requirements for document designing, and open interface design allows users to customize document styles.
- **Support predetermined rules between parameter sets**: complete verification of mutual exclusion and parameter dependency relationships in the parsing stage to reduce the amount of extra code;
- **Development mode**: Allows debugging parameters and user parameters to be developed in the same script file without interfering with each other.



> [!COMMENT|label:Contact Developer] Liubin Zhang, suranyi.sysu@gmail.com

What is command-line-interface?

Command Line Interface (CLI) is a text-based user interface for program running, file management, and computer interaction. Similar to software graphical interface and Web service, command-line interface helps implement the transformation between the internal form of program operation and the human-acceptable form. Generally, the command line interface receives the commands from external hardware (e.g. keyboard), and parses the commands into parameters or associated settings, and then initiates the back-end computing tasks.

Packaging the program as a runnable JAR package in Java facilitates its transmission and use on different platforms, without considering the IDE environment and tedious configuration. A JAR package usually contains all the codes for a set of complete calculation, analysis and service. In order to enable the JAR package to execute different tasks according to input parameters, the command line needs to be efficiently parsed. Thus, an effective command line parsing tool will helps improve development efficiency and interactive experience.

Download

CommandParser was developed in JDK 8. Benefit from the cross-platform and backward compatibility of Java, it also works in all hardware and software environments that support the Java8.

- commandParser-1.1.jar: Used to be imported into other projects to build a minimal runtime environment for driving parse work.
- commandParserDesigner-1.1.jar: Used to start graphical interface for management and development.

Type	URL
Package	http://pmglab.top/commandParser/commandParser-1.1.jar
Designer	http://pmglab.top/commandParser/commandParserDesigner-1.1.jar
Source	https://github.com/Zhangliubin/commandParser-1.1
Usage	http://pmglab.top/commandParser/
API	http://pmglab.top/commandParser/api-docs/
Example	http://pmglab.top/commandParser/bgzip/BGZToolkitParser.java http://pmglab.top/commandParser/downloader/HttpDownloaderParser.java

Download software package by wget

```
mkdir commandParser-1.1
cd commandParser-1.1
wget http://pmglab.top/commandParser/commandParser-1.1.jar
wget http://pmglab.top/commandParser/commandParserDesigner-1.1.jar
```

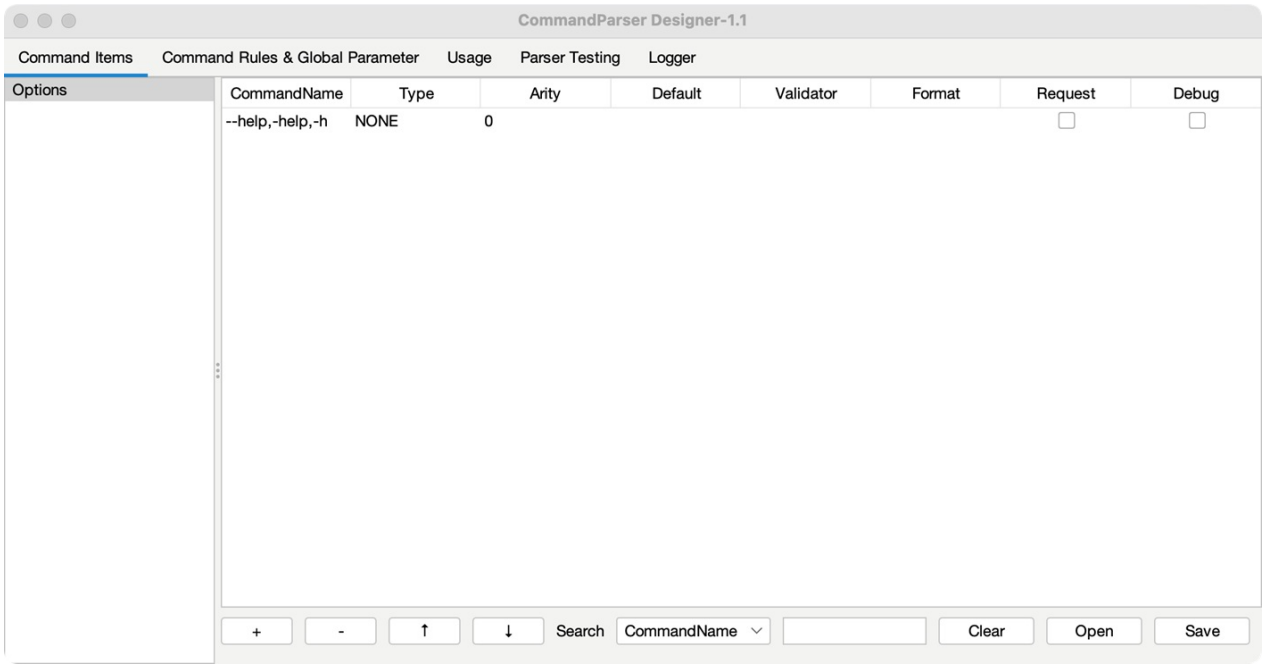
Updates

> [!UPDATE|label:2022/06/01]
>
> - Release the second version of CommandParser, version number 1.1, Github: https://github.com/Zhangliubin/commandParser-1.1
> - The old version now stops updating and maintenance.

> [!UPDATE|label:2021/11/23] Release the first version of CommandParser, version number 1.0, Github: https://github.com/Zhangliubin/commandParser-1.0

Launch Software

Enter `java -jar ./commandParserDesigner-1.1.jar` in the terminal or double click JAR package to start the graphical interface.



> [!TIP|label:Set shortcut command]
>
> Use the following way to set `java -jar . /commandParserDesigner-1.1.jar` as a shortcut command on Macos or Unix systems.

```
# open the file of environment variables
vim ~/.zshrc

# Add instruction shorthand (${path} is the abosulotely path of "commandParserDesigner-1.1.jar")
alias commandParser="java -jar ${path}"

# Click Esc, enter ":x" and press "Enter", sava and quit this file
source ~/.zshrc
```

Check For Updates

Click the right mouse button on the tab bar to expand the menu. Clicking "Check for Updates" will help to check for the new version of `Command Parser` and `Command Parser Designer`.

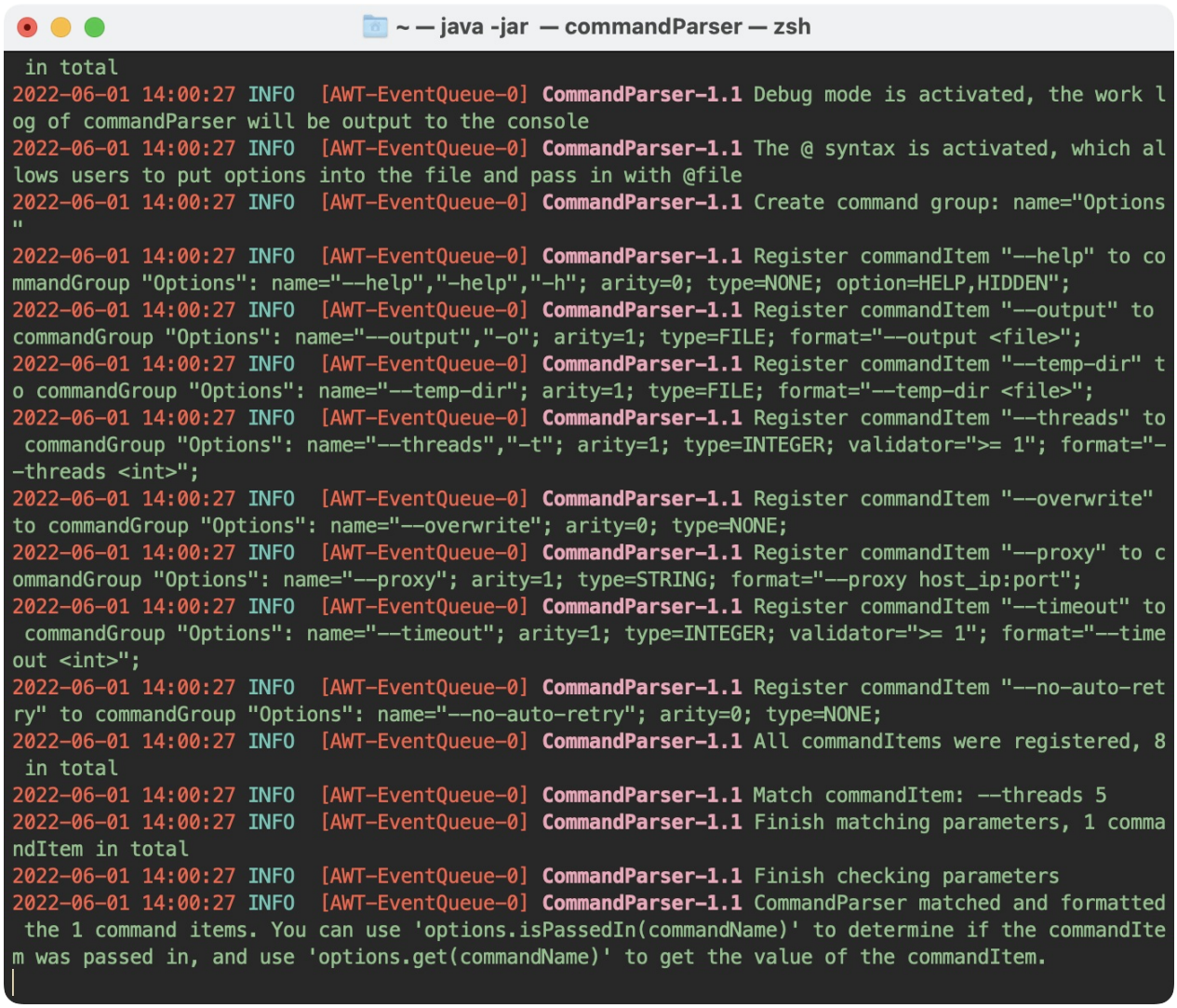
View Logbook

Users can view the work log of the current program in the `Logger` tab, the log system was built based on log4j + Logback.

> [!DANGER|label:Command Parser Designer is only used as single software]

> Please launch the GUI interface before loading the log system for redirecting logs to the JSwing panel requires successfully. Thus, when importing the commandParserDesigner.jar as package, GUI components will be loaded whether or not the GUI is actively invoked, and then effects the main business logic.

Log information is also displayed at the terminal when starting from the terminal (garbled characters will be displayed if the terminal does not support this character set):

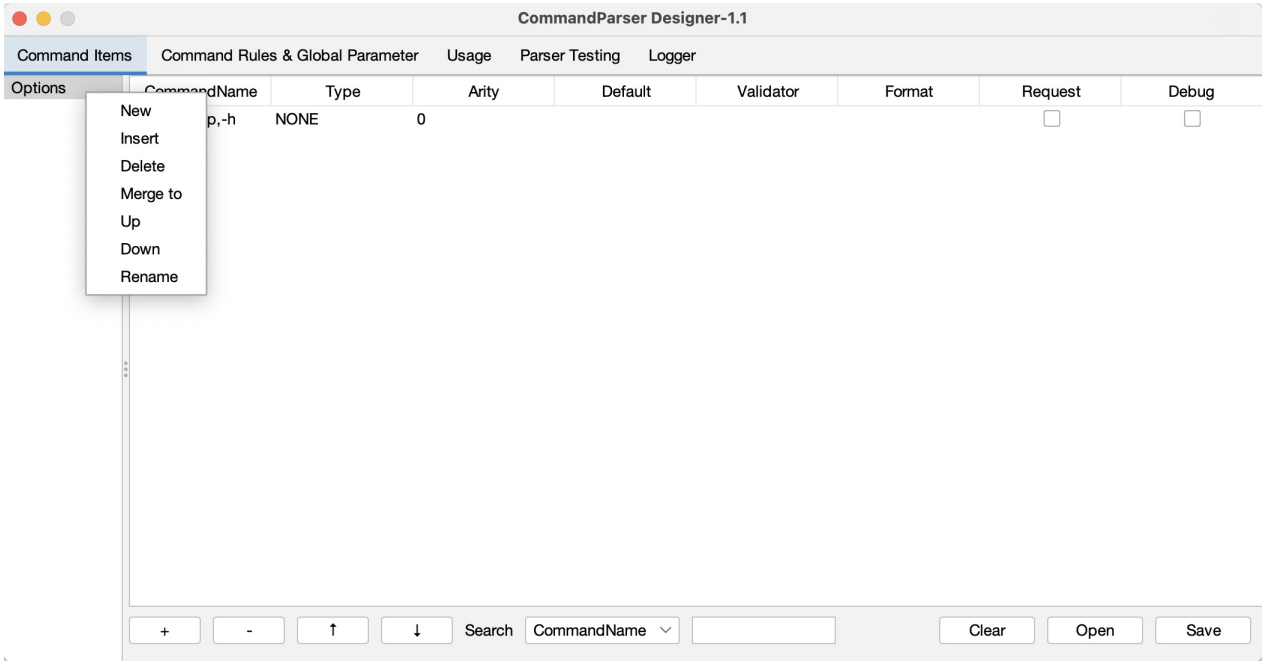

~ — java -jar — commandParser — zsh
in total
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Debug mode is activated, the work log of commandParser will be output to the console
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 The @ syntax is activated, which allows users to put options into the file and pass in with @file
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Create command group: name="Options"
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--help" to commandGroup "Options": name="--help","-help","-h"; arity=0; type=NONE; option=HELP,HIDDEN";
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--output" to commandGroup "Options": name="--output","-o"; arity=1; type=FILE; format="--output <file>";
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--temp-dir" to commandGroup "Options": name="--temp-dir"; arity=1; type=FILE; format="--temp-dir <file>";
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--threads" to commandGroup "Options": name="--threads","-t"; arity=1; type=INTEGER; validator=">= 1"; format="--threads <int>";
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--overwrite" to commandGroup "Options": name="--overwrite"; arity=0; type=NONE;
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--proxy" to commandGroup "Options": name="--proxy"; arity=1; type=STRING; format="--proxy host_ip:port";
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--timeout" to commandGroup "Options": name="--timeout"; arity=1; type=INTEGER; validator=">= 1"; format="--timeout <int>";
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Register commandItem "--no-auto-retry" to commandGroup "Options": name="--no-auto-retry"; arity=0; type=NONE;
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 All commandItems were registered, 8 in total
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Match commandItem: --threads 5
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Finish matching parameters, 1 commandItem in total
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 Finish checking parameters
2022-06-01 14:00:27 INFO [AWT-EventQueue-0] CommandParser-1.1 CommandParser matched and formatted the 1 command items. You can use 'options.isPassedIn(commandName)' to determine if the commandItem was passed in, and use 'options.get(commandName)' to get the value of the commandItem.

Command Group Management

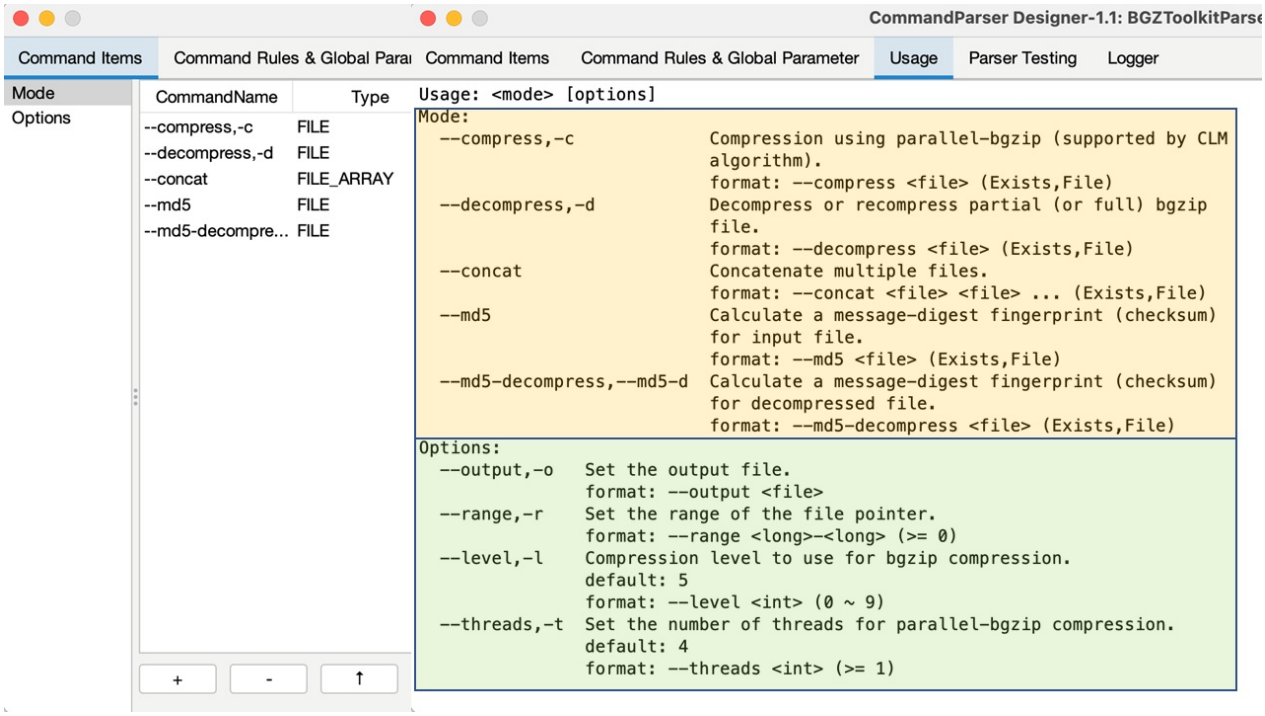
Command items can be roughly divided into several categories according to their functions or attributes, these categories are named as command groups. In CommandParser, the command group is the basic unit for organizing command items. When initializing the parser, a command group name `Options` will be created by default and the first command item (`--help, -help, -h`) used to export help document in this command group will be created automatically.

On the `Command Items` tab, the command group panel is on the left. Click the right mouse button in the blank area of the command group panel or select the command group to expand the management menu. The management menu contains the following seven operations:

- **New:** Create a new command group.
- **Insert:** Insert a new command group at the current location.
- **Delete:** Delete the selected command group.
- **Merge to:** Merge all command items from this command group to another command group.
- **Up:** Move up the selected command group.
- **Down:** Move down the selected command group.
- **Rename:** Rename the command group.



The order of the command groups will affect the order of the command groups displayed in the automated document:

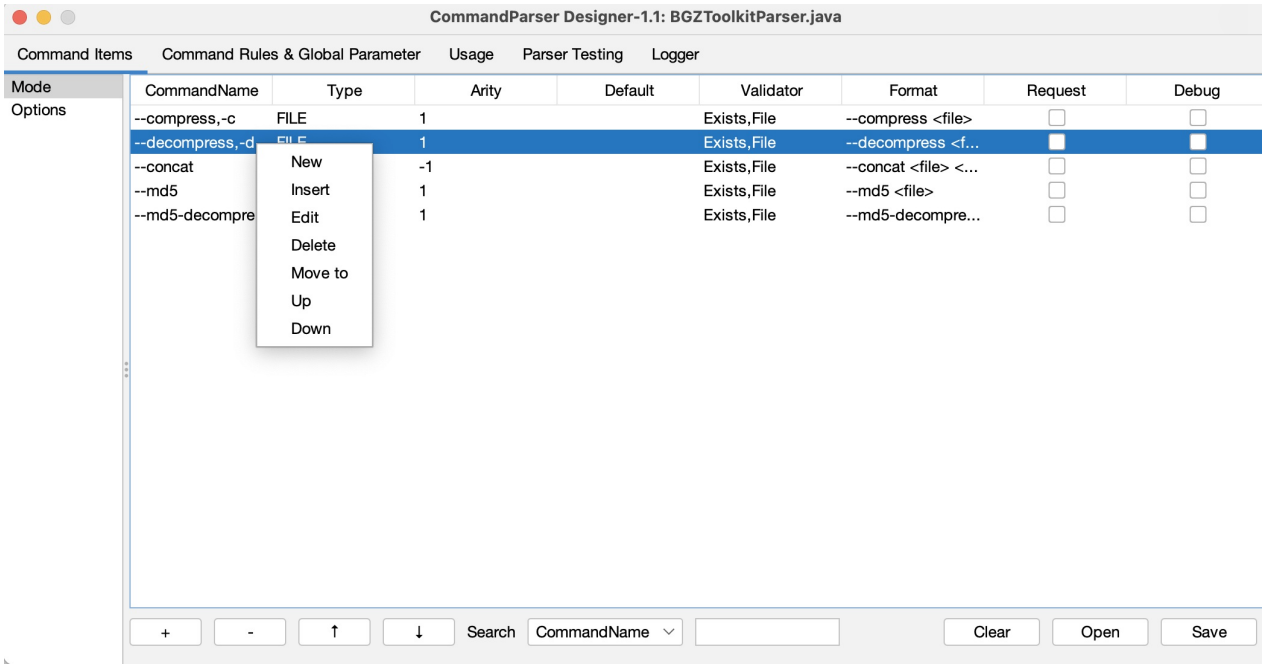


Command Item Management

On the `Command Items` tab, the command items panel is on the right. Right click a command item or the blank area of the command items panel to expand the management menu. The management menu contains the following seven operations:

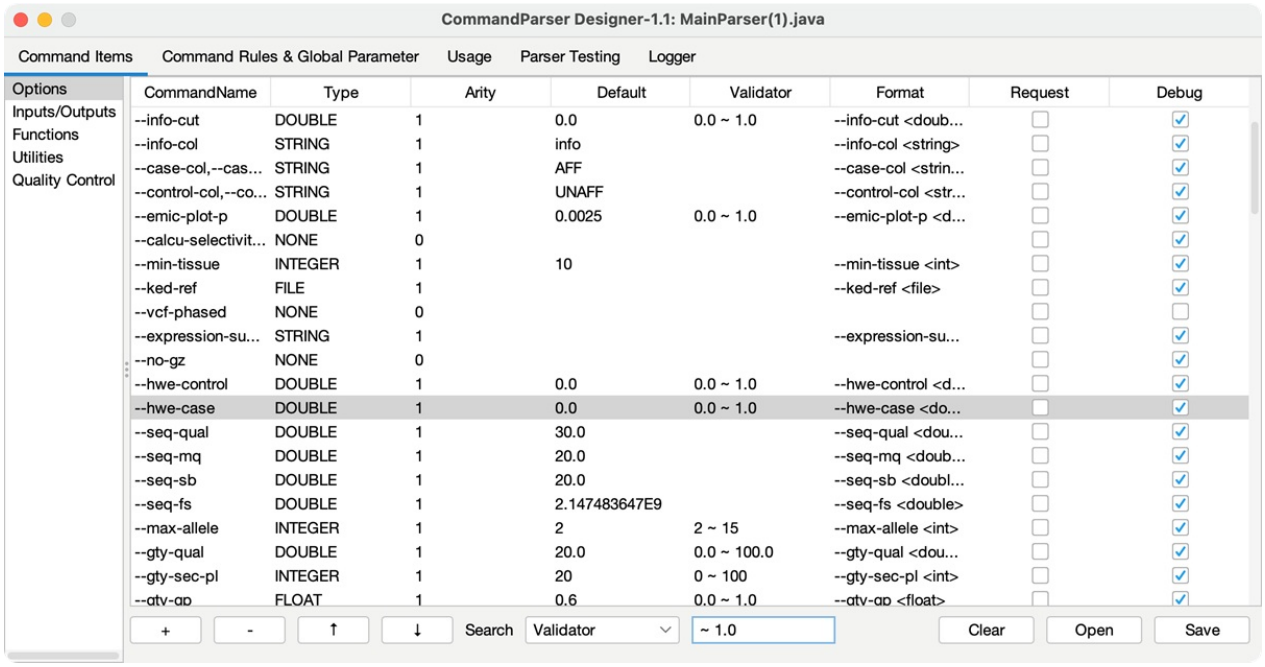
- **New:** Create a new command item (shortcut: Ctrl + N).
- **Insert:** Insert a new command item at the current position.
- **Edit:** Edit the command item (shortcut: Double-click the command item).
- **Delete:** Delete the selected command item (shortcut: Ctrl + Delete).
- **Merge to:** Merge the selected command item to another command group.
- **Up:** Move up the selected command item (shortcut: Ctrl + U).
- **Down:** Move down the selected command item (shortcut: Ctrl + D).

In the menu bar below, "+" corresponds to New, "-" corresponds to Delete, "↑" corresponds to Up, and "↓" corresponds to Down.



Search Command Item

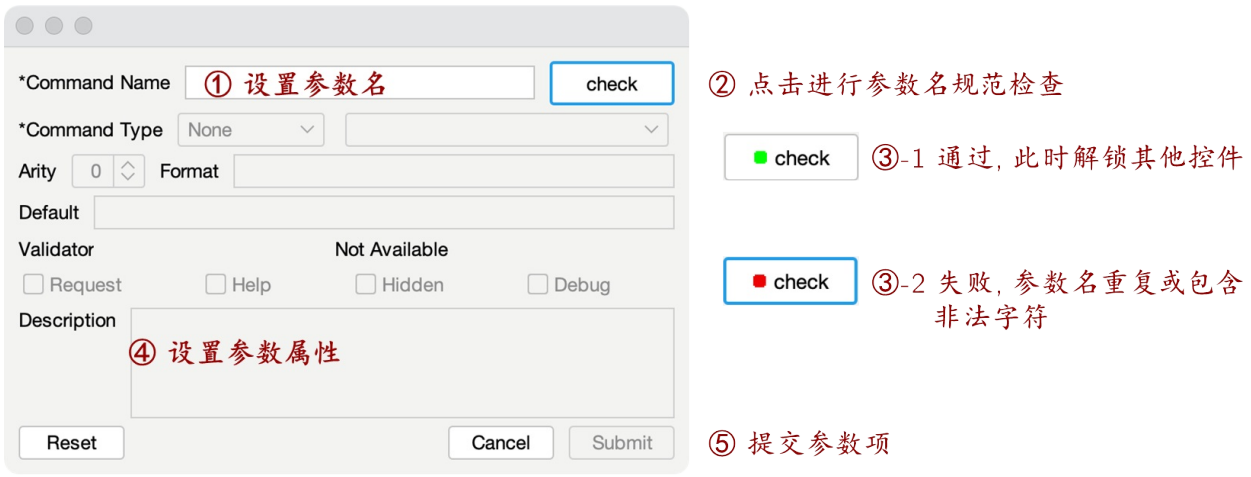
In the search box at the lower part of the command items panel (shortcut: Ctrl + F), users can set the search attribute (check box) and content (text box, ignores case), and then press the Enter to search the matched command items.



Edit Command Item

The basic work unit of the CommandParser is Command Item, which has declared the keyword, the number of parameters, the transformation from the captured values to Java object, and the validation of the captured values.

When creating or editing a command item, the command item sub-panel will pop up, which contains 12 command properties. `Command Name` and `Command Type` are two mandatory attributes, which help to define the keyword and data Type of this command item.



The description of each property is shown as following:

Propertie	Description
Command Name	1. Separate with commas if the command item has multiple names (e.g. `--output,-o`), and the first name will be regarded as the main command name. 2. Format of parameter name: 0-9A-ZA-Z +-_. 3. Click the Check button to check for the format and duplicate names after entering the command name. Only the eligible command name can have other properties and be submitted.

Command Type	1. Including None (by default, check whether be passed in or not), Boolean, Byte, Short, Integer, Long, Float, Double, String, File. 2. Derived data types: Value, Array, Set, Range etc.
Validator	Different types of command item have different validators, see details in Validator.
Default	The default value should be transform by Command Type and validated by the Validator. And the input format should be consistent with the format defined by `Format` and `Arity`.
Arity	Set the arity. "-1" indicates an indefinite length (0 parameter is also approved). All parameters before the next command item are regarded as the value of this command item.
Format	The format is set according to Command Type (usually be set as: Primary Command Name + Default format).
Description	The description document of command item.
Help	When a command item marked `Help` is passed in, the format transformation and validators will be disabled.
Request	The command item marked `Request` must be passed in.
Hidden	The command item marked `Hidden` will not be displayed in the document.
Debug	When the parser is in `non-Debug mode`, the command item marked `Debug` is disabled, and the corresponding command rule will be invalidated, and the command item will not be displayed in the document.

Command Type

Command Type `MainType.DerivedType`. Main types include IType.NONE (check whether be passed in or not), BOOLEAN, BYTE, SHORT, INTEGER, LONG, FLOAT, DOUBLE, STRING, FILE. Derived types include the following 16 types. The default format document (i.e., `CommandName Format`) will be assigned once the type is specified. Types with the `...` tag can use `arity` to control the number of parameters.

Command Derived Type	**Default Format**
VALUE	value
ARRAY	value value …
ARRAY_COMMA	value,value,…
ARRAY_SEMICOLON	value;value;…
SET	value value …
SET_COMMA	value,value,…
SET_SEMICOLON	value;value;…
MAP	key=value key=value …
MAP_COMMA	key=value,key=value,…
MAP_SEMICOLON	key=value;key=value;…
RANGE	value-value
LABEL_RANGE	label:value-value label:value-value …
LABEL_RANGE_COMMA	label:value-value,label:value-value,…
LABEL_RANGE_SEMICOLON	label:value-value;label:value-value;…
LABEL_ARRAY	label:value,value,… label:value,value,…
LABEL_ARRAY_SEMICOLON	label:value,value,…;label:value,value,…;…

Validator

Validators for different types of command item are shown as below:

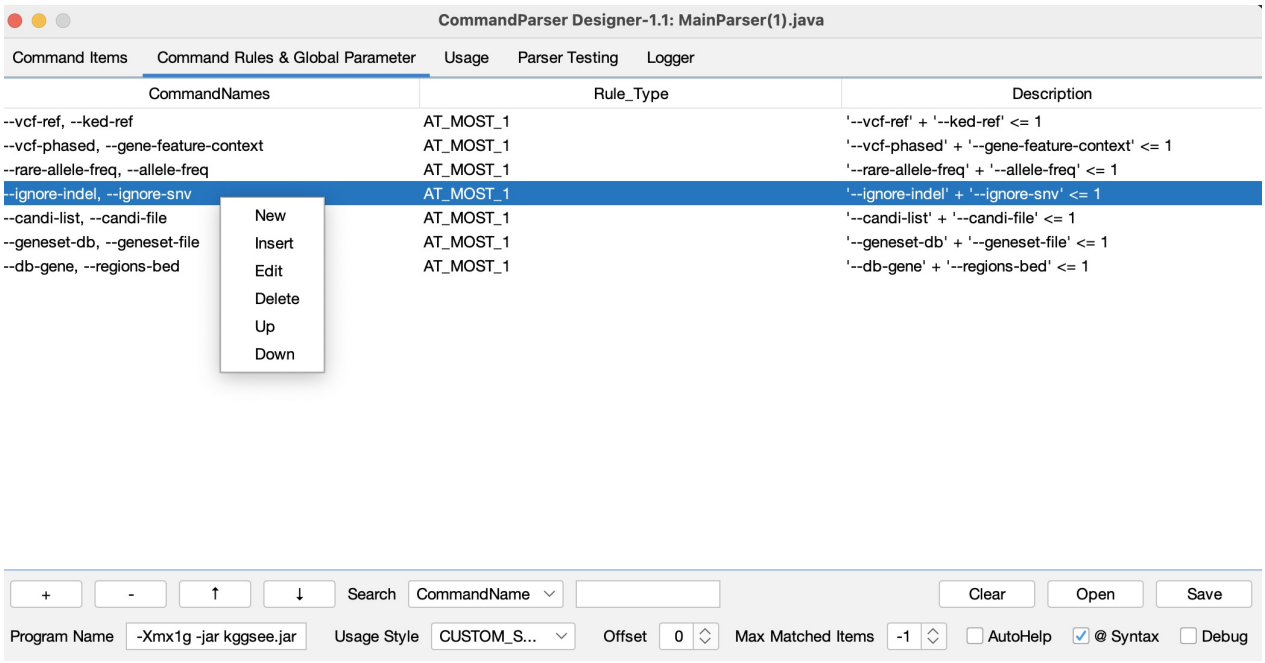
Command Type	Support Types
None, Boolean	Don't support validator
Byte, Short, Integer, Long, Float, Double	Validators for numeric range : 1. Range (including boundary values): minimum to maximum; 2. Specify minimum value: ≥ minimum value.
String	Validators for qualified values: Multiple qualified values are separated by spaces. ignoreCase: whether to ignore case or not; indexAccess: allow the use of indexes instead of specific values (0 represents the first qualified value...).
File	Validators for files: File Exists: The file path must exist. Single File: The file path cannot be a folder; Directory: The file path must be a folder

Command Rule Management

Add command rules between command items in the `Command Rules & Global Parameter` tab. Click the right mouse button in the blank area of the command rule panel or select the command rule to expand the management menu. The management menu contains the following six operations:

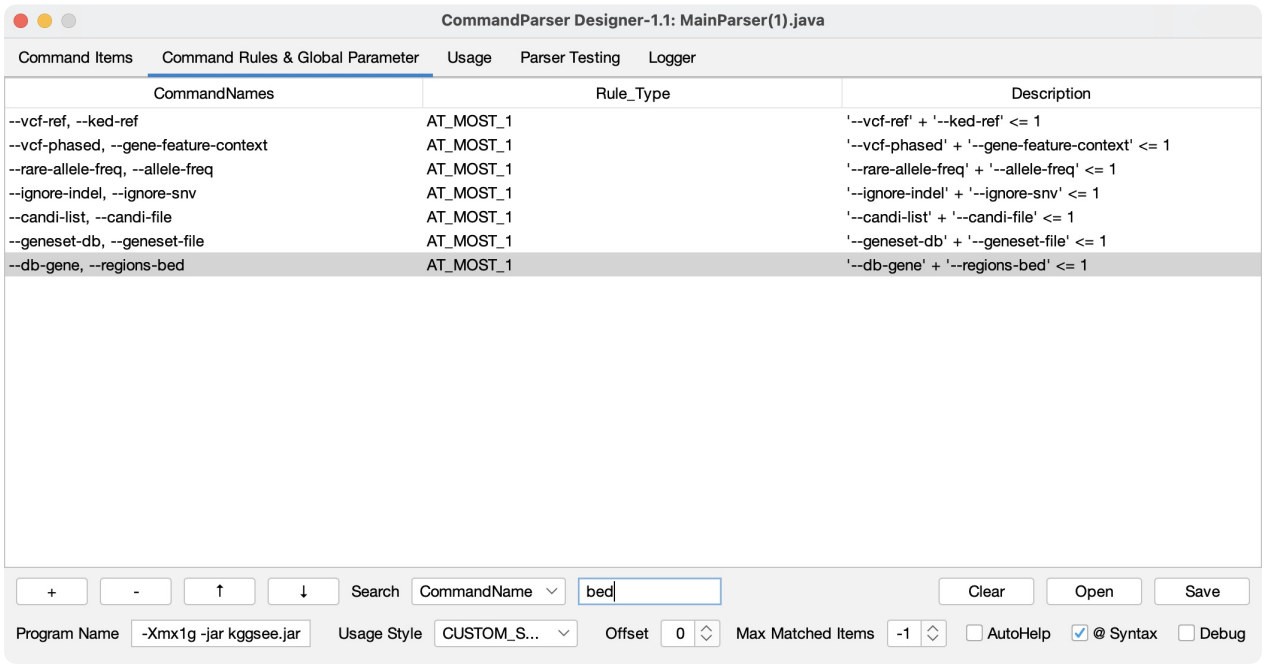
- **New:** Create a new command rule (shortcut: Ctrl + N).
- **Insert:** Insert a new command rule at the current location.
- **Edit:** Edit the specified command rule (shortcut: Double-click the command rule).
- **Delete:** Delete the specified command rule (shortcut: Ctrl + Delete).
- **Up:** Move up the specified command rule (shortcut: Ctrl + U).
- **Down:** Move down the specified command rule (shortcut: Ctrl + D).

In the menu bar below, "+" corresponds to New, "-" corresponds to Delete, "↑" corresponds to Up, and "↓" corresponds to Down.



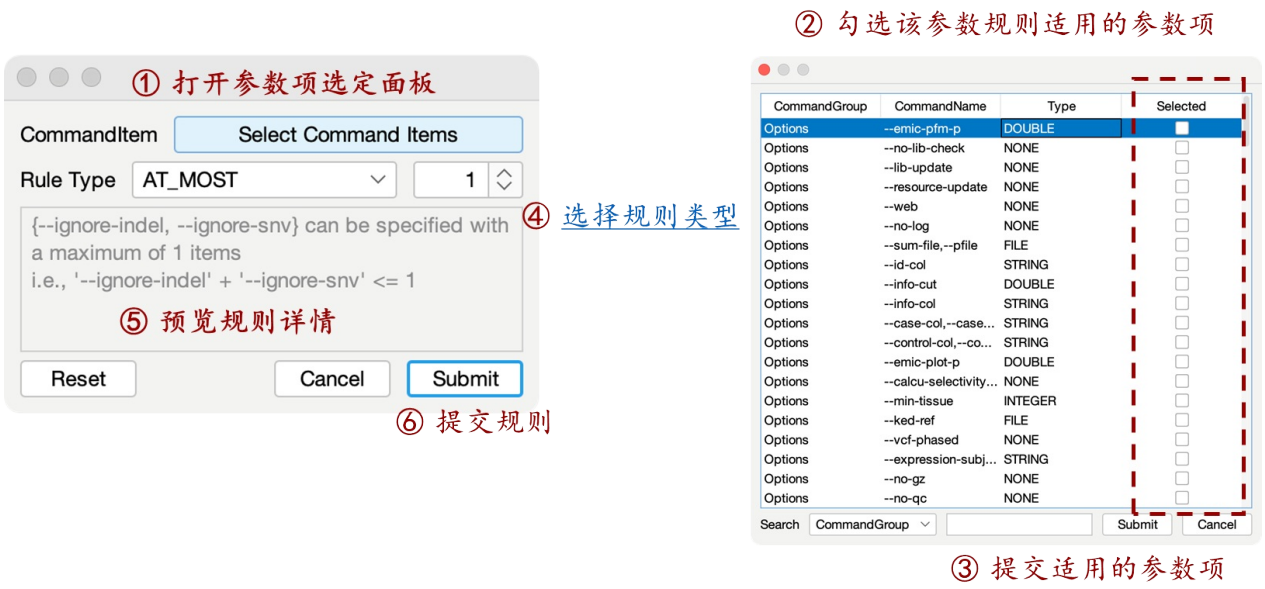
Search Command Rule

In the search box (Ctrl + F) at the lower part of the command rule panel, users can set the search attribute (check box) and content (text box), and then press Enter to search the matched command rule.



Edit Command Rule

When creating or editing a command rule, the command rule sub-panel will pop up. Set rules for command items including "Help", "Request" is not allowed.



Command Rule Type

When the command items are specified from $\{p_1, p_2, \cdots, p_n\}$, the meanings of different command rule types are showing as follows:

Command Rule Type	Conditional Number	Description
AT_MOST	k	$p_1, p_2, \cdots, p_n$ can be specified with a maximum of k items.
AT_LEAST	k	$p_1, p_2, \cdots, p_n$ should be specified with at least k items.

EQUAL	k	$p_1, p_2, \cdots, p_n$ should be specified with k items.
MUTUAL_EXCLUSION	k	$p_1, p_2, \cdots, p_k$ and $p_{k+1}, p_{k+2}, \cdots, p_n$ are not allowed to be used together.
SYMBIOSIS	/	$p_1, p_2, \cdots, p_n$ should be specified concurrently or not at all.
PRECONDITION	/	When p_j is passed in， $p_i (i < j)$ should be passed in concurrently.

Set Global Property

Set the global properties for CommandParser in `Command Rules & Global Parameter` tab. Global properties include followings options:

- **Program Name**
- **Usage Style:** The format of automatic document. Double click the check box may help to edit format, and choose "..." will help to create a new document format.
- **Offset:** offset of input commands.
  - Skip the first `offset` commands of the input commands.

```
# when offset = 3, the following commands will skip the first three parameter and parse "--level 5 -t 4 -o ~
/test.gz"
bgzip compress <file> --level 5 -t 4 -o ~/test.gz
```

- **Max Matched Items:** Set the maximum number of matched command items, 0 and -1 indicates no limitation.
  - When reaching the maximum number of command items, the subsequent commands are no longer parsed, and will be regarded as the parameter value of the last matched command item.

```
# when maxMatchedItems = 1, the following commmands only matched "bgzip" and following parameters, "compress
<file> decompress <file>",  will be regarded as the value of "bgzip"
bgzip compress <file> decompress <file>
```

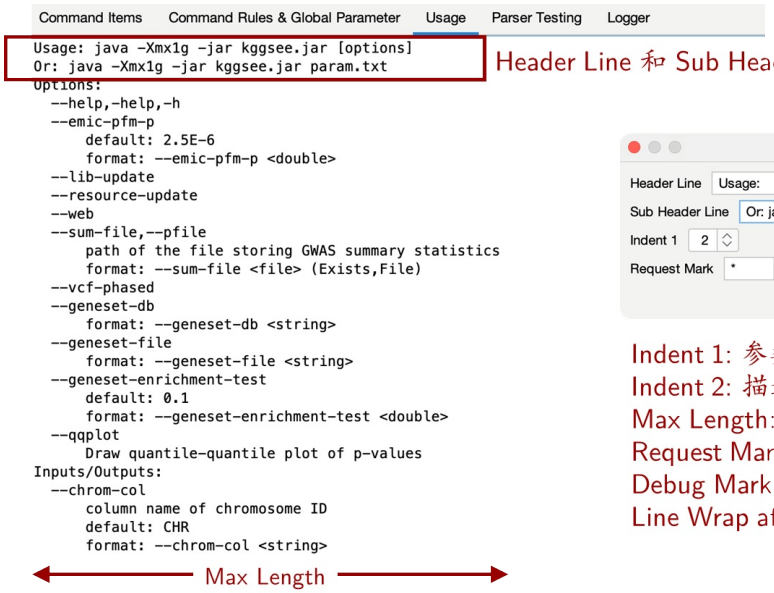
- **AutoHelp:** When no parameter is passed in, add "help" parameter automatically.
- **@Syntax:** `@` Grammar switch
  - Under `@` grammar, the parameter will be replaced by file content if“@file”exist.

```
# If the file content is:  compress <file> --level 5 -t 4, the following two commands will have the same res
ults.
bgzip @file -o ~/test.gz
bgzip compress <file> --level 5 -t 4 -o ~/test.gz
```

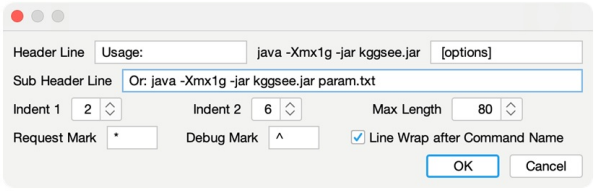
- **Debug:** Debug model switch. The command items tagged with "Debug" will only be shown and used in debug mode.

Setting the format of automated documents

On the `Command Rules & Global Parameter` tab, double click the `Usage Style` check box or drop down to select `...` , and open the editor for document format. The format editor is used to control the automated document format in the `Usage` tab (as shown on the left), where users can implement copy and search (Ctrl + F).



Header Line 和 Sub Header Line



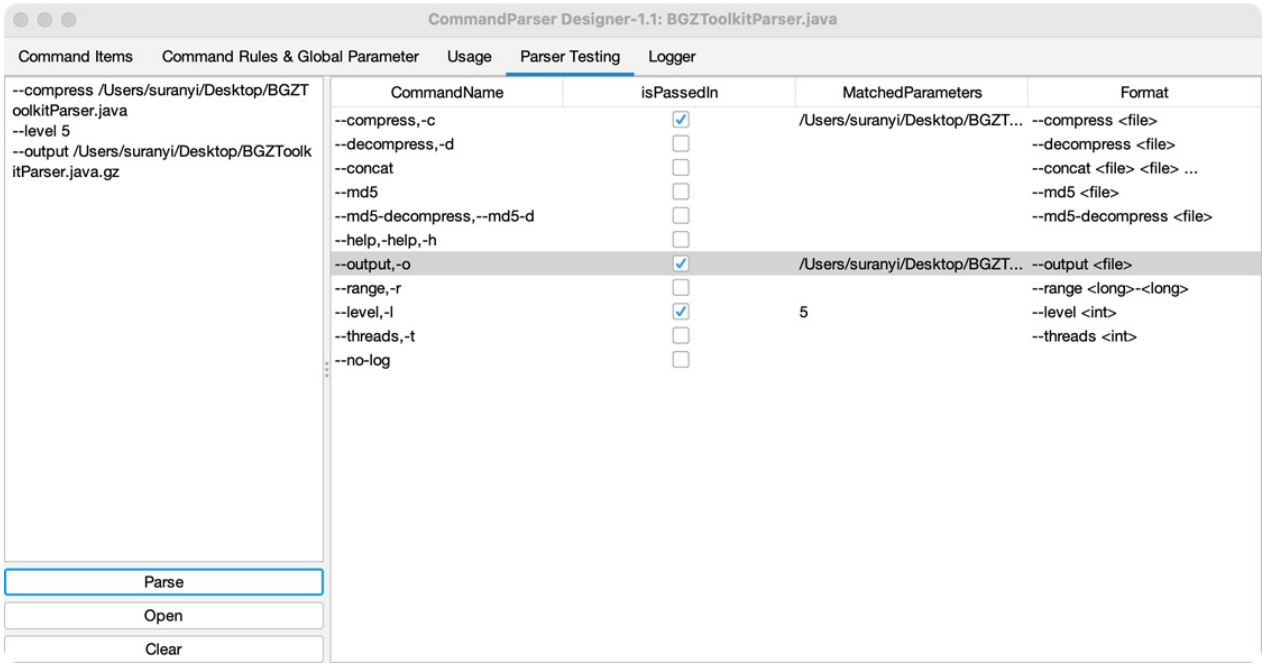
Indent 1: 参数名前空格数
Indent 2: 描述文档前空格数
Max Length: 文档最大宽度 (超过宽度自动换行)
Request Mark: 必备参数前添加的标记
Debug Mark: 调试参数前添加的标记
Line Wrap after Command Name: 参数名后换行

Test Parse Parameter

Parameter parsing can be tested in `Parser Testing` tab. There will be a more detailed log file under Debug mode. Input commands in edit box and click `Parse` button, the result of the parsing will be displayed in the right panel:

- **isPassedIn:** Whether the command item is passed in or not.
- **MatchedParameters:** The captured value of this command item.
- **Format:** The format of this command item.

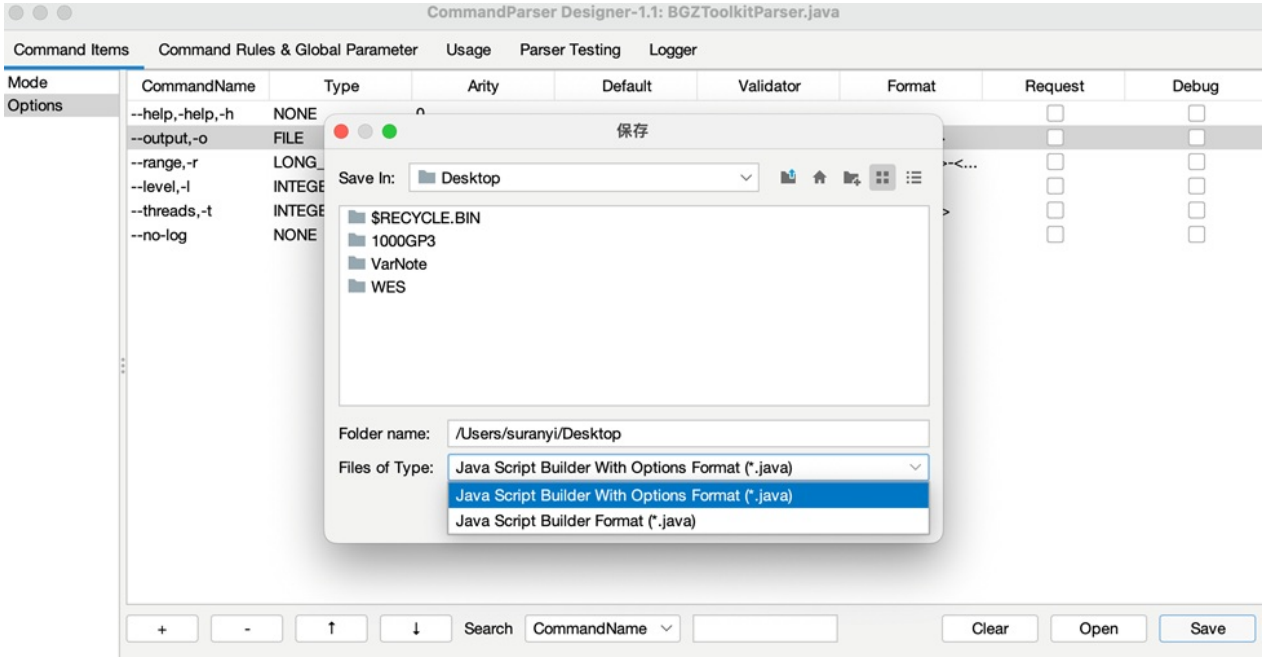
Double click command item will skip to the command item position in the `Command Items` tab.



Export to Java Script

On the `Command Items` tab or the `Command Rules & Global Parameter` tab, click the `Save` button at the right corner below to save the current parser (global shortcut: Ctrl + S). Two formats for saving are available:

- **Java Script Builder With Options Format:** Single instance of parser and the creation of java variables.
- **Java Script Builder Format:** Single instance of parser.



Java Script Builder With Options Format

`Java Script Builder With Options Format` infers the variable name (camel naming) from the main command name of the command item, and sets the variable type according to the command type of the command item, without a format conversion of variables. This format supports the following API methods:

API	Return Type	Description
Parser options = Parser.parse(String[] args)	Parser	Parser input command.
Parser options = Parser.parse(File argsFile)	Parser	Parser input command in file.
Parser.usage()	String	Get parser's document.
Parser.getParser()	CommandParser	Get parser's object.
parser.getOptions()	CommandOptions	Get parser's command item value's object.
options.commandName.value	T (Generics)	Get command item value (When not passed in, value is default value)
options.commandName.isPassedIn	boolean	Whether this command item is passed in.
options.commandName.matchedParameter	String	The value of the parameters matched by this command item (in raw string format).

Use the parser to bridge input parameters with business logic in the entry function:

```
public static void main(String[] args) {
    if (args.length == 0) {
        System.out.println(HttpDownloaderParser.usage());
```

```
            return;
        }

        HttpDownloaderParser options = HttpDownloaderParser.parse(args);
        if (options.help.isPassedIn) {
            System.out.println(HttpDownloaderParser.usage());
            return;
        }

        try {
            HttpDownloader2.instance(args[0])
                    .setOutputFile(options.output.value)
                    .setThreads(options.threads.value)
                    .setPrintLog(true)
                    .setTempDir(options.tempDir.value)
                    .setProxy(options.proxy.value)
                    .setTimeOut(options.timeout.value)
                    .clean(options.overwrite.isPassedIn)
                    .download();
        } catch (IOException e) {
            logger.error("{}", e.getMessage());
        }
    }
```

Java Script Builder Format

`Java Script Builder Format` creates a single instance of the CommandParser, which achieves accessing the parameter information (value, whether it was passed in, captured value) by command names (any command name of the command item), and requires formatting when getting the values. This format supports the following API methods:

API	Return Type	Description
Parser.parse(String[] args)	CommandOptions	Parser input command.
Parser.parse(FileargsFile)	CommandOptions	Parser input command in file.
Parser.usage()	String	Get parser's document.
Parser.getParser()	CommandParser	Get parser's object.
options.get(commandName)	Object	Get the command item's value (if not passed in, the value is the default value).
options.isPassedIn(commandName)	boolean	Whether this command item is passed in.
options.getMatchedParameter(commandName)	String	The value of the parameters matched by this command item (in raw string format).

Use the parser to bridge input parameters with business logic in the entry function:

```
public static void main(String[] args) throws IOException {
    if (args.length == 0) {
        // 没有传入参数时, 打印文档
        System.out.println(HttpDownloaderParser.usage());
        return;
    }

    CommandOptions options = HttpDownloaderParser.parse(args);
    if (options.isHelp()) {
        // 传入 help 指令时, 打印文档
        System.out.println(HttpDownloaderParser.usage());
        return;
    }

    // 业务逻辑
    try {
        HttpDownloader.instance(args[0])
                .setOutputFile((File) options.get("--output"))
                .setThreads((int) options.get("--threads"))
                .setPrintLog(!options.isPassedIn("--no-log"))
                .setTempDir((File) options.get("--temp-dir"))
```

```
                .setProxy((String) options.get("--proxy"))
                .setTimeOut((int) options.get("--time-out"))
                .clean(options.isPassedIn("--overwrite"))
                .download();
    } catch (IOException e) {
        logger.error("{}", e.getMessage());
    }
}
```

Modify the Parser File

The exported parsing file is the Java source code (as shown in the following figure). Users can modify the Java source code directly to modify the parser. In addition, users can also import the source file to the graphical interface designer for reediting (drag the file to the interface window, click the Open button, or the shortcut: Ctrl + O).

CommandParserDesigner helps to compile the source code to a class file dynamically and obtains the parser object by `.getParser()`. Finally, the graphical interface designer will reproduces the parser based on the class member's information of the parser object.

```
parser.setProgramName("<mode>");
parser.offset(0);
parser.debug(false);
parser.usingAt(true);
parser.setMaxMatchedNum(-1);
parser.setUsageStyle(DefaultStyleUsage.UNIX_TYPE_1);

CommandGroup group001 = parser.addCommandGroup( groupName: "Mode");
group001.register(FILE.VALUE,  ...commandNames: "--compress", "-c")
        .validateWith(FILE.validateWith( checkIsExists: true,  checkIsFile: true))
        .setDescription("Compression using parallel-bgzip (supported by CLM algorithm).");
group001.register(FILE.VALUE,  ...commandNames: "--decompress", "-d")
        .validateWith(FILE.validateWith( checkIsExists: true,  checkIsFile: true))
        .setDescription("Decompress or recompress partial (or full) bgzip file.");
group001.register(FILE.ARRAY,  ...commandNames: "--concat")
        .arity(-1)
        .validateWith(FILE.validateWith( checkIsExists: true,  checkIsFile: true))
        .setDescription("Concatenate multiple files.");
group001.register(FILE.VALUE,  ...commandNames: "--md5")
        .validateWith(FILE.validateWith( checkIsExists: true,  checkIsFile: true))
        .setDescription("Calculate a message-digest fingerprint (checksum) for input file.");
group001.register(FILE.VALUE,  ...commandNames: "--md5-decompress", "--md5-d")
        .validateWith(FILE.validateWith( checkIsExists: true,  checkIsFile: true))
        .setDescription("Calculate a message-digest fingerprint (checksum) for decompressed file.");
```

Initialize Parser

CommandParser has four constructions, parameter `boolean init` is used to confirm whether to create initial `help` parameter (default: true); parameter `String programName` indicates the program name of the parser (default: `<main class>`):

- CommandParser parser = new CommandParser()
- CommandParser parser = new CommandParser(boolean init)
- CommandParser parser = new CommandParser(String programName)
- CommandParser parser = new CommandParser(boolean init, String programName)

```
CommandParser parser = new CommandParser(false);
```

Set Global Property

CommandParser has 7 globe properties:

- parser.setProgramName(String programName)

  Set program name (default value: `<main class>`).

- parser.offset(int length)

  Set offset value (default value: `0`). Skip the first `offset` parameters of the input commands.

  ```
  # when offset = 3, the following commmands will skip the first three parameter and parse "--level 5 -t 4 -o ~/te
  st.gz"
  bgzip compress <file> --level 5 -t 4 -o ~/test.gz
  ```

- parser.debug(boolean enable)

  Debug mode or not (default value: `false`).

- parser.usingAt(boolean enable)

  Identify `@` as symbol of getting address or not (Under `@` grammar, the parameter will be replaced by file content if“@file”exist.) (default value: `true`).

- parser.setMaxMatchedNum(int length)

  Set the maximum number of matched command items (default value: `-1`). When reaching the maximum number of command items, the subsequent commands are no longer parsed, and will be regarded as the parameter value of the last matched command item.

  ```
  # when maxMatchedItems = 1, the following commmands only matched "bgzip" and following parameters, "compress <fi
  le> decompress <file>",  will be the value of "bgzip"
  bgzip compress <file> decompress <file>
  ```

- parser.setAutoHelp(boolean enable)

  When no parameter is passed in, add `--help` parameter automatically or not (default value: `false`).

- parser.setUsageStyle(IUsage usageStyle)

  Set the format of the document (default value: `DefaultStyleUsage.UNIX_TYPE_1`).

  ```
  // DefaultStyleUsage construct:
  public DefaultStyleUsage(String before, String after, String subTitle, int indent1, int indent2, int maxLength,
  boolean newLineAfterCommandName, String requestMark, String debugMark)
  ```

```
parser.setProgramName("<mode>");
parser.offset(0);
parser.debug(false);
parser.usingAt(true);
parser.setMaxMatchedNum(-1);
parser.setAutoHelp(true);
parser.setUsageStyle(DefaultStyleUsage.UNIX_TYPE_1);
```

Add Command Group

CommandParser adds command group by two ways and return this group:

- CommandGroup group = parser.addCommandGroup(String groupName)

  Add the command group named as `groupName`, and then return this new command group. If there exists the command group with the same name, then return the namesake.

- CommandGroup group = parser.addCommandGroup(CommandGroup group)

  If a command group with the same name already exists in the parser, register all command items to the existing command group; otherwise, add the command group to the parser.

```
CommandGroup group001 = parser.addCommandGroup("Mode");
CommandGroup group002 = parser.addCommandGroup("Options");
```

Add Command Item

Command items can be added into the parser through command groups:

- CommandItem item = group.register(IType type, String... commandNames)

  The main method. Set command type and command names, and then register command item.

- CommandItem item = group.register(Class<?> tClass, String... commandNames)

  Set command type and command name, and then register command item. CommandParser can only identify two command types using this method, including VALUE (e.g. Integer.class) and ARRAY (e.g. int[].class).

- CommandItem item = group.register(CommandItem commandItem)

  Add command item to the parser. If the namesake existing , there will be exceptions (`CommandParserException`).

When registered command item directly through the parser, the command item is registered into the last added command group:

- CommandItem item = parser.register(IType type, String... commandNames)

  Set command type and command name, and then register command item.

- CommandItem item = parser.register(IType type, String... commandNames)

  Set command type and command name, and then register command item. CommandParser can only identify two command types using this method, including VALUE (e.g. Integer.class) and ARRAY (e.g. int[].class).

- CommandItem item = parser.register(CommandItem commandItem)

  Add command item to the parser. If the namesake existing , there will be exceptions (`CommandParserException`).

> [!NOTE|label:Command Type `MainType.DerivedType`]
>
> The main types include IType.NONE (be passed in or not), BOOLEAN, BYTE, SHORT, INTEGER, LONG, FLOAT, DOUBLE, STRING, FILE. Derived types include 16 types listed in input format of parameter type.

Set Properties of Command Item

After setting properties for command item, it will return a reference to this object, and this process can be achieved by chain calls. The meaning of the properties is shown in Edit Command Item.

- CommandItem item = item.addOptions(String... options)

  Add Options: HIDDEN, HELP, REQUEST, DEBUG.

- CommandItem item = item.arity(int length)

  Set the arity. Only the variable-length parameter can have access to this setting.

- CommandItem item = item.defaultTo(String... defaultValue)

  Set the default value (using string-array as input, converted to the corresponding value according to the converter).

- CommandItem item = item.validateWith(IValidator validator)

  Set validator for command item. Different types of command item have different validators, see details in Validator.

Command Type	**Validator Support Type**
None, BOOLEAN	Do not support validator.
BYTE, SHORT, INTEGER, LONG, FLOAT, DOUBLE	type.validateWith(minValue, maxValue) type.validateWith(minValue)
STRING	STRING.validateWith(String... elements) STRING.validateWith(boolean ignoreCase, final boolean indexAccess, String... elements) Multiple qualified values are separated by spaces. ignoreCase: whether to ignore case or not; indexAccess: allow the use of indexes instead of specific values (0 represents the first qualified value...).
FILE	FILE.validateWith(boolean checkIsExists, boolean checkIsFile, boolean checkIsDirectory) checkIsExists: The file path must exist. checkIsFile: The file path cannot be a folder; checkIsDirectory: The file path must be a folder

- CommandItem item = item.setFormat(String format)

  Set format for command item.

- CommandItem item = item.setDescription(java.lang.String description)

  Set descriptions for command item.

```
group001.register(FILE.VALUE, "--compress", "-c")
        .validateWith(FILE.validateWith(true, true))
        .setDescription("Compression using parallel-bgzip (supported by CLM algorithm).");
group001.register(FILE.VALUE, "--decompress", "-d")
        .validateWith(FILE.validateWith(true, true))
        .setDescription("Decompress or recompress partial (or full) bgzip file.");
group001.register(FILE.ARRAY, "--concat")
        .arity(-1)
        .validateWith(FILE.validateWith(true, true))
        .setDescription("Concatenate multiple files.");
group001.register(FILE.VALUE, "--md5")
        .validateWith(FILE.validateWith(true, true))
        .setDescription("Calculate a message-digest fingerprint (checksum) for input file.");
group001.register(FILE.VALUE, "--md5-decompress", "--md5-d")
        .validateWith(FILE.validateWith(true, true))
        .setDescription("Calculate a message-digest fingerprint (checksum) for decompressed file.");

group002.register(IType.NONE, "--help", "-help", "-h")
```

```
        .addOptions(HELP, HIDDEN);
group002.register(FILE.VALUE, "--output", "-o")
        .setDescription("Set the output file.");
group002.register(LONG.RANGE, "--range", "-r")
        .validateWith(LONG.validateWith(0L))
        .setDescription("Set the range of the file pointer.");
group002.register(INTEGER.VALUE, "--level", "-l")
        .defaultTo("5")
        .validateWith(INTEGER.validateWith(0, 9))
        .setDescription("Compression level to use for bgzip compression.");
group002.register(INTEGER.VALUE, "--threads", "-t")
        .defaultTo("4")
        .validateWith(INTEGER.validateWith(1))
        .setDescription("Set the number of threads for parallel-bgzip compression.");
```

Add Command Rule

CommandParser adds a command rule by three following ways. For the meaning of different types of command rule is shown in Command Rule Type.

- parser.addRule(String ruleType, String... commands)

  Add command rule. Available types: SYMBIOSIS or PRECONDITION.

- parser.addRule(String ruleType, int conditionalValue, String... commands)

  Add command rule. Available types: AT_MOST, AT_LEAST, EQUAL, or MUTUAL_EXCLUSION.

- parser.addRule(CommandRule rule)

  Add command rule.

```
parser.addRule(EQUAL, 1, "--md5-decompress", "--md5", "--compress", "--decompress", "--concat");
parser.addRule(MUTUAL_EXCLUSION, 1, "--md5", "--output", "--level", "--range", "--threads");
parser.addRule(MUTUAL_EXCLUSION, 1, "--md5-decompress", "--output", "--range", "--level", "--threads");
parser.addRule(MUTUAL_EXCLUSION, 1, "--concat", "--range", "--level", "--threads");
```

Formatted Parser

After formatting the parser according to the standard template, the parser can be dragged directly into the graphical design interface for editing and management if the parser meets the following requirements:

- The class file is complete, and can be compiled independently;
- Retain field `package ${path}` ;
- The Class Access Modifier is `public` ;
- Contain static method `public static CommandParser getParser()` .

CommandParserDesigner helps to compile the source code to a class file dynamically and obtains the parser object by `.getParser()` . Finally, the graphical interface designer will reproduces the parser based on the class member's information of the parser object.

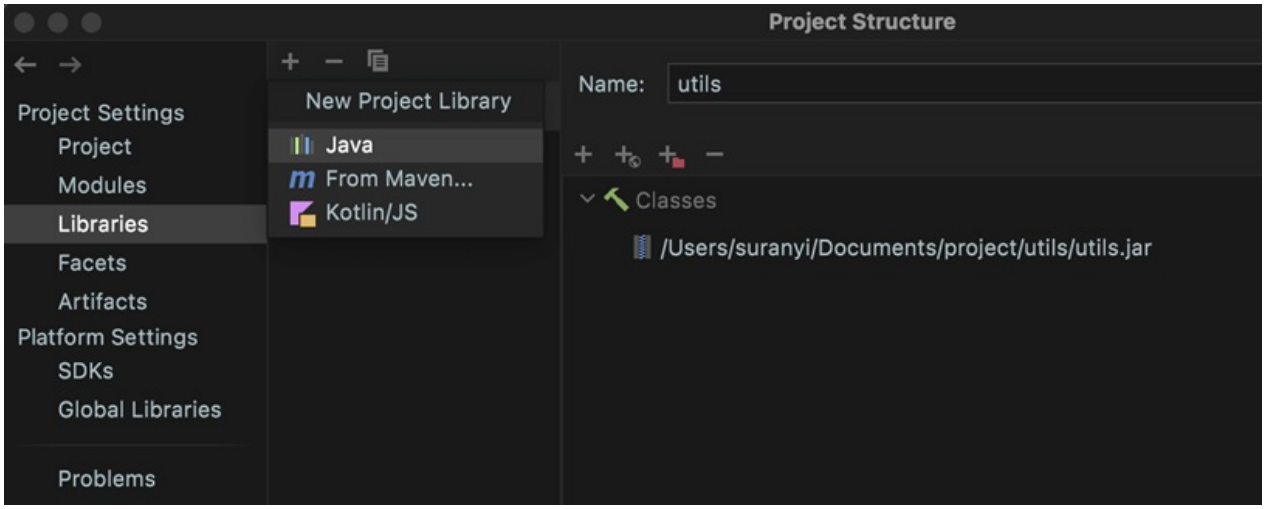
```
package edu.sysu.pmglab.bgztools;

import edu.sysu.pmglab.commandParser.CommandGroup;
import edu.sysu.pmglab.commandParser.CommandOption;
import edu.sysu.pmglab.commandParser.CommandOptions;
import edu.sysu.pmglab.commandParser.CommandParser;
import edu.sysu.pmglab.commandParser.types.FILE;
import edu.sysu.pmglab.commandParser.types.INTEGER;
import edu.sysu.pmglab.commandParser.types.IType;
import edu.sysu.pmglab.commandParser.types.LONG;
import edu.sysu.pmglab.commandParser.usage.DefaultStyleUsage;
import edu.sysu.pmglab.container.File;

import java.io.IOException;
```

```
import static edu.sysu.pmglab.commandParser.CommandItem.HELP;
import static edu.sysu.pmglab.commandParser.CommandItem.HIDDEN;
import static edu.sysu.pmglab.commandParser.CommandRule.EQUAL;
import static edu.sysu.pmglab.commandParser.CommandRule.MUTUAL_EXCLUSION;

public class BGZToolkitParser {
    /**
     * build by: CommandParser-1.1
     * time: 2022-05-31 18:06:01
     */
    private static final CommandParser PARSER = new CommandParser(false);

    private final CommandOptions options;
    public final CommandOption<File> compress;
    public final CommandOption<File> decompress;
    public final CommandOption<File[]> concat;
    public final CommandOption<File> md5;
    public final CommandOption<File> md5Decompress;
    public final CommandOption<?> help;
    public final CommandOption<File> output;
    public final CommandOption<long[]> range;
    public final CommandOption<Integer> level;
    public final CommandOption<Integer> threads;

    BGZToolkitParser(String... args) {
        this.options = PARSER.parse(args);
        this.compress = new CommandOption<>("--compress", this.options);
        this.decompress = new CommandOption<>("--decompress", this.options);
        this.concat = new CommandOption<>("--concat", this.options);
        this.md5 = new CommandOption<>("--md5", this.options);
        this.md5Decompress = new CommandOption<>("--md5-decompress", this.options);
        this.help = new CommandOption<>("--help", this.options);
        this.output = new CommandOption<>("--output", this.options);
        this.range = new CommandOption<>("--range", this.options);
        this.level = new CommandOption<>("--level", this.options);
        this.threads = new CommandOption<>("--threads", this.options);
    }

    public static BGZToolkitParser parse(String... args) {
        return new BGZToolkitParser(args);
    }

    public static BGZToolkitParser parse(File argsFile) throws IOException {
        return new BGZToolkitParser(CommandParser.readFromFile(argsFile));
    }

    /**
     * Get CommandParser
     */
    public static CommandParser getParser() {
        return PARSER;
    }

    /**
     * Get the usage of CommandParser
     */
    public static String usage() {
        return PARSER.toString();
    }

    /**
     * Get CommandOptions
     */
    public CommandOptions getOptions() {
        return this.options;
    }

    static {
        PARSER.setProgramName("<mode>");
        PARSER.offset(0);
        PARSER.debug(false);
        PARSER.usingAt(true);
        PARSER.setMaxMatchedNum(-1);
        PARSER.setAutoHelp(true);
        PARSER.setUsageStyle(DefaultStyleUsage.UNIX_TYPE_1);

        CommandGroup group001 = PARSER.addCommandGroup("Mode");
```

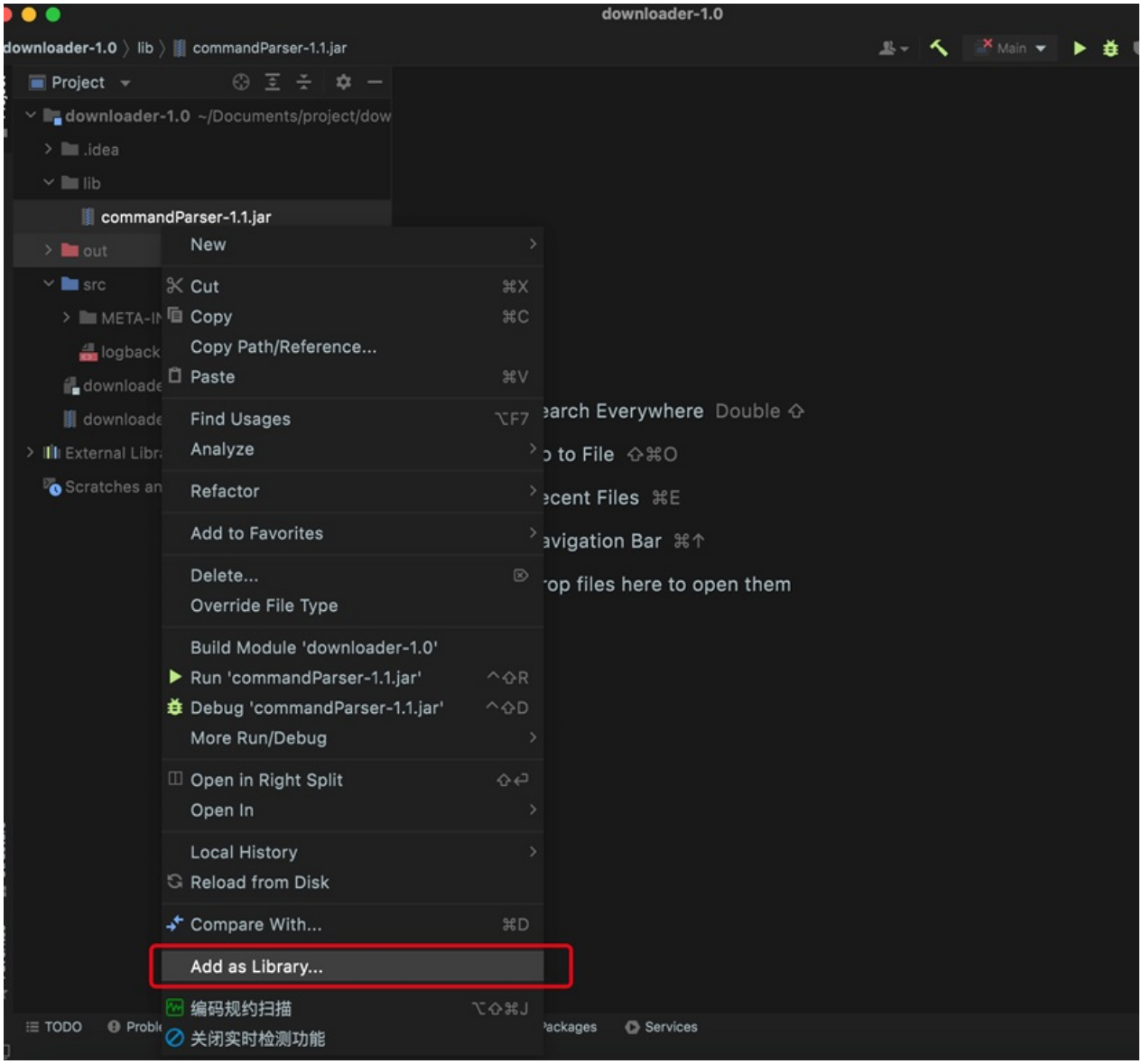
```
        group001.register(FILE.VALUE, "--compress", "-c")
                .validateWith(FILE.validateWith(true, true))
                .setDescription("Compression using parallel-bgzip (supported by CLM algorithm).");
        group001.register(FILE.VALUE, "--decompress", "-d")
                .validateWith(FILE.validateWith(true, true))
                .setDescription("Decompress or recompress partial (or full) bgzip file.");
        group001.register(FILE.ARRAY, "--concat")
                .arity(-1)
                .validateWith(FILE.validateWith(true, true))
                .setDescription("Concatenate multiple files.");
        group001.register(FILE.VALUE, "--md5")
                .validateWith(FILE.validateWith(true, true))
                .setDescription("Calculate a message-digest fingerprint (checksum) for input file.");
        group001.register(FILE.VALUE, "--md5-decompress", "--md5-d")
                .validateWith(FILE.validateWith(true, true))
                .setDescription("Calculate a message-digest fingerprint (checksum) for decompressed file.");

        CommandGroup group002 = PARSER.addCommandGroup("Options");
        group002.register(IType.NONE, "--help", "-help", "-h")
                .addOptions(HELP, HIDDEN);
        group002.register(FILE.VALUE, "--output", "-o")
                .setDescription("Set the output file.");
        group002.register(LONG.RANGE, "--range", "-r")
                .validateWith(LONG.validateWith(0L))
                .setDescription("Set the range of the file pointer.");
        group002.register(INTEGER.VALUE, "--level", "-l")
                .defaultTo("5")
                .validateWith(INTEGER.validateWith(0, 9))
                .setDescription("Compression level to use for bgzip compression.");
        group002.register(INTEGER.VALUE, "--threads", "-t")
                .defaultTo("4")
                .validateWith(INTEGER.validateWith(1))
                .setDescription("Set the number of threads for parallel-bgzip compression.");

        PARSER.addRule(EQUAL, 1, "--md5-decompress", "--md5", "--compress", "--decompress", "--concat");
        PARSER.addRule(MUTUAL_EXCLUSION, 1, "--md5", "--output", "--level", "--range", "--threads");
        PARSER.addRule(MUTUAL_EXCLUSION, 1, "--md5-decompress", "--output", "--range", "--level", "--threads");
        PARSER.addRule(MUTUAL_EXCLUSION, 1, "--concat", "--range", "--level", "--threads");
    }
}
```

Import CommandParser.jar

Method 1 : Select Libraries in Project Structure and add commandParser-1.1.jar.



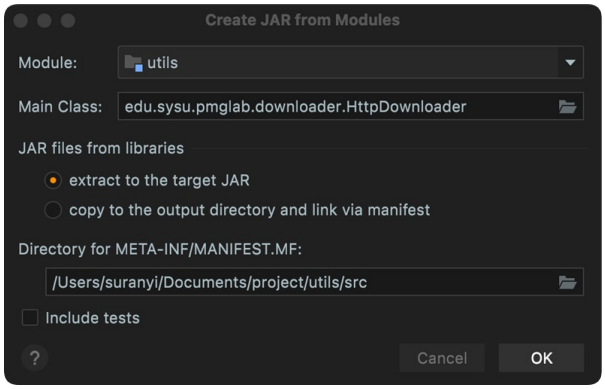
Method 2: Create a lib folder in your project, add the package of commandParser-1.1.jar, and right-click "Add as Library...".

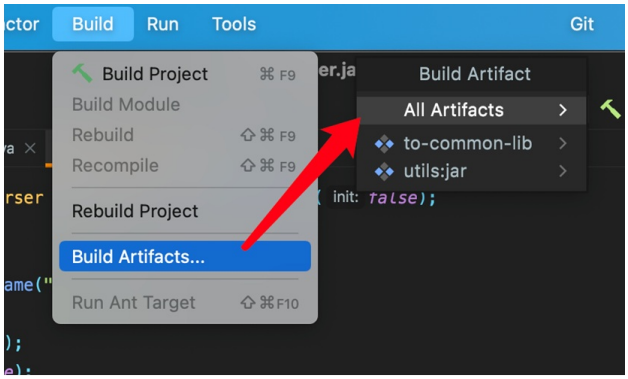


Create Jar Package

The command-line program development will be completed after creating the parser's source file, importing the commandParser-1.1.jar package, and writing the entry function (main class). Finally, package the Java project as a JAR package (take IDEA as an example):

Click: "Project Structure"... > "Artifacts" > "+" > "JAR" > "From modules with dependencies...". The left image is then displayed. Select the location of the entry function in Main Class, and click "OK". Finally, click "Build" > "Build Artifacts" to build a JAR package.

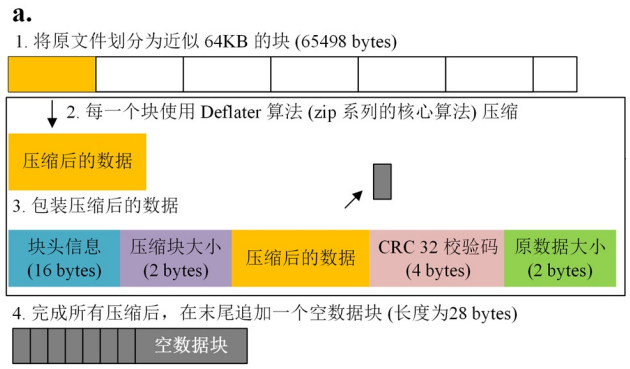




About BGZToolkit

BGZIP is a common compression tool in the field of bioinformatics. BGZIP tool cuts the input file into small blocks (approximately 64KB) and compresses them into a series of small BGZF blocks by Deflater respectively. This principle makes it possible for building indexes against the compressed blocks, which allows quickly retrieving parts of the data without decompressing the entire file.

We developped this project with pure Java script for manipulating BGZ files, including parallel compression, decompression, file concat, md5 check and so on.



Type	URL
Software	http://pmglab.top/commandParser/bgzip/bgzip.jar
Source	http://pmglab.top/commandParser/bgzip/BGZToolkit.java
Parser File	http://pmglab.top/commandParser/bgzip/BGZToolkitParser.java

API

The main API methods for BGZIP can be found in `edu.sysu.pmglab.bgztools.BGZToolkit`, which has been integrated in the commandParser package), including:

- **compression:** BGZToolkit.Compress.instance(File inputFile, File outputFile)
  - Set the number of threads for parallel-bgzip compression: .setThreads(int nThreads)
  - Set the range of the file pointer: .limit(long start, long end)
  - Set compression level for compression: .setCompressionLevel(int compressionLevel)
- **decompression:** BGZToolkit.Extract.instance(File inputFile, File outputFile)
  - Set the number of threads for parallel-bgzip decompression: .setThreads(int nThreads)
  - Set the range of the file pointer: .limit(long start, long end)
  - Set the output format (compressed or not): .setOutputParam(BGZOutputParam outputParam)
- **md5 check:** BGZToolkit.MD5.instance(File inputFile)
  - Calculate md5 for decompressed file or not: .setDecompression(boolean decompression)
- **concat:** BGZToolkit.Concat.instance(File[] inputFiles, File outputFile)

When the instantiation task is completed, submit the corresponding task by `.submit()`.

Design Parser

The BGZIP toolset has 5 independent modes of operation, including Compression, Decompression, Concat, MD5 Check, MD5 Check (decompressed file), and there are 5 parameters in total, including the number of threads, file pointer range, output file, compression level, and whether to print the program log. The creation steps are as follows:

Step1: Create command group： Mode and Options;

Step2: Create command items for `Mode` group: --compress; --decompress; --concat; --md5; --md5-decompress;

Step3: Create command items for `Options` group: --output; --range; --level; --threads;

Step4: Create command rules:

- command item --compress, --decompress, --concat, --md5, -- md5-decompress must be passed one;
- --output, --range, --level, --threads cannot be used when the --md5 or --md5-decompress mode is actived;
- --range, --level, --threads cannot be used when the --concat mode is actived;

Step5: Set program name to: `<mode>` ;

Step6: Export to Java Script Builder With Options Format.

Design Main Function

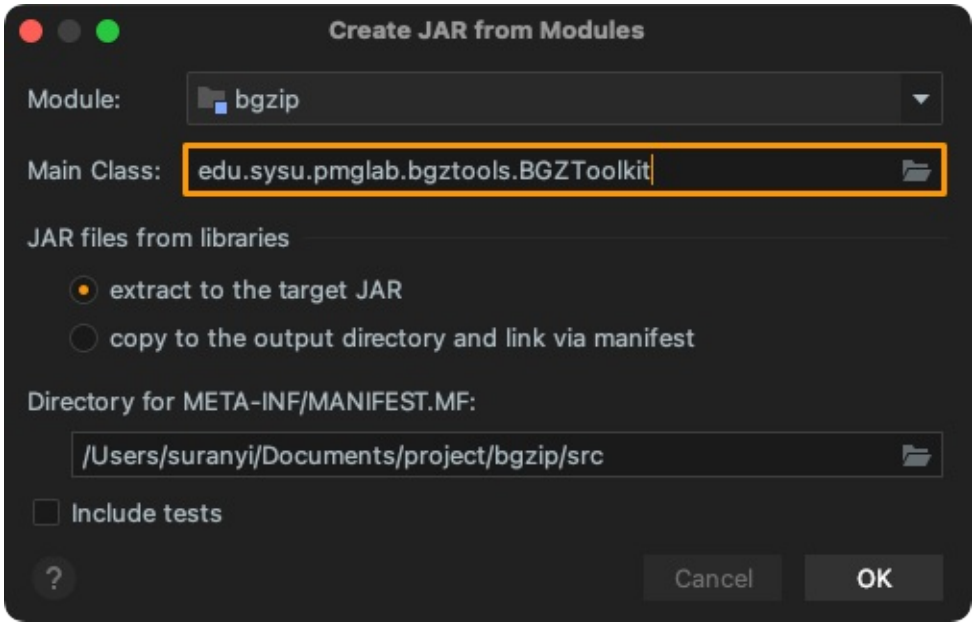
Use CommandParser to bridge input parameters with business logic. See: BGZToolkit.java.

```
public static void main(String[] args) {
    try {
        BGZToolkitParser options = BGZToolkitParser.parse(args);
        if (options.getOptions().isHelp()) {
            System.out.println(BGZToolkitParser.usage());
            return;
        }

        if (options.compress.isPassedIn) {
            // 压缩文件
            long startTime = System.currentTimeMillis();
            File outputFile = options.output.isPassedIn ? options.output.value : options.compress.value.addExtension(
".gz");
            Compress task = Compress.instance(options.compress.value, outputFile)
                    .setThreads(options.threads.value)
                    .setCompressionLevel(options.level.value)
                    .setPrintLog(true);

            if (options.range.isPassedIn) {
                task.limit(options.range.value[0], options.range.value[1]);
            }
            task.submit();
            logger.info("Compression completed. Total time: {} s; File size: {}", (System.currentTimeMillis() - start
Time) / 1000d, outputFile.formatSize(3));
            return;
        }

        if (options.decompress.isPassedIn) {
            long startTime = System.currentTimeMillis();

            // 解压文件
            File outputFile;
            Extract task;
            if (options.output.isPassedIn) {
                outputFile = options.output.value;

                task = Extract.instance(options.decompress.value, outputFile)
                        .setThreads(options.threads.value);

                if (options.level.isPassedIn || options.output.value.withExtension(".gz")) {
                    task.setOutputParam(new BGZOutputParam(options.level.value));
                }
            } else {
                if (!options.level.isPassedIn) {
                    // 没有传入 level 时, 认为是解压文件
                    outputFile = options.decompress.value.changeExtension("", ".gz");
                    if (outputFile.equals(options.decompress.value)) {
                        logger.error("can't remove an extension from {} -- please rename", options.decompress.value);
                        return;
                    }
                } else {
```

```
                    // 传入了 level, 认为是提取文件子集
                    if (options.decompress.value.withExtension(".gz")) {
                        logger.error("missing required positional argument: --output");
                        return;
                    } else {
                        outputFile = options.decompress.value.addExtension(".gz");
                    }
                }

                task = Extract.instance(options.decompress.value, outputFile)
                        .setThreads(options.threads.value);
            }

            if (options.range.isPassedIn) {
                task.limit(options.range.value[0], options.range.value[1]);
            }

            task.submit();

            logger.info("Decompression completed. Total time: {} s; File size: {}", (System.currentTimeMillis() - sta
rtTime) / 1000d, outputFile.formatSize(3));
            return;
        }

        if (options.concat.isPassedIn) {
            // 连接文件
            long startTime = System.currentTimeMillis();
            if (!options.output.isPassedIn) {
                logger.error("missing required positional argument: --output");
                return;
            }

            Concat.instance(options.concat.value, options.output.value).submit();

            logger.info("Concat completed. Total time: {} s; File size: {}", (System.currentTimeMillis() - startTime)
/ 1000d, options.output.value.formatSize(3));
            return;
        }


        // md5 校验文件
        if (options.md5.isPassedIn) {
            MD5 task = MD5.instance(options.md5.value);
            task.setDecompression(false);
            System.out.println("MD5 (" + options.md5.value + ") = " + task.submit());
            return;
        }

        if (options.md5Decompress.isPassedIn) {
            MD5 task = MD5.instance(options.md5Decompress.value);
            task.setDecompression(true);
            System.out.println("MD5 (" + options.md5Decompress.value + ") = " + task.submit());
            return;
        }
    } catch (Exception | Error e) {
        logger.error("{}", e.getMessage(), e);
    }
}
```

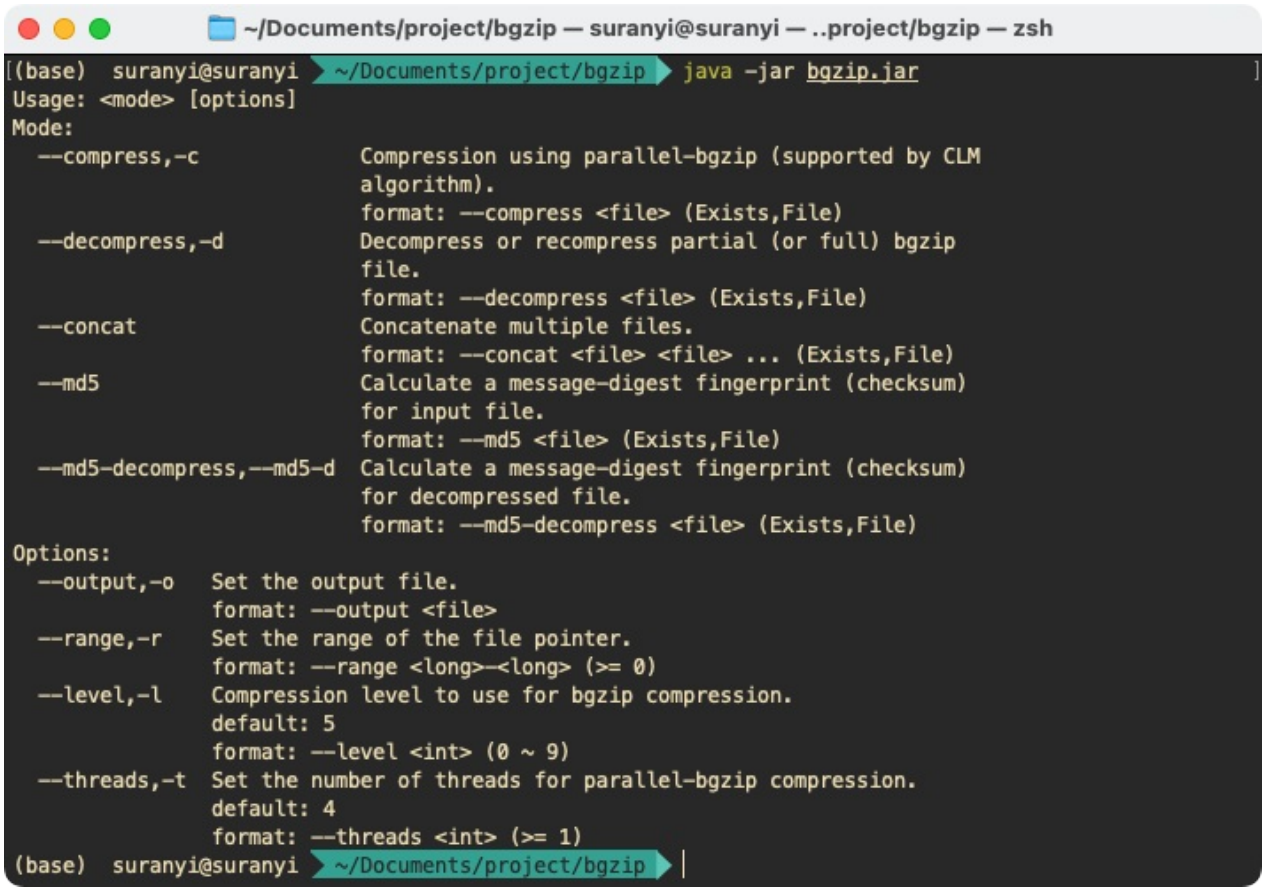
Create Jar Package

Click: "Project Structure..."> "Artifacts" > "+" > "JAR" > "From modules with dependencies...". Select the entry function in Main Class, and packaged as bgzip.jar:



Go to the folder where `bgzip.jar` is located, and enter the command on the console to display the document:

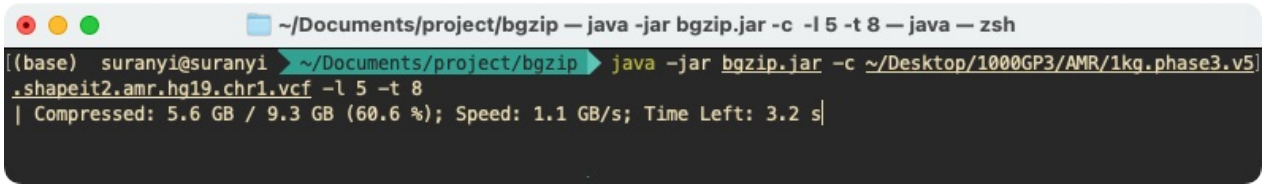
```
java -jar bgzip.jar
```



Example

1. Compress a single file with multiple threads

```
# Download from: http://pmglab.top/genotypes/#/
java -jar bgzip.jar -c ~/Desktop/1000GP3/AMR/1kg.phase3.v5.shapeit2.amr.hg19.chr1.vcf -l 5 -t 8
```



2. Decompress a file and recompress into BGZIP format

```
# Extract the first 1 GB of content
java -jar bgzip.jar -d ~/Desktop/1000GP3/AMR/1kg.phase3.v5.shapeit2.amr.hg19.chr1.vcf.gz \
--range 0-1073741824 -o ~/Desktop/subFile.vcf.gz

# Decompress the above file
java -jar bgzip.jar -d  ~/Desktop/subFile.vcf.gz
```

3. Verify the MD5 code of a file

```
# Result: 29ccebe9d2748328f6cc8b16c6563261
java -jar bgzip.jar --md5 ~/Desktop/subFile.vcf

# Result: 29ccebe9d2748328f6cc8b16c6563261
java -jar bgzip.jar --md5-d ~/Desktop/subFile.vcf.gz
```

4. Connect multiple files

```
# Split a file into 2 files
java -jar bgzip.jar -d ~/Desktop/1000GP3/AMR/1kg.phase3.v5.shapeit2.amr.hg19.chr1.vcf.gz \
--range 0-536870912 -o ~/Desktop/subFile1.vcf.gz
java -jar bgzip.jar -d ~/Desktop/1000GP3/AMR/1kg.phase3.v5.shapeit2.amr.hg19.chr1.vcf.gz \
--range 536870912-1073741824 -o ~/Desktop/subFile2.vcf.gz

# Concatenate two subfiles
java -jar bgzip.jar --concat ~/Desktop/subFile1.vcf.gz ~/Desktop/subFile2.vcf.gz -o ~/Desktop/subFile.merge.vcf.gz

# Check md5, result: 29ccebe9d2748328f6cc8b16c6563261
java -jar bgzip.jar --md5-d  ~/Desktop/subFile.merge.vcf.gz
```

About HttpDownloader

Researches on bioinformatics and medical informatics require a large amount of public resources. However, due to their huge data volume (e.g., dbNSFP database reaches over 30 GB, gnomAD database reaches over 1 TB), users are always suffering from many problems (e.g., redirection jump, speed limit in the cloud, breakpoint transfer, etc.) when downloading these resources. In order to simplify the process of downloading and updating these databases, and integrate this process as an API tool into other public projects (mainly for Java development) effectively, we have developed the HttpDownloader. In addition, support for FTP downloads is being planned.

HttpDownloader is a simple Http downloader based on the Java platform, designed to enhance the download function for basic resources. HttpDownloader now supports for: breakpoint transfer, URL redirection based on the Location of the response header, parallel download (requires the server to support chunked or get 206 response codes after sending segmented requests), dynamically getting file size for resource downloads, proxy setting.

Type	URL
Software	http://pmglab.top/commandParser/downloader/downloader-1.0.jar
Source	http://pmglab.top/commandParser/downloader/HttpDownloader.java
Parser File	http://pmglab.top/commandParser/downloader/HttpDownloaderParser.java

API

The main API methods for HttpDownloader can be found in `edu.sysu.pmglab.downloader.HttpDownloader`, which has been integrated in the commandParser package, including:

- **download:** HttpDownloader.instance(String url)
  - Set the number of threads for parallel download: .setThreads(int nThreads)
  - Set the output file name: .setOutputFile(File outputFile) 和 .setOutputFile(String outputFileName)
  - Set the temporary path for cache data: .setTempDir(File tempDir) 和 .setTempDir(File tempDirName)
  - Set the proxy: .setProxy(String host, String port) 和 .setProxy(String hostPort)
  - Set the maximum waitting time (unit: second): .setTimeOut(int timeOut)
  - Clearing cached data: .clean(boolean clean)

When the instantiation task is complete, submit the corresponding task by `.download()`.

Design Parser

The first parameter of input command will be identified as the URL address by HttpDownloader, and the following parameters will be parsed as options (set offset=1). The steps of parser creation are described as following:

Step1: Create command group：Options;

Step2: Create command items for `Options` group: --output; --temp-dir, --threads, --overwrite, --proxy, --time-out, --no-auto-retry;

Step3: Set the program name: `<mode>`, set offset as `1`;

Step4: Set the Usage Style as `Unix_Style_3` and set the subtitle as follows:

```
About: Download file from an URL. The program supports breakpoints, parallel
       downloads, proxy IP settings and is suitable for downloading a wide
       range of http and thunder type urls.
```

Step5: Export to Java Script Builder With Options Format.

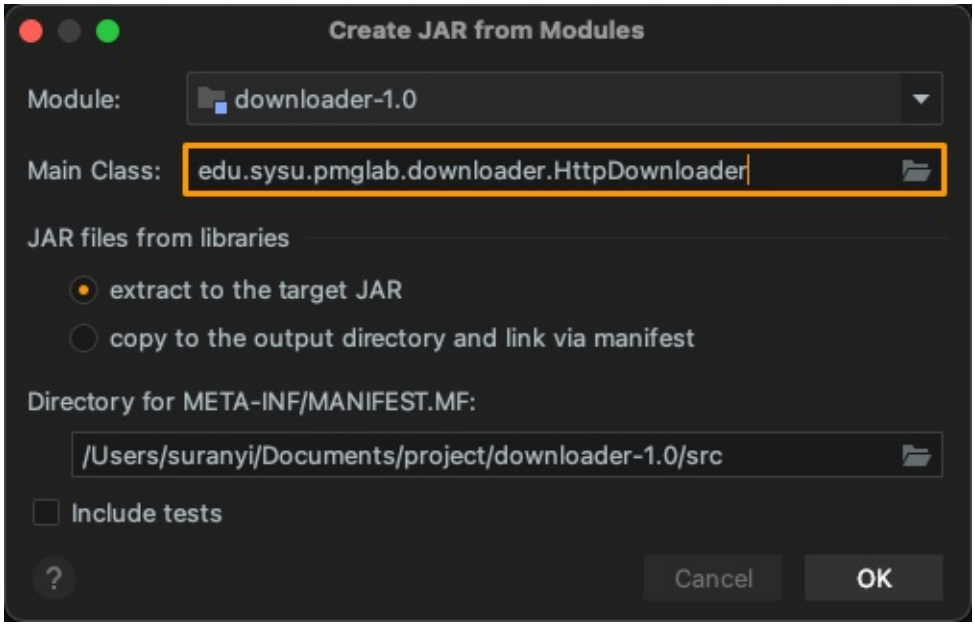
Design Main Function

Use CommandParser to bridge input parameters with business logic. See: HttpDownloader.java.

```
public static void main(String[] args) {
    try {
        if (args.length == 0) {
            System.out.println(HttpDownloaderParser.getParser());
            return;
        }

        HttpDownloaderParser parser = HttpDownloaderParser.parse(args);
        if (parser.help.isPassedIn) {
            System.out.println(HttpDownloaderParser.getParser());
            return;
        }

        do {
            try {
                HttpDownloader.instance(args[0])
                        .setOutputFile(parser.output.value)
                        .setThreads(parser.threads.value)
                        .setPrintLog(true)
                        .setTempDir(parser.tempDir.value)
                        .setProxy(parser.proxy.value)
                        .setTimeOut(parser.timeout.value)
                        .clean(parser.overwrite.isPassedIn)
                        .download();
            } catch (IOException e) {
                logger.error("{}", e.getMessage());

                if (!parser.noAutoRetry.isPassedIn) {
                    logger.error("Download failed, resume download in 3 seconds.");

                    Thread.sleep(3000);
                    continue;
                }
            }

            break;
        } while (true);
    } catch (Exception | Error e) {
        logger.error("{}", e.getMessage());
    }
}
```

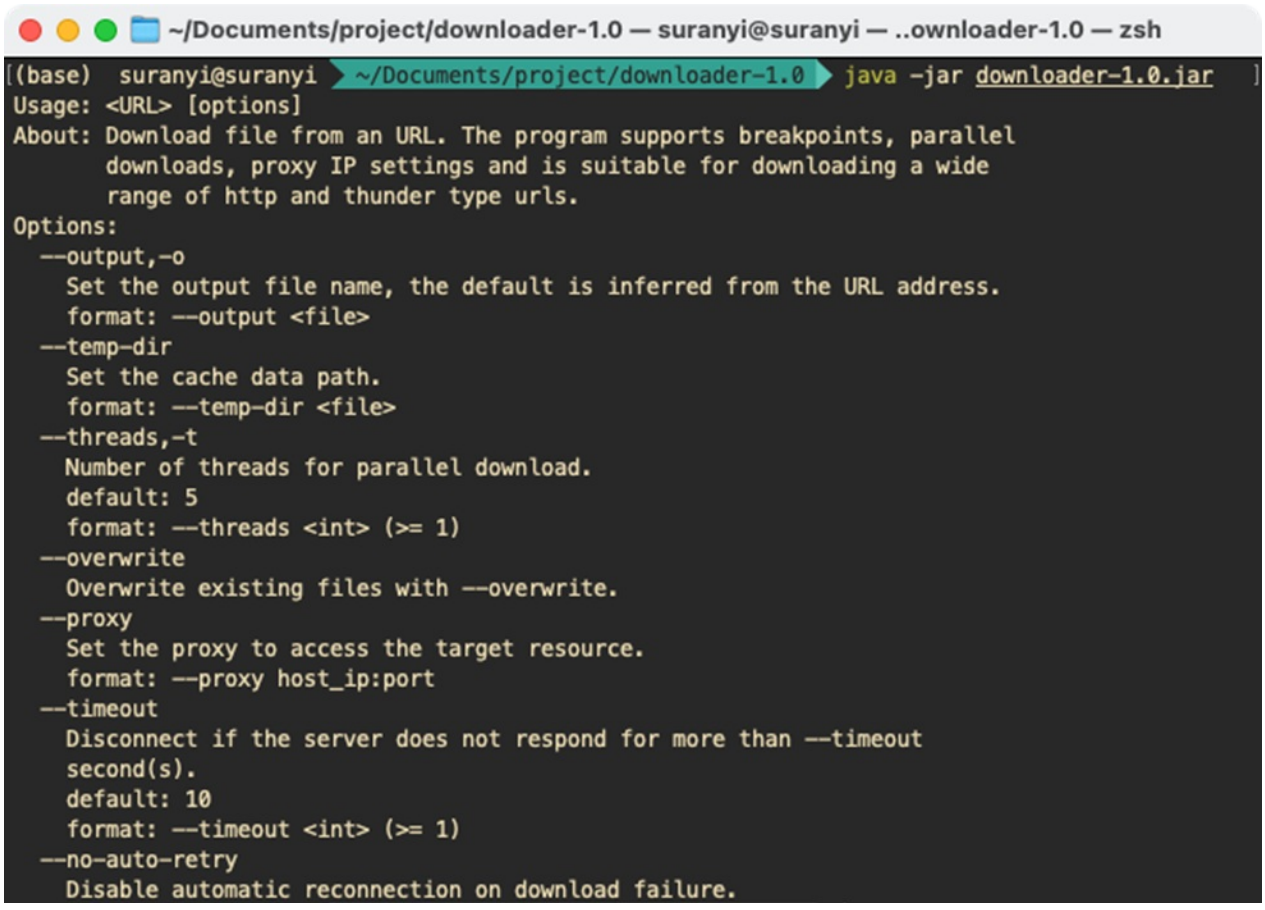
Create Jar Package

Click: "Project Structure..." > "Artifacts" > "+" > "JAR" > "From modules with dependencies...". Select the entry function in Main Class, and packaged as downloader-1.0.jar:



Go to the folder where `downloader-1.0.jar` is located, and enter the command on the console to display the document:

```
java -jar downloader-1.0.jar
```



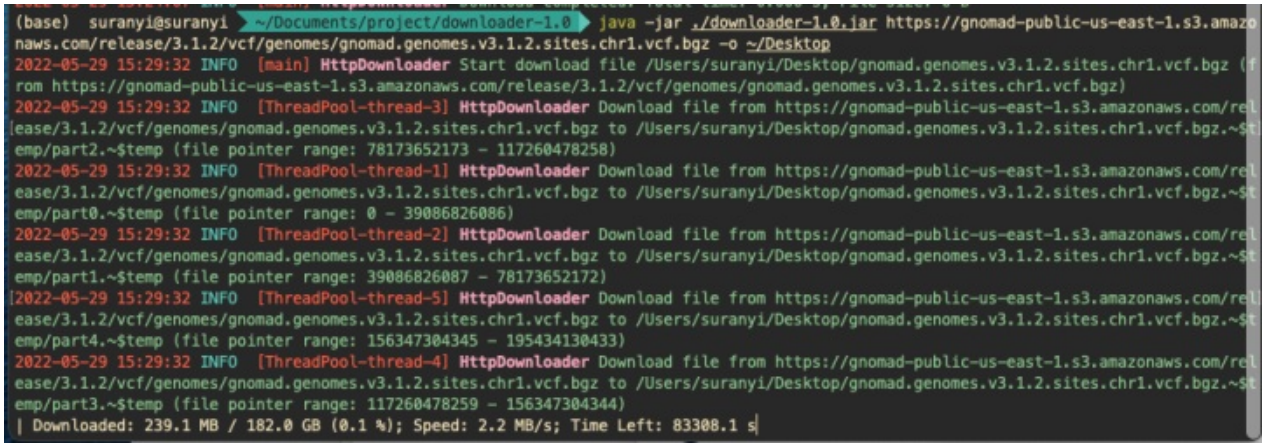
Example

1. Download gnomAD

The Genome Aggregation Database (gnomeAD) is a collaborative genomic mutation frequency database created by national researchers to aggregate and coordinate large-scale sequencing projects at different levels, including whole-exome and whole-genome data, for a wide range of scientific research communities. The database currently includes 125,748 whole-exome data and 15,708 whole-genome data (v2) from different disease research projects and large population sequencing projects. The database includes the previously used 1,000 genomes data, the ESP database and most of the ExAC database.

Amazon's links support parallel downloading and resumable without the need for a proxy.

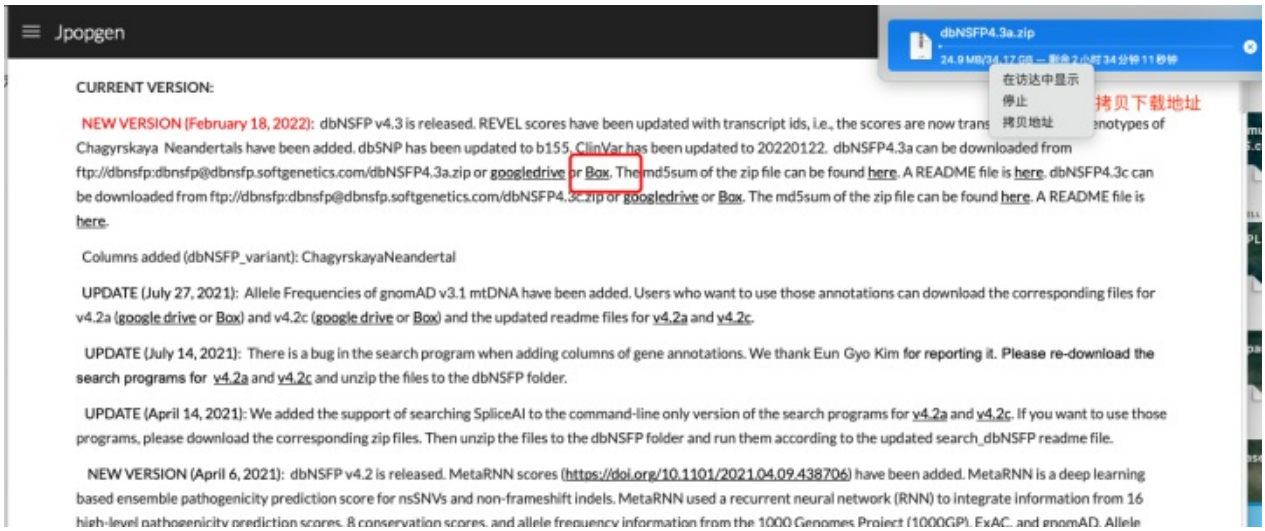

```
java -jar ./downloader-1.0.jar https://gnomad-public-us-east-1.s3.amazonaws.com/release/3.1.2/vcf/genomes/gnomad.genomes.v3.1.2.sites.chr1.vcf.bgz -o ~/Desktop
```



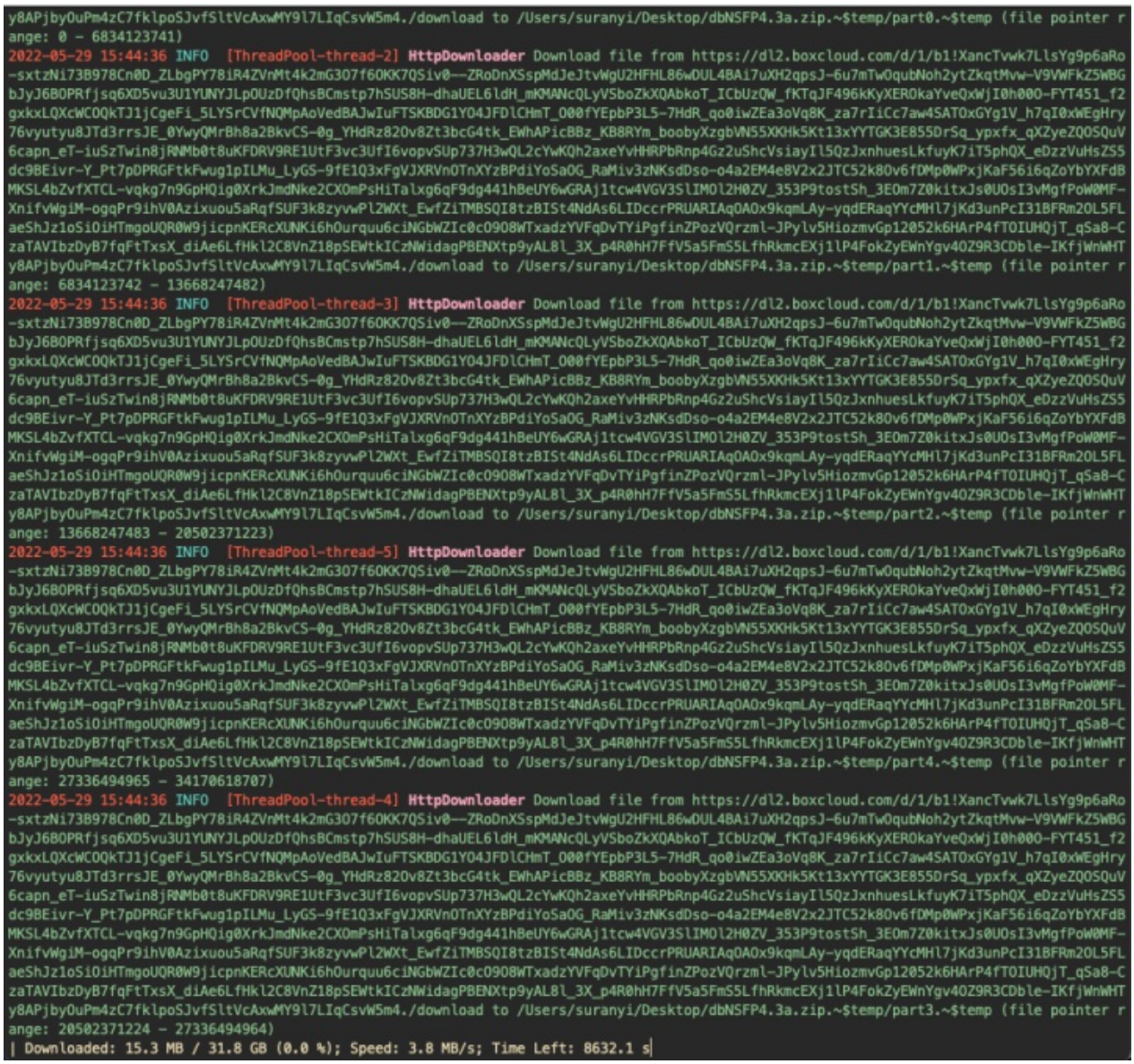
2. Download dbNSFP

dbNSFP is a database developed for functional prediction and annotation of all potential non-synonymous single-nucleotide variants (nsSNVs) in the human genome. Its current version is based on the Gencode release 29 / Ensembl version 94 and includes a total of 84,013,490 nsSNVs and ssSNVs (splicing-site SNVs). It compiles prediction scores from 38 prediction algorithms (SIFT, SIFT4G,

Polyphen2-HDIV, Polyphen2-HVAR, LRT, MutationTaster2, MutationAssessor, FATHMM, MetaSVM, MetaLR, MetaRNN, CADD, CADD_hg19, VEST4, PROVEAN, FATHMM-MKL coding, FATHMM-XF coding, fitCons x 4, LINSIGHT, DANN, GenoCanyon, Eigen, Eigen-PC, M-CAP, REVEL, MutPred, MVP, MPC, PrimateAI, GEOGEN2, BayesDel_addAF, BayesDel_noAF, ClinPred, LIST-S2, ALoFT), 9 conservation scores (PhyloP x 3, phastCons x 3, GERP++, SiPhy and bStatistic) and other related information including allele frequencies observed in the 1000 Genomes Project phase 3 data, UK10K cohorts data, ExAC consortium data, gnomAD data and the NHLBI Exome Sequencing Project ESP6500 data, various gene IDs from different databases, functional descriptions of genes, gene expression and gene interaction information, etc.




```
java -jar ./downloader-1.0.jar https://dl2.boxcloud.com/d/1/b1\!XancTvwk7LlsYg9p6aRo-sxtzNi73B978Cn0D_ZLbgPY78iR4ZVnM
t4k2mG3O7f6OKK7QSiv0--ZRoDnXSspMdJeJtvWgU2HFHL86wDUL4BAi7uXH2qpsJ-6u7mTwOqubNoh2ytZkqtMvw-V9VWFkZ5WBGbJyJ6BOPRfjsq6XD
5vu3U1YUNYJLpOUzDfQhsBCmstp7hSUS8H-dhaUEL6ldH_mKMANcQLyVSboZkXQAbkoT_ICbUzQW_fKTqJF496kKyXERORkaYveQxWjI0h00O-FYT451_f
2gxkxLQXcWCOQkTJ1jCgeFi_5LYSrCVfNQMpAoVedBAJwIuFTSKBDG1YO4JFDlCHmT_O00fYEpbP3L5-7HdR_qo0iwZEa3oVq8K_za7rIiCc7aw4SATOx
GYg1V_h7qI0xWEgHry76vyutyu8JTd3rrsJE_0YwyQMrBh8a2BkvCS-0g_YHdRz82Ov8Zt3bcG4tk_EWhAPicBBz_KB8RYm_boobyXzgbVN55XKHk5Kt1
3xYYTGK3E855DrSq_ypxfx_qXZyeZQOSQuV6capn_eT-iuSzTwin8jRNMb0t8uKFDRV9RE1UtF3vc3UfI6vopvSUp737H3wQL2cYwKQh2axeYvHHRPbRn
p4Gz2uShcVsiayIl5QzJxnhuesLkfuyK7iT5phQX_eDzzVuHsZS5dc9BEivr-Y_Pt7pDPRGFtkFwug1pILMu_LyGS-9fE1Q3xFgVJXRVnOTnXYzBPdiYo
SaOG_RaMiv3zNKsdDso-o4a2EM4e8V2x2JTC52k8Ov6fDMp0WPxjKaF56i6qZoYbYXFdBMKSL4bZvfXTCL-vqkg7n9GpHQig0XrkJmdNke2CXOmPsHiTa
lxg6qF9dg441hBeUY6wGRAj1tcw4VGV3SlIMOl2H0ZV_353P9tostSh_3EOm7Z0kitxJs0UOsI3vMgfPoW0MF-XnifvWgiM-ogqPr9ihV0Azixuou5aRq
fSUF3k8zyvwPl2WXt_EwfZiTMBSQI8tzBISt4NdAs6LIDccrPRUARIAqOAOx9kqmLAy-yqdERaqYYcMHl7jKd3unPcI31BFRm2OL5FLaeShJz1oSiOiHT
mgoUQR0W9jicpnKERcXUNKi6hOurquu6ciNGbWZIc0cO9O8WTxadzYVFqDvTYiPgfinZPozVQrzml-JPylv5HiozmvGp12052k6HArP4fTOIUHQjT_qSa
8-CzaTAVIbzDyB7fqFtTxsX_diAe6LfHkl2C8VnZ18pSEWtkICzNWidagPBENXtp9yAL8l_3X_p4R0hH7FfV5a5FmS5LfhRkmcEXj1lP4FokZyEWnYgv4
OZ9R3CDble-IKfjWnWHTy8APjbyOuPm4zC7fklpoSJvfSltVcAxwMY9l7LIqCsvW5m4./download -o ~/Desktop/dbNSFP4.3a.zip
```


y8APjbyOuPm4zC7fklpoSJvfSltVcAxwMY9l7LIqCsvW5m4./download to /Users/suranyi/Desktop/dbNSFP4.3a.zip.~$temp/part0.~$temp (file pointer range: 0 - 6834123741)
2022-05-29 15:44:36 INFO [ThreadPool-thread-2] HttpDownloader Download file from https://dl2.boxcloud.com/d/1/b1!XancTvwk7LlsYg9p6aRo-sxtzNi73B978Cn0D_ZLbgPY78iR4ZVnMt4k2mG3O7f6OKK7QSiv0--ZRoDnXSspMdJeJtvWgU2HFHL86wDUL4BAi7uXH2qpsJ-6u7mTwOqubNoh2ytZkqtMvw-V9VWFkZ5WBGbJyJ6BOPRfjsq6XD5vu3U1YUNYJLpOUzDfQhsBCmstp7hSUS8H-dhaUEL6ldH_mKMANcQLyVSboZkXQAbkoT_ICbUzQW_fKTqJF496kKyXERORkaYveQxWjI0h00O-FYT451_f2gxkxLQXcWCOQkTJ1jCgeFi_5LYSrCVfNQMpAoVedBAJwIuFTSKBDG1YO4JFDlCHmT_O00fYEpbP3L5-7HdR_qo0iwZEa3oVq8K_za7rIiCc7aw4SATOxGYg1V_h7qI0xWEgHry76vyutyu8JTd3rrsJE_0YwyQMrBh8a2BkvCS-0g_YHdRz82Ov8Zt3bcG4tk_EWhAPicBBz_KB8RYm_boobyXzgbVN55XKHk5Kt13xYYTGK3E855DrSq_ypxfx_qXZyeZQOSQuV6capn_eT-iuSzTwin8jRNMb0t8uKFDRV9RE1UtF3vc3UfI6vopvSUp737H3wQL2cYwKQh2axeYvHHRPbRnp4Gz2uShcVsiayIl5QzJxnhuesLkfuyK7iT5phQX_eDzzVuHsZS5dc9BEivr-Y_Pt7pDPRGFtkFwug1pILMu_LyGS-9fE1Q3xFgVJXRVnOTnXYzBPdiYoSaOG_RaMiv3zNKsdDso-o4a2EM4e8V2x2JTC52k8Ov6fDMp0WPxjKaF56i6qZoYbYXFdBMKSL4bZvfXTCL-vqkg7n9GpHQig0XrkJmdNke2CXOmPsHiTalxg6qF9dg441hBeUY6wGRAj1tcw4VGV3SlIMOl2H0ZV_353P9tostSh_3EOm7Z0kitxJs0UOsI3vMgfPoW0MF-XnifvWgiM-ogqPr9ihV0Azixuou5aRqfSUF3k8zyvwPl2WXt_EwfZiTMBSQI8tzBISt4NdAs6LIDccrPRUARIAqOAOx9kqmLAy-yqdERaqYYcMHl7jKd3unPcI31BFRm2OL5FLaeShJz1oSiOiHTmgoUQR0W9jicpnKERcXUNKi6hOurquu6ciNGbWZIc0cO9O8WTxadzYVFqDvTYiPgfinZPozVQrzml-JPylv5HiozmvGp12052k6HArP4fTOIUHQjT_qSa8-CzaTAVIbzDyB7fqFtTxsX_diAe6LfHkl2C8VnZ18pSEWtkICzNWidagPBENXtp9yAL8l_3X_p4R0hH7FfV5a5FmS5LfhRkmcEXj1lP4FokZyEWnYgv4OZ9R3CDble-IKfjWnWHTy8APjbyOuPm4zC7fklpoSJvfSltVcAxwMY9l7LIqCsvW5m4./download to /Users/suranyi/Desktop/dbNSFP4.3a.zip.~$temp/part1.~$temp (file pointer range: 6834123742 - 13668247482)
2022-05-29 15:44:36 INFO [ThreadPool-thread-3] HttpDownloader Download file from https://dl2.boxcloud.com/d/1/b1!XancTvwk7LlsYg9p6aRo-sxtzNi73B978Cn0D_ZLbgPY78iR4ZVnMt4k2mG3O7f6OKK7QSiv0--ZRoDnXSspMdJeJtvWgU2HFHL86wDUL4BAi7uXH2qpsJ-6u7mTwOqubNoh2ytZkqtMvw-V9VWFkZ5WBGbJyJ6BOPRfjsq6XD5vu3U1YUNYJLpOUzDfQhsBCmstp7hSUS8H-dhaUEL6ldH_mKMANcQLyVSboZkXQAbkoT_ICbUzQW_fKTqJF496kKyXERORkaYveQxWjI0h00O-FYT451_f2gxkxLQXcWCOQkTJ1jCgeFi_5LYSrCVfNQMpAoVedBAJwIuFTSKBDG1YO4JFDlCHmT_O00fYEpbP3L5-7HdR_qo0iwZEa3oVq8K_za7rIiCc7aw4SATOxGYg1V_h7qI0xWEgHry76vyutyu8JTd3rrsJE_0YwyQMrBh8a2BkvCS-0g_YHdRz82Ov8Zt3bcG4tk_EWhAPicBBz_KB8RYm_boobyXzgbVN55XKHk5Kt13xYYTGK3E855DrSq_ypxfx_qXZyeZQOSQuV6capn_eT-iuSzTwin8jRNMb0t8uKFDRV9RE1UtF3vc3UfI6vopvSUp737H3wQL2cYwKQh2axeYvHHRPbRnp4Gz2uShcVsiayIl5QzJxnhuesLkfuyK7iT5phQX_eDzzVuHsZS5dc9BEivr-Y_Pt7pDPRGFtkFwug1pILMu_LyGS-9fE1Q3xFgVJXRVnOTnXYzBPdiYoSaOG_RaMiv3zNKsdDso-o4a2EM4e8V2x2JTC52k8Ov6fDMp0WPxjKaF56i6qZoYbYXFdBMKSL4bZvfXTCL-vqkg7n9GpHQig0XrkJmdNke2CXOmPsHiTalxg6qF9dg441hBeUY6wGRAj1tcw4VGV3SlIMOl2H0ZV_353P9tostSh_3EOm7Z0kitxJs0UOsI3vMgfPoW0MF-XnifvWgiM-ogqPr9ihV0Azixuou5aRqfSUF3k8zyvwPl2WXt_EwfZiTMBSQI8tzBISt4NdAs6LIDccrPRUARIAqOAOx9kqmLAy-yqdERaqYYcMHl7jKd3unPcI31BFRm2OL5FLaeShJz1oSiOiHTmgoUQR0W9jicpnKERcXUNKi6hOurquu6ciNGbWZIc0cO9O8WTxadzYVFqDvTYiPgfinZPozVQrzml-JPylv5HiozmvGp12052k6HArP4fTOIUHQjT_qSa8-CzaTAVIbzDyB7fqFtTxsX_diAe6LfHkl2C8VnZ18pSEWtkICzNWidagPBENXtp9yAL8l_3X_p4R0hH7FfV5a5FmS5LfhRkmcEXj1lP4FokZyEWnYgv4OZ9R3CDble-IKfjWnWHTy8APjbyOuPm4zC7fklpoSJvfSltVcAxwMY9l7LIqCsvW5m4./download to /Users/suranyi/Desktop/dbNSFP4.3a.zip.~$temp/part2.~$temp (file pointer range: 13668247483 - 20502371223)
2022-05-29 15:44:36 INFO [ThreadPool-thread-5] HttpDownloader Download file from https://dl2.boxcloud.com/d/1/b1!XancTvwk7LlsYg9p6aRo-sxtzNi73B978Cn0D_ZLbgPY78iR4ZVnMt4k2mG3O7f6OKK7QSiv0--ZRoDnXSspMdJeJtvWgU2HFHL86wDUL4BAi7uXH2qpsJ-6u7mTwOqubNoh2ytZkqtMvw-V9VWFkZ5WBGbJyJ6BOPRfjsq6XD5vu3U1YUNYJLpOUzDfQhsBCmstp7hSUS8H-dhaUEL6ldH_mKMANcQLyVSboZkXQAbkoT_ICbUzQW_fKTqJF496kKyXERORkaYveQxWjI0h00O-FYT451_f2gxkxLQXcWCOQkTJ1jCgeFi_5LYSrCVfNQMpAoVedBAJwIuFTSKBDG1YO4JFDlCHmT_O00fYEpbP3L5-7HdR_qo0iwZEa3oVq8K_za7rIiCc7aw4SATOxGYg1V_h7qI0xWEgHry76vyutyu8JTd3rrsJE_0YwyQMrBh8a2BkvCS-0g_YHdRz82Ov8Zt3bcG4tk_EWhAPicBBz_KB8RYm_boobyXzgbVN55XKHk5Kt13xYYTGK3E855DrSq_ypxfx_qXZyeZQOSQuV6capn_eT-iuSzTwin8jRNMb0t8uKFDRV9RE1UtF3vc3UfI6vopvSUp737H3wQL2cYwKQh2axeYvHHRPbRnp4Gz2uShcVsiayIl5QzJxnhuesLkfuyK7iT5phQX_eDzzVuHsZS5dc9BEivr-Y_Pt7pDPRGFtkFwug1pILMu_LyGS-9fE1Q3xFgVJXRVnOTnXYzBPdiYoSaOG_RaMiv3zNKsdDso-o4a2EM4e8V2x2JTC52k8Ov6fDMp0WPxjKaF56i6qZoYbYXFdBMKSL4bZvfXTCL-vqkg7n9GpHQig0XrkJmdNke2CXOmPsHiTalxg6qF9dg441hBeUY6wGRAj1tcw4VGV3SlIMOl2H0ZV_353P9tostSh_3EOm7Z0kitxJs0UOsI3vMgfPoW0MF-XnifvWgiM-ogqPr9ihV0Azixuou5aRqfSUF3k8zyvwPl2WXt_EwfZiTMBSQI8tzBISt4NdAs6LIDccrPRUARIAqOAOx9kqmLAy-yqdERaqYYcMHl7jKd3unPcI31BFRm2OL5FLaeShJz1oSiOiHTmgoUQR0W9jicpnKERcXUNKi6hOurquu6ciNGbWZIc0cO9O8WTxadzYVFqDvTYiPgfinZPozVQrzml-JPylv5HiozmvGp12052k6HArP4fTOIUHQjT_qSa8-CzaTAVIbzDyB7fqFtTxsX_diAe6LfHkl2C8VnZ18pSEWtkICzNWidagPBENXtp9yAL8l_3X_p4R0hH7FfV5a5FmS5LfhRkmcEXj1lP4FokZyEWnYgv4OZ9R3CDble-IKfjWnWHTy8APjbyOuPm4zC7fklpoSJvfSltVcAxwMY9l7LIqCsvW5m4./download to /Users/suranyi/Desktop/dbNSFP4.3a.zip.~$temp/part4.~$temp (file pointer range: 27336494965 - 34170618707)
2022-05-29 15:44:36 INFO [ThreadPool-thread-4] HttpDownloader Download file from https://dl2.boxcloud.com/d/1/b1!XancTvwk7LlsYg9p6aRo-sxtzNi73B978Cn0D_ZLbgPY78iR4ZVnMt4k2mG3O7f6OKK7QSiv0--ZRoDnXSspMdJeJtvWgU2HFHL86wDUL4BAi7uXH2qpsJ-6u7mTwOqubNoh2ytZkqtMvw-V9VWFkZ5WBGbJyJ6BOPRfjsq6XD5vu3U1YUNYJLpOUzDfQhsBCmstp7hSUS8H-dhaUEL6ldH_mKMANcQLyVSboZkXQAbkoT_ICbUzQW_fKTqJF496kKyXERORkaYveQxWjI0h00O-FYT451_f2gxkxLQXcWCOQkTJ1jCgeFi_5LYSrCVfNQMpAoVedBAJwIuFTSKBDG1YO4JFDlCHmT_O00fYEpbP3L5-7HdR_qo0iwZEa3oVq8K_za7rIiCc7aw4SATOxGYg1V_h7qI0xWEgHry76vyutyu8JTd3rrsJE_0YwyQMrBh8a2BkvCS-0g_YHdRz82Ov8Zt3bcG4tk_EWhAPicBBz_KB8RYm_boobyXzgbVN55XKHk5Kt13xYYTGK3E855DrSq_ypxfx_qXZyeZQOSQuV6capn_eT-iuSzTwin8jRNMb0t8uKFDRV9RE1UtF3vc3UfI6vopvSUp737H3wQL2cYwKQh2axeYvHHRPbRnp4Gz2uShcVsiayIl5QzJxnhuesLkfuyK7iT5phQX_eDzzVuHsZS5dc9BEivr-Y_Pt7pDPRGFtkFwug1pILMu_LyGS-9fE1Q3xFgVJXRVnOTnXYzBPdiYoSaOG_RaMiv3zNKsdDso-o4a2EM4e8V2x2JTC52k8Ov6fDMp0WPxjKaF56i6qZoYbYXFdBMKSL4bZvfXTCL-vqkg7n9GpHQig0XrkJmdNke2CXOmPsHiTalxg6qF9dg441hBeUY6wGRAj1tcw4VGV3SlIMOl2H0ZV_353P9tostSh_3EOm7Z0kitxJs0UOsI3vMgfPoW0MF-XnifvWgiM-ogqPr9ihV0Azixuou5aRqfSUF3k8zyvwPl2WXt_EwfZiTMBSQI8tzBISt4NdAs6LIDccrPRUARIAqOAOx9kqmLAy-yqdERaqYYcMHl7jKd3unPcI31BFRm2OL5FLaeShJz1oSiOiHTmgoUQR0W9jicpnKERcXUNKi6hOurquu6ciNGbWZIc0cO9O8WTxadzYVFqDvTYiPgfinZPozVQrzml-JPylv5HiozmvGp12052k6HArP4fTOIUHQjT_qSa8-CzaTAVIbzDyB7fqFtTxsX_diAe6LfHkl2C8VnZ18pSEWtkICzNWidagPBENXtp9yAL8l_3X_p4R0hH7FfV5a5FmS5LfhRkmcEXj1lP4FokZyEWnYgv4OZ9R3CDble-IKfjWnWHTy8APjbyOuPm4zC7fklpoSJvfSltVcAxwMY9l7LIqCsvW5m4./download to /Users/suranyi/Desktop/dbNSFP4.3a.zip.~$temp/part3.~$temp (file pointer range: 20502371224 - 27336494964)
| Downloaded: 15.3 MB / 31.8 GB (0.0 %); Speed: 3.8 MB/s; Time Left: 8632.1 s