Day 7: DS HashMap for scores.

## **Objective (one-liner for doc)**

Adding a thread-safe in-memory cache (ScoreCache) backed by a ConcurrentHashMap to speed up score retrieval. Cache is updated after successful persistence and used as first-level read before hitting DynamoDB.

1. **UML Diagrams**

**ClassDiagram4.puml**

@startuml

package "Persistence" {

class DynamoDbScoreRepository

class ScoreCache {

- cache : ConcurrentHashMap<String, List<ScoreRecord>>

- expiries : ConcurrentHashMap<String,Long>

- ttlMillis : long

+ put(rec:ScoreRecord) : void

+ getByStudentId(studentId:String) : List<ScoreRecord>

+ invalidate(studentId:String) : void

+ clear() : void

}

class ScoreRecord

}

package "Evaluation" {

class EvaluationEngine {

- scoreRepo : DynamoDbScoreRepository

- logRepo : DynamoDbLogRepository

- cache : ScoreCache

+ evaluateAndPersist(...) : int

+ getScores(studentId:String) : List<ScoreRecord>

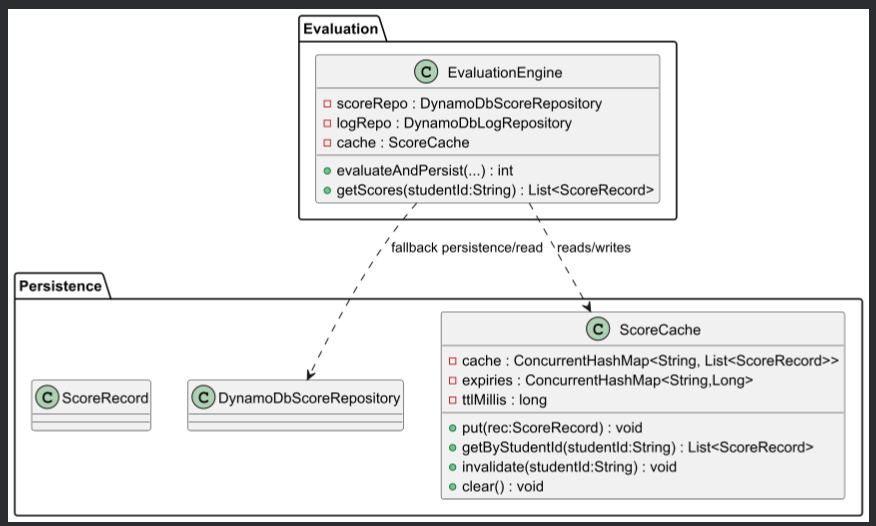
}

}

EvaluationEngine ..> ScoreCache : reads/writes

EvaluationEngine ..> DynamoDbScoreRepository : fallback persistence/read

@enduml



**SequenceDiagram5.puml**

@startuml

actor Student

participant "Controller" as API

participant "EvaluationEngine" as Engine

participant "ScoreCache" as Cache

participant "DynamoDbScoreRepository" as Repo

Student -> API : GET /scores/{studentId}

API -> Engine : getScores(studentId)

Engine -> Cache : getByStudentId(studentId)

alt cache HIT

Cache --> Engine : cached List<ScoreRecord>

else cache MISS

Engine -> Repo : queryByStudentId(studentId)

Repo --> Engine : List<ScoreRecord>

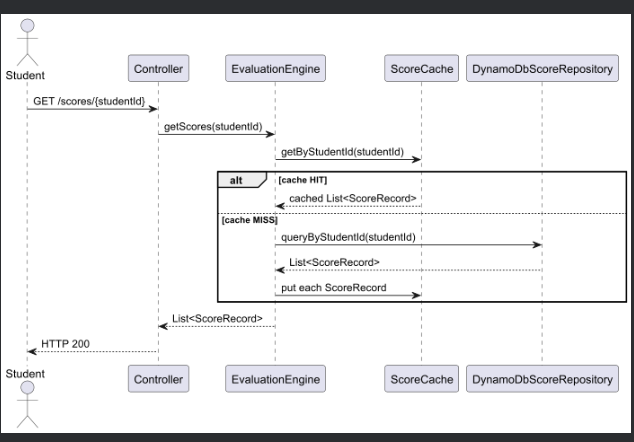
Engine -> Cache : put each ScoreRecord

end

Engine --> API : List<ScoreRecord>

API --> Student : HTTP 200

@enduml



1. **Java code**

## 1) ScoreCache.java

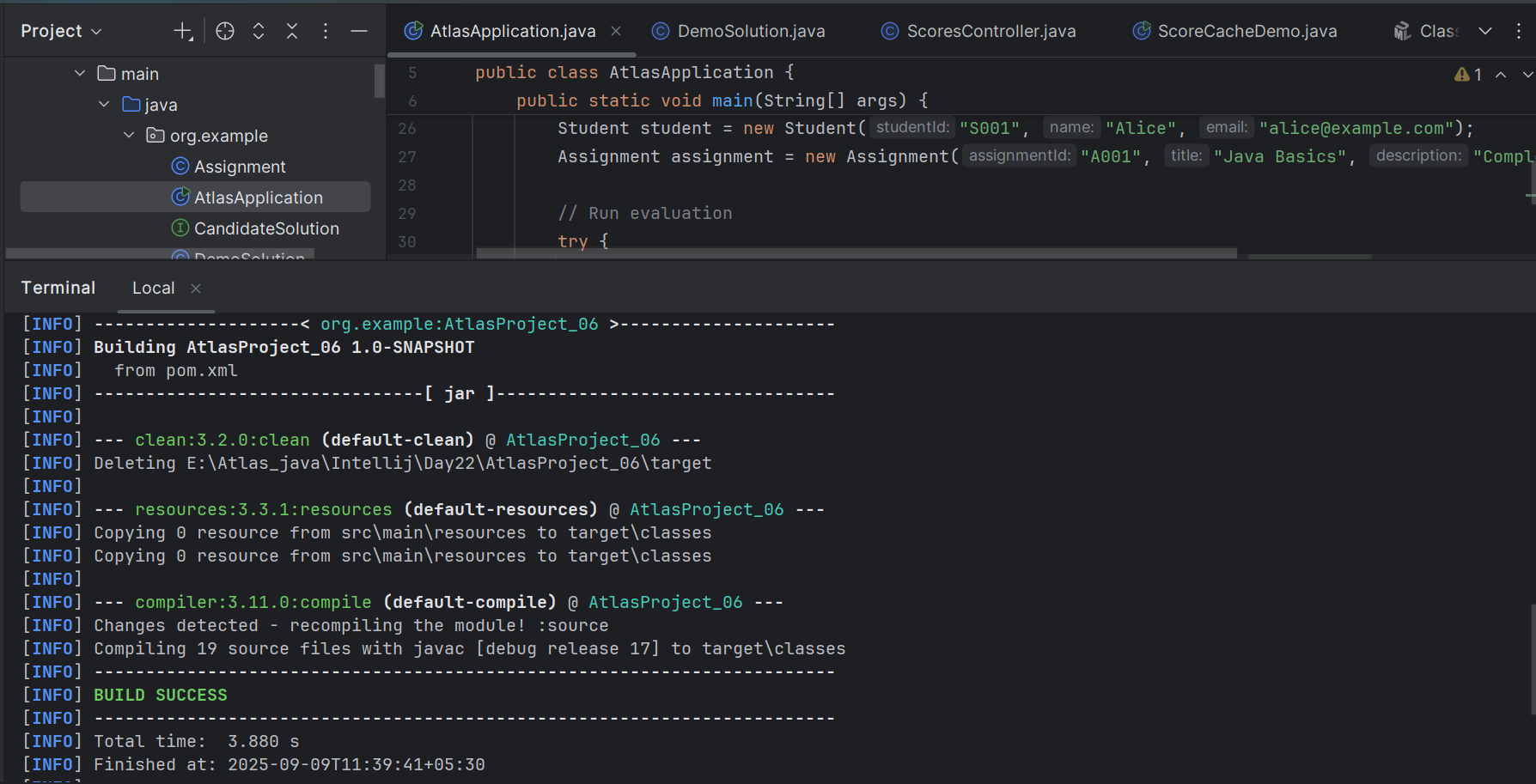
2) EvaluationEngine.java (updated)

3) ScoresController.java

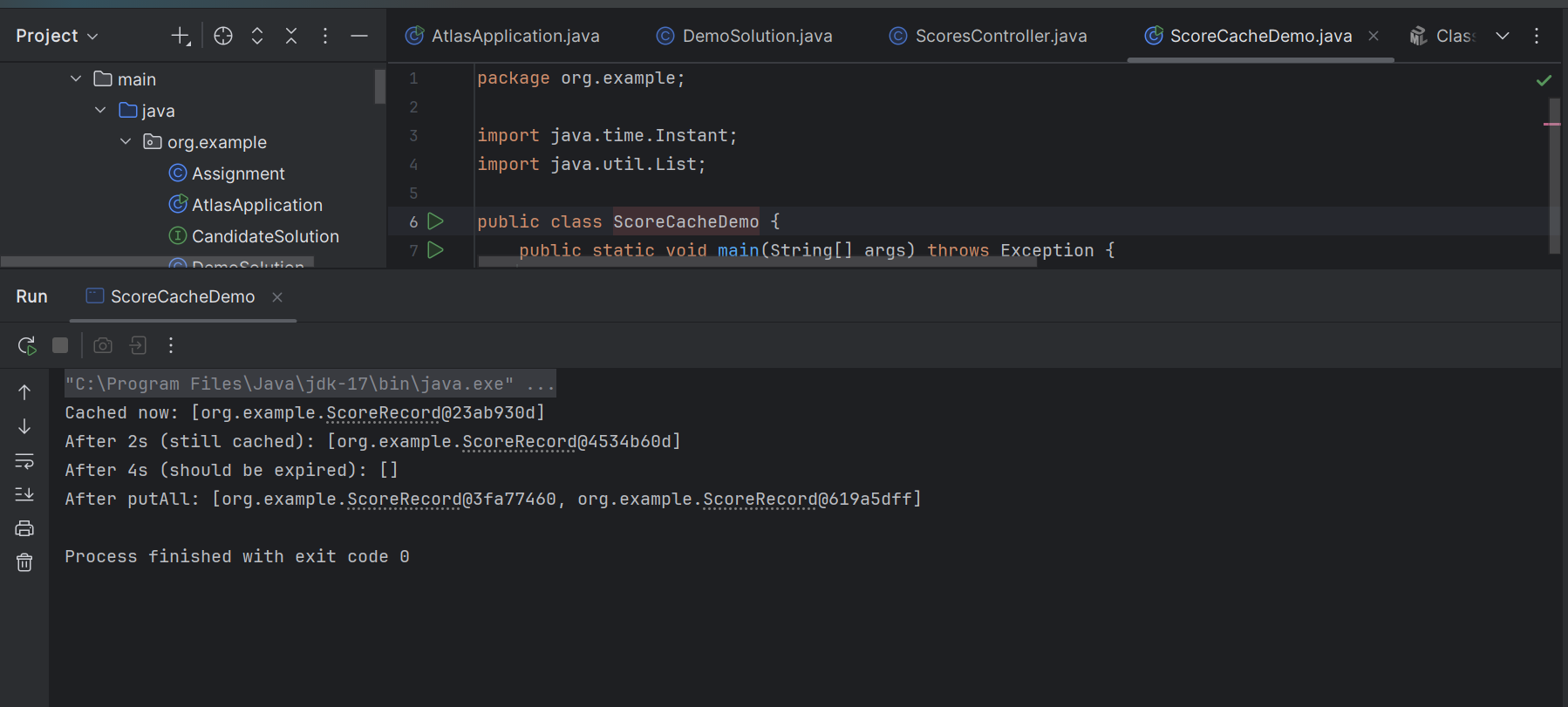
4) ScoreCacheDemo.java

## **Testing steps (IntelliJ)**

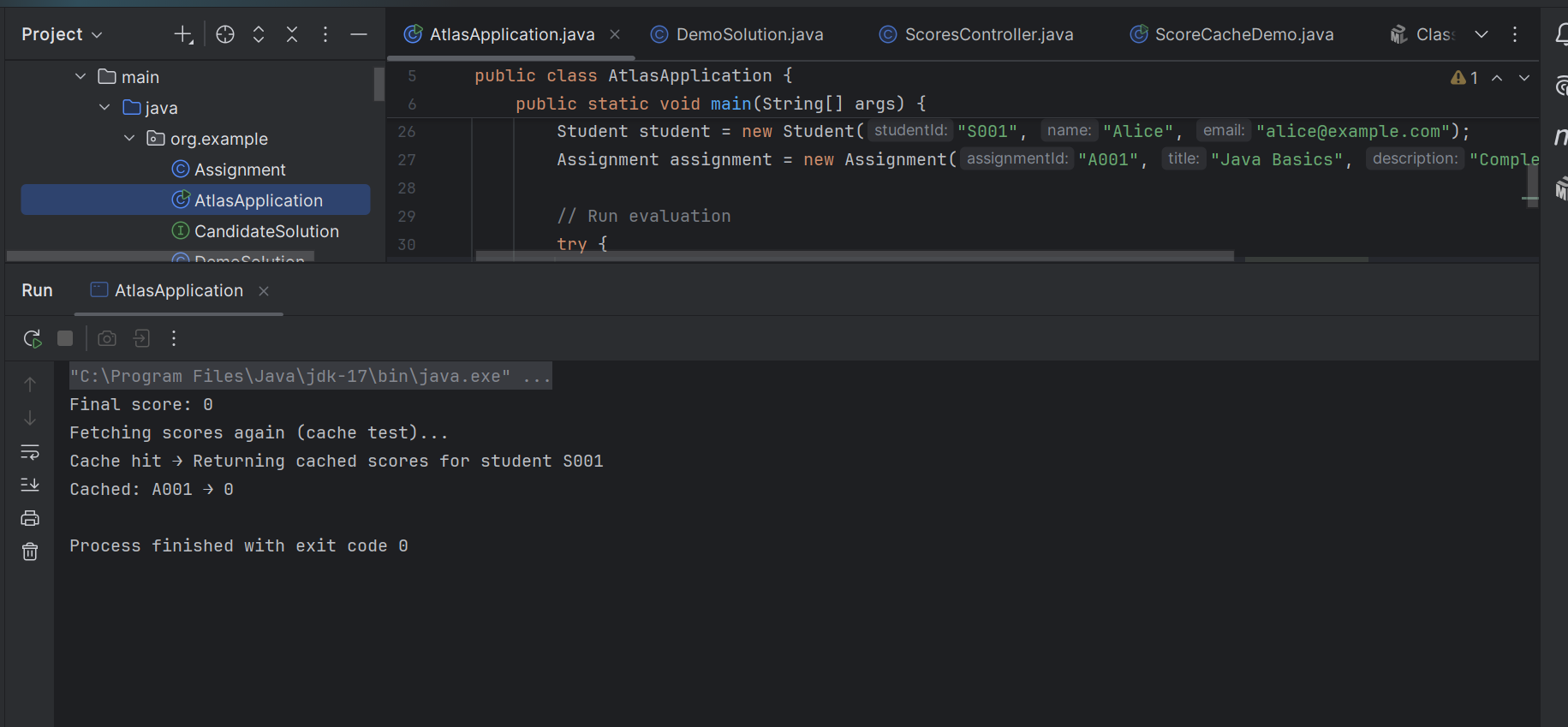
1. Adding ScoreCache.java and ScoreCacheDemo.java to src/main/java/org/example.
2. Build: mvn clean package (or Build → Rebuild Project).



1. Run ScoreCacheDemo in IntelliJ — observe TTL behavior.



1. Run AtlasApplication — submit an evaluation; confirm console shows Score persisted: ... and then you can query cache via debug/logging or extend API to read from cache (next step).



**Day 7 – Conclusion:**

* Implemented an in-memory `ScoreCache` (thread-safe `ConcurrentHashMap`) with optional TTL to accelerate score lookups.
* Integrated cache updates into `EvaluationEngine` so new scores are cached immediately after persistence.
* Added demo and test steps to validate put/get and TTL behavior.
* Next: expose a read API endpoint that prefers cache and falls back to DynamoDB, and discuss cache invalidation strategies for scale.