Game Service App

Build Steps

1. **Building the project**
   Use the below command to build the project

   mvn clean package

   Executable jar will be created under target directory

   game-service-app-1.0.0.jar

2. **Running the project**
   Run the jar using below command
   java -jar game-service-app-1.0.0.jar

   Custom property file
   If customisation is needed (eg: Port Number),
   Modify the AppConfig.proprties file from project parent directory and place it in the same directory as application jar (java -jar game-service-app-1.0.0.jar)
   if no AppConfig.properties file is given, default values will be taken as below.

```
#Port Number
server.port=8081

#Session timeout value in Minutes
session.timeout=10

#Session cleanup cycle value in Minutes
session.cleanupCycle=5
```

Java Version

Java 17

Design details

Application uses the HttpServer for creating the server and handling the requests. A request handler layer is in place to receive and handle the incoming requests. Incoming requests are validated first to avoid any error due to invalid data. The validated request is then passed to service layer which implements the business logic based on the use case (Use cases are Login, Update Score, High score list). Application also implements a scheduled cleanup thread for periodically clearing the sessions after session time out value is met.

1. Datastructure

Application maintains two in memory datastrcutures for handling the operations.

SessionMap: Concurrent Map holding the user session details

UserId1	SessionKey1
UserId2	SessionKey2

ScoreMap: Concurrent Map holding the user score details based on level

UserId1	LevelId1	Score1
	LevelId2	Score2
UserId2	LevelId1	Score1
	LevelId2	Score2

ConcurrentMap structure is used for implementation considering the requirement for handling multiple requests simultaneously without data consistency issues. Above datastructures are live throughout the application lifetime as there is no persistence layer needed as per the requirement.

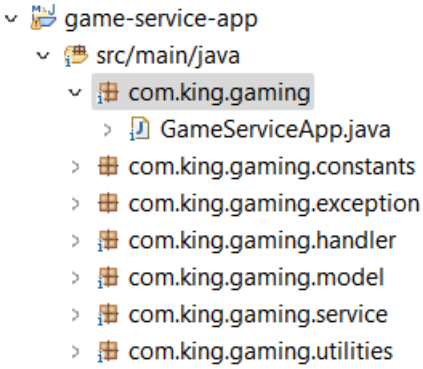
Additionaly one adhoc TreeSet is used for the high score list.

SortedScores: TreeSet holding the list of high scores.

UserScoreObject1 < userId1, score1>
UserScoreObject2 < userId2, score2>

Above TreeSet will be populated dynamically whenever request is placed for high score list for a level. Tree Set datastructure is used to leverage the capability of automatic sorting and replacing the lowest score element for high score list.

2. Package structure



Package	**Purpose**
com.king.gaming	Root package with Main class
constants	Application specific constant values
exception	Custom exceptions
handler	Request handler for incoming http requests
model	Model class for application objects
service	Business logic implementation for use cases
utilities	Utility classes like Validator,Logger, Session Cleanup etc.