Cache Simulation

This project is a cache simulation application. It is designed to read cache configurations and memory traces from files, simulate cache behavior, and generate a JSON report of cache hits, misses, and main memory accesses.

1 Features

- Read cache configurations from a JSON file.
- Simulate cache behavior based on the provided memory trace file.
- Generate a detailed report of cache hits, misses, and main memory accesses in JSON format.

2 Getting Started

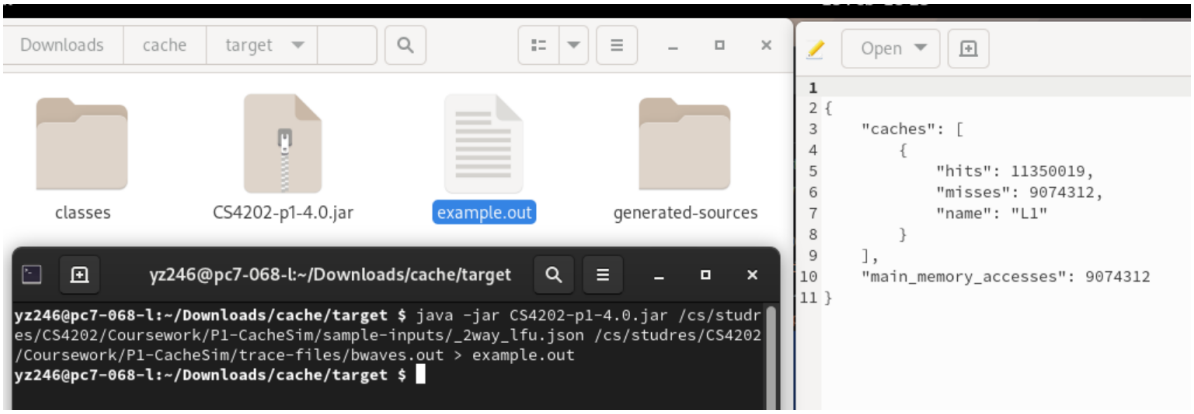
Prerequisites

- Java 17
- Maven

3 Running the Application

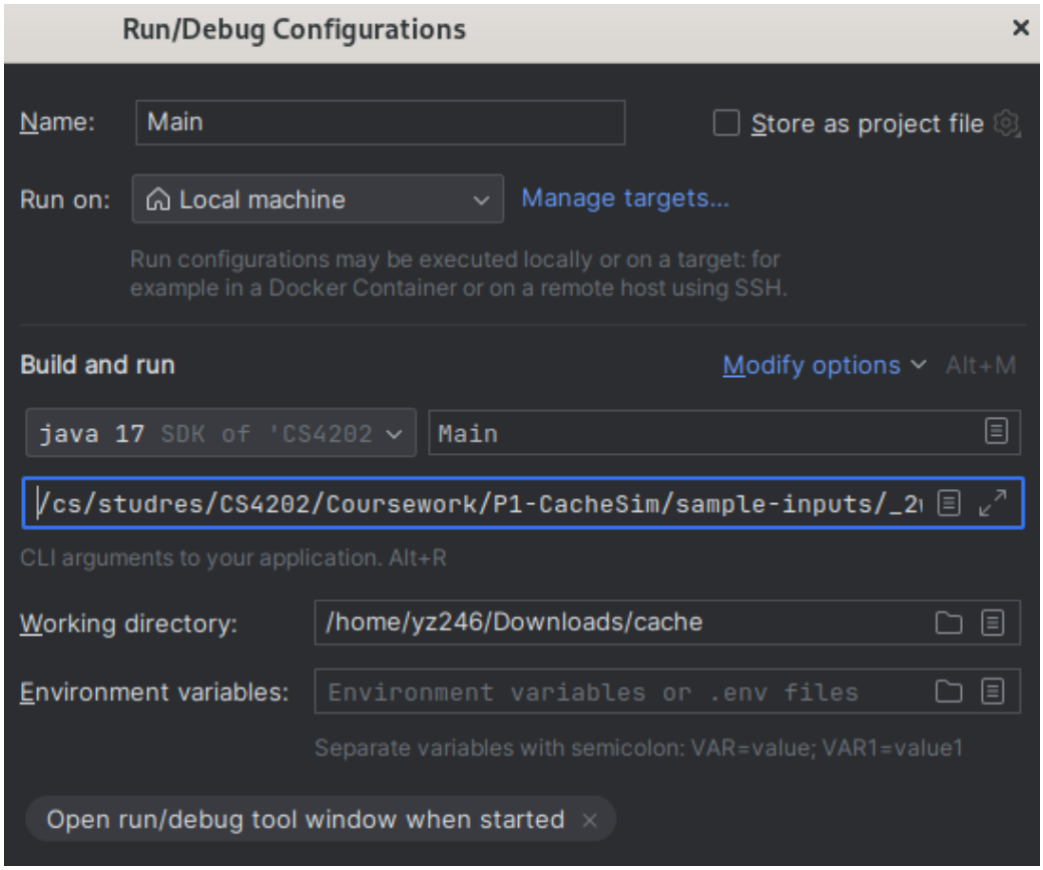
Method 1: Building and Running with Maven

1. **Build the Project**
    - Navigate to the project directory in your terminal.
    - Run `mvn clean package`. This will create an executable JAR file in the `target` directory.
2. **Run the JAR File**
    - After building, navigate to the `target` directory.
    - Run the JAR file using the command: `java -jar CS4202-p1-4.0.jar <config.json> <tracefile> > example.json`, replacing `<config.json>` and `<tracefile>` with your actual file paths. **Then the output file is in the `target` directory.**
    - The JAR file is **CS4202-p1-4.0.jar**, **NOT** the original-CS4202-p1-4.0.jar.



Method 2: Running via IntelliJ IDEA (Output in console)

1. **Import the Project**
   - Open IntelliJ IDEA and import the project by selecting the `pom.xml` file.
2. **Set Up Configuration**
   - Go to `Run` -> `Edit Configurations`.
   - Add a new `Application` configuration.
   - Set the `Main class` to `Main`.
   - In the `Program arguments`, provide the paths to your configuration and trace files. For example: `<config.json> <tracefile>`.



3. **Build and Run**
   - Build the project using `Build` -> `Build Project`.
   - Run the project with the configuration you set up.

4 Class Functionality

Cache

An abstract class representing a memory cache simulator. It defines the basic structure and behavior of a cache, including block size, cache size, replacement policy, associativity, hit number, and visit number. Subclasses must implement the `visit` method for cache simulation.

CacheConfigReader

Responsible for reading cache configuration from a JSON file. It parses the JSON file and constructs `CacheConfig` objects for each cache defined in the file. The `CacheConfig` nested class represents the configuration of a cache, containing fields for cache name, size, line size, cache kind, and replacement policy.

CacheFactory

A factory class responsible for creating different types of caches. It provides methods to create a direct-mapped cache (`plantCacheDirect`) and a set-associative cache (`plantCacheGroup`).

DirectCache

Represents a direct-mapped cache implementation. It extends the `Cache` class and implements the `visit` method for cache simulation. This class manages an array to store tags for cache blocks and handles cache hits and misses.

GroupCache

Represents a set-associative cache implementation. It extends the `Cache` class and implements the `visit` method for cache simulation. This class manages an array of linked lists representing cache block groups and supports different replacement policies.

ReplacementPolicy

An enum representing different cache replacement policies. It includes policies like Least Recently Used (LRU), Round Robin (RR), and Least Frequently Used (LFU).

TraceFileReader

Responsible for reading memory traces from a file. It reads memory addresses from the specified file and returns an array of these addresses.