SCD-Lab

Lab#12

Name: Anas-Altaf

Roll.no: 22F-3639

Java Codes:

T-1:

dto.StudentDTO.java

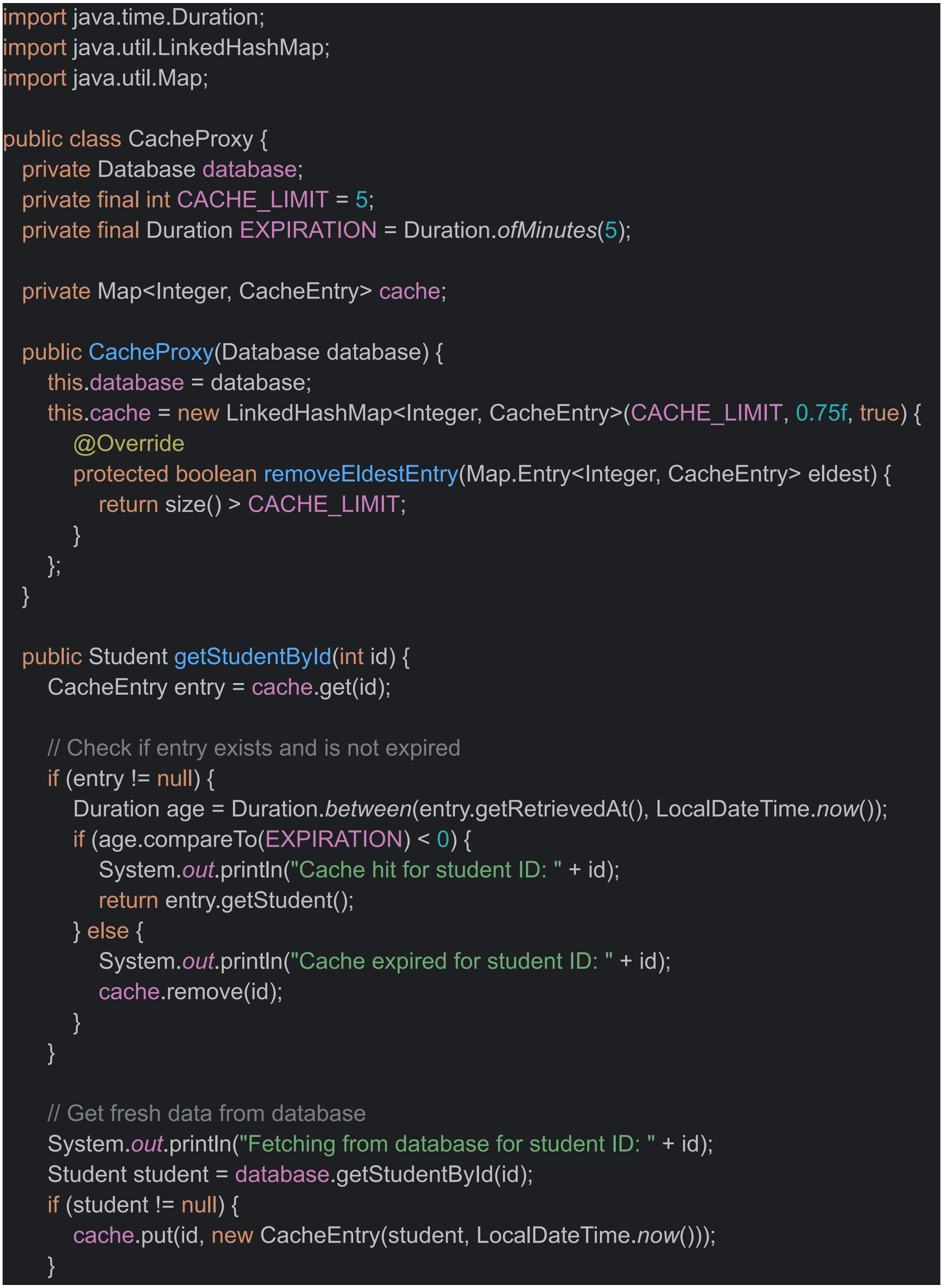
```java
package org.scblab.dto;

import java.time.LocalDateTime;

public class Student {
  private int id;
  private String name;
  private String major;
  private LocalDateTime timestamp;

  public Student(int id, String name, String major, LocalDateTime timestamp) {
      this.id = id;
      this.name = name;
      this.major = major;
      this.timestamp = timestamp;
  }

  // Getters and setters
  public int getId() { return id; }
  public String getName() { return name; }
  public String getMajor() { return major; }
  public LocalDateTime getTimestamp() { return timestamp; }
}
```

dal.Database.java

```
package org.scblab.dal;

import org.scblab.dto.Student;

import java.sql.*;

public class Database {
  private static final String URL = "jdbc:mysql://localhost:3306/lab12";
  private static final String USER = "root";
  private static final String PASSWORD = "";

  public Student getStudentById(int id) {
     String query = "SELECT id, name, major, timestamp FROM student WHERE id = ?";
     try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
           PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setInt(1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
           return new Student(
                 rs.getInt("id"),
                 rs.getString("name"),
                 rs.getString("major"),
                 rs.getTimestamp("timestamp").toLocalDateTime());
        }
     } catch (SQLException e) {
        System.out.println("Error: " + e.getMessage());
     }
     return null;
  }
}
```

dal.CacheProxy.java

```
package org.scblab.dal;

import org.scblab.dto.Student;

import java.time.LocalDateTime;
```



```
import java.time.Duration;
import java.util.LinkedHashMap;
import java.util.Map;

public class CacheProxy {
  private Database database;
  private final int CACHE_LIMIT = 5;
  private final Duration EXPIRATION = Duration.ofMinutes(5);

  private Map<Integer, CacheEntry> cache;

  public CacheProxy(Database database) {
    this.database = database;
    this.cache = new LinkedHashMap<Integer, CacheEntry>(CACHE_LIMIT, 0.75f, true) {
      @Override
      protected boolean removeEldestEntry(Map.Entry<Integer, CacheEntry> eldest) {
        return size() > CACHE_LIMIT;
      }
    };
  }

  public Student getStudentById(int id) {
    CacheEntry entry = cache.get(id);

    // Check if entry exists and is not expired
    if (entry != null) {
      Duration age = Duration.between(entry.getRetrievedAt(), LocalDateTime.now());
      if (age.compareTo(EXPIRATION) < 0) {
        System.out.println("Cache hit for student ID: " + id);
        return entry.getStudent();
      } else {
        System.out.println("Cache expired for student ID: " + id);
        cache.remove(id);
      }
    }

    // Get fresh data from database
    System.out.println("Fetching from database for student ID: " + id);
    Student student = database.getStudentById(id);
    if (student != null) {
      cache.put(id, new CacheEntry(student, LocalDateTime.now()));
    }
```

```
        return student;
    }
}
```

dal.CacheEntry.java

```
package org.scblab.dal;

import org.scblab.dto.Student;

import java.time.LocalDateTime;

public class CacheEntry {
    private Student student;
    private LocalDateTime retrievedAt;

    public CacheEntry(Student student, LocalDateTime retrievedAt) {
        this.student = student;
        this.retrievedAt = retrievedAt;
    }

    public Student getStudent() {
        return student;
    }

    public LocalDateTime getRetrievedAt() {
        return retrievedAt;
    }
}
```

TestCachingProxy.java

```
package org.scblab;

import org.scblab.dal.CacheProxy;
import org.scblab.dal.Database;
import org.scblab.dto.Student;

public class TestCachingProxy {
    public static void main(String[] args) throws InterruptedException {
        Database db = new Database();
        CacheProxy cache = new CacheProxy(db);
        System.out.println("\nTest 1: Multiple requests for same student");
        printStudent(cache.getStudentById(1));
        Thread.sleep(1000);
        printStudent(cache.getStudentById(1));
        System.out.println("\nTest 2: Cache eviction (accessing 6 different students)");
        for (int i = 1; i <= 6; i++) {
            printStudent(cache.getStudentById(i));
        }


        System.out.println("\nTest 3: Verify LRU eviction");
        printStudent(cache.getStudentById(1));


        System.out.println("\nTest 4: Cache expiration (waiting 5 minutes)");
        System.out.println("Waiting for cache to expire...");
        Thread.sleep(300000);
        printStudent(cache.getStudentById(1));
    }

    private static void printStudent(Student student) {
        if (student != null) {
            System.out.printf("Student: ID=%d, Name=%s, Major=%s, Timestamp=%s%n",
                    student.getId(),
                    student.getName(),
                    student.getMajor(),
                    student.getTimestamp());
        } else {
            System.out.println("Student not found");
        }
    }
}
```

T-2: