Java Shell Project Documentation

Copyright © 2025 Adnan Mazharuddin Shaikh. All rights reserved.

Trademark: Adnan Mazharuddin Shaikh™

Contact: adnanmazharuddinshaikh@gmail.com

GitHub: 10adnan75

Table of Contents

Introduction

Welcome to your Java Shell! This project is a fully functional, Unix-like shell written in Java. It supports built-in commands, external programs, pipelines, redirection, tab completion, and command history. The code is organized in a clean, object-oriented way, making it easy to understand and extend.

Project Structure

```
codecrafters-shell-java/
|-- src/
|   |-- main/
|   |   |-- java/
|   |   |   |-- core/
|   |   |   |   |-- Main.java
|   |   |   |   |-- CommandHandler.java
|   |   |   |   |-- ExternalCommand.java
|   |   |   |   |-- ShellHistory.java
|   |   |   |   |-- ShellInputHandler.java
|   |   |   |   |-- TabCompleter.java
|   |   |   |   |-- Tokenizer.java
|   |   |   |   |-- TokenizerResult.java
|   |   |   |   `-- package-info.java
|   |   |   `-- builtins/
|   |   |       |-- CdCommand.java
|   |   |       |-- Command.java
|   |   |       |-- EchoCommand.java
|   |   |       |-- ExitCommand.java
|   |   |       |-- HistoryCommand.java
|   |   |       |-- NoOpCommand.java
|   |   |       |-- PwdCommand.java
|   |   |       |-- TypeCommand.java
|   |   |       `-- package-info.java
|   |   `-- test/
|   |       `-- java/
|   |           `-- core/
|   |               |-- CommandHandlerTest.java
|   |               |-- TestFileUtils.java
|   |               |-- TestOutputCapture.java
|   |               `-- TestShellRunner.java
|-- .gitignore
|-- .gitattributes
|-- LICENSE
|-- CHANGELOG.md
|-- PROJECT_DOCUMENTATION.md
|-- README.md
|-- codecrafters.yml
|-- your_program.sh
`-- ...
```

Core Classes

CommandHandler

- **What it does:**
  - Parses user input
  - Decides if a command is built-in or external
  - Handles pipelines and redirection
  - Keeps track of the current working directory
- **Key methods:**
  - `handleCommand`: Main entry for executing a command or pipeline
  - `handlePipeline`: Runs a sequence of commands connected by pipes
  - `handleExternalCommand`: Runs external (non-builtin) commands
  - `tokenize`: Splits input into tokens, respecting quotes and escapes

ShellInputHandler

- **What it does:**
  - Runs the main shell loop (REPL)
  - Reads user input and handles special keys (arrows, tab, etc.)
  - Manages command history and tab completion
- **Key method:**
  - `run`: Starts the shell and keeps it running until exit

ShellHistory

- **What it does:**
  - Stores and navigates command history (up/down arrows)
- **Key methods:**
  - `add`, `previous`, `next`, `resetIndex`

TabCompleter

- **What it does:**
  - Provides tab completion for commands and files
- **Key method:**
  - `complete`: Returns a completed command or argument

ExternalCommand

- **What it does:**
  - Represents and runs external programs (not built-in)
- **Key method:**
  - `execute`: Runs the external command

Built-in Commands

All built-ins implement the `Command` interface:

- `execute(String[] args, String rawInput, Path currentDirectory)`

List of Built-ins:

- **cd**: Change directory
- **pwd**: Print working directory
- **echo**: Print arguments to the terminal
- **exit**: Exit the shell
- **type**: Show if a command is built-in or external
- **history**: Show command history
- **NoOpCommand**: Used for unknown commands (does nothing)

How the Shell Works

1. **Start the shell**: The main loop waits for user input.
2. **Parse input**: Input is tokenized, and redirections/pipelines are detected.
3. **Decide command type**: If the command is built-in, it runs directly. Otherwise, it runs as an external process.
4. **Handle pipelines**: If there are pipes (`|`), commands are connected so output from one is input to the next.
5. **Handle redirection**: Output (`>`, `>>`) and error (`2>`, `2>>`) redirections are supported.
6. **Update state**: The current directory and command history are updated as needed.

Extending the Shell

- **To add a new built-in:**
  1. Create a new class in `builtins/` that implements `Command`.
  2. Register it in `CommandHandler`'s constructor.
- **To add new features:**
  - Extend or modify the relevant class in `core/`.
- **To debug:**
  - Start with `ShellInputHandler.run()` and follow the flow to `CommandHandler`.

Getting Help

If you ever get stuck:

- Refer to this document to understand the class or method responsible for a feature.
- Check the code comments and method names—they are designed to be self-explanatory.

Contact

- For more details, contact:
  - **Adnan Mazharuddin Shaikh**
  - **Email:** adnanmazharuddinshaikh@gmail.com
  - **GitHub:** 10adnan75

Happy hacking!