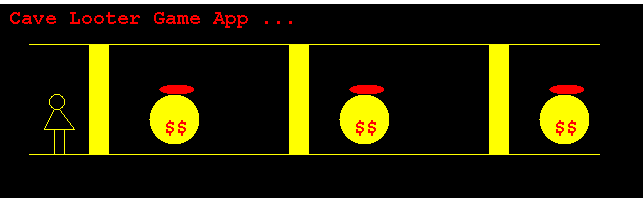
**Tarzen-The Cave Looter**

**Overview of the application:**

Looting a cave is fun, isn’t it? A cave has multiple barriers of rock. Tarzen (user’s player) uses his power to fight and break the barrier! There is ‘$$’ hidden behind each barrier which makes Tarzen Looter of the game. Now, let’s understand the game more technically.

Here is the **Pictorial representation** of the game application –



**Features:**

There are following features in the game:

1. Create player profile
2. Play game
3. View all players
4. View game history
5. Resume last played game
6. Exit

1. Create Player profile: Takes inputs from user for his player’s details and saves the profile if provided details are valid & non duplicate. It identifies duplicate records by id.

2. Play Game: Organizes game with user’s choice of player in case of multiple players found. Loads barriers with its capacity and ‘$$’. User is provided with following options to fight in order to break the rock barrier-

1. Punch – 30

2. Push – 20

3. Kick – 50

4. Save and Exit

On player’s choice in above actions, above specified amount of power will be impacted on player’s total power.

3. View all players: displays all player profiles.

4. View game history: Display all the games finished by user’s players.

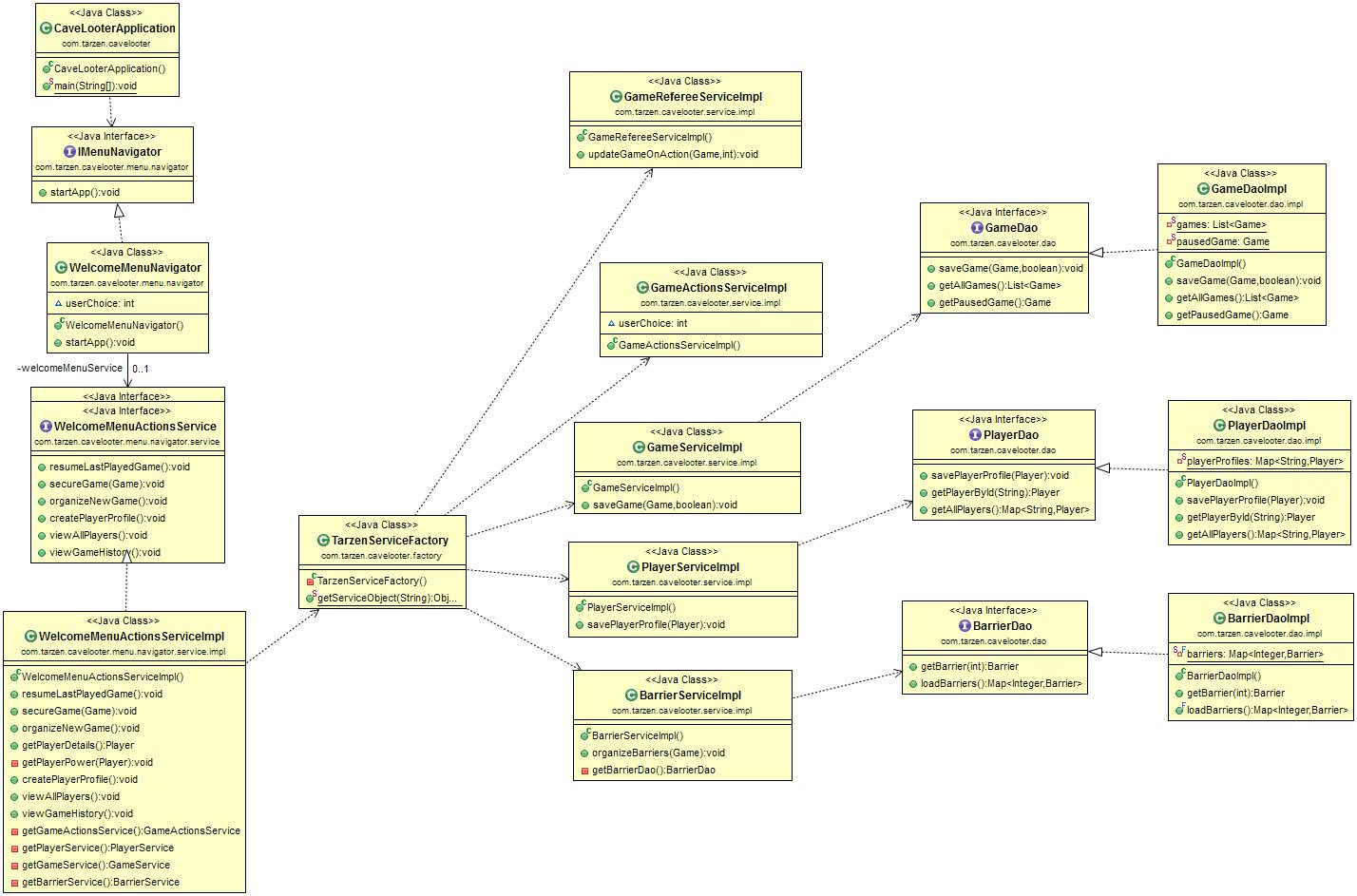
5. Resume last played game: Resumes game from the point where it was stopped earlier.

6. Exit: User exits the game.

**Pre-requisites:**

1. Maven
2. Java 8
3. Mockito and power mockito
4. Jacoco & PMD for code quality & code coverage

**Class Diagram:**

****

**Steps to Execute:**

1. Go to project location
2. Run ‘mvn clean install’
3. On Successful build of application – jar named ‘**Tarzen-Cavelooter-0.0.1-SNAPSHOT.jar’** is generated in target folder.
4. Open command prompt and run **java –jar Tarzen-Cavelooter-0.0.1-SNAPSHOT.jar** to start the execution.

**Sample Output with all possible scenarios:**

****

**Areas of Extension:**

1. **Adding multiple number of barriers –** For demonstration purpose, I have limited number of barriers to be 3. We can add ‘n’ number of barriers loaded with random capacity and bonus in required range of integers to enhance the game app. This change can be accommodated by just adding new barrier objects inBarrierDaoImpl -> loadBarriers() methodand rest of the app remains same.
2. **Adding more player’s actions -** Currently, I have added 3 actions of player (kick,push & punch) using which he can break the barrier. We can add more actions along with use of some small weapons – hit wooden rod, throw bomb on barrier etc. This enhanced feature can be accommodated by adding cases in GameActionsServiceImpl-> playGame() method’s switch case calling updateGameOnAction(game,‘associatedPower’). Rest of the things work as it is.
3. **Boosting player’s life (power):** As of now, after each barrier is broken, player finds bonus to be grabbed. On few random barriers, we can introduce lifePower attribute, which increases player’s power (life) on breaking that specific barrier. This change can be accommodated by making changes in minimal number of classes –
   1. Barrier.java – to hold below objects:
      1. isLifePowerEnabled – boolean
      2. lifePower – integer
   2. Provide above mentioned attributes in BarrierDaoImpl -> loadBarriers() method to initialize the barrier objects and rest of the app remains same.
   3. GameRefereeServiceImpl – updateGameOnAction() method – added condition to check if current barrier isLifePowerEnabled. If yes, add lifePower to player’s power attribute with one liner print.