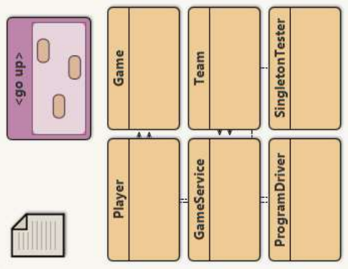
<go up>
Game
Player
Team
GameService
SingletonTester
ProgramDriver

```java
package src.com.gamingroom;

/**
 * A simple class to hold information about a player
 * <p>
 * Notice the overloaded constructor that requires
 * an id and name to be passed when creating.
 * Also note that no mutators (setters) defined so
 * these values cannot be changed once a player is
 * created.
 * </p>
 * @author coce@snhu.edu
 *
 */
public class Player {
    long id;
    String name;

    /*
     * Constructor with an identifier and name
     */
    public Player(long id, String name) {
        this.id = id;
        this.name = name;
    }

    /**
     * @return the id
     */
    public long getId() {
        return id;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Player [id=" + id + ", name=" + name + "]";
    }
}

```

```java
package src.com.gamingroom;

/**
 * A simple class to hold information about a game
 *
 * <p>
 * Notice the overloaded constructor that requires
 * an id and name to be passed when creating.
 * Also note that no mutators (setters) defined so
 * these values cannot be changed once a game is
 * created.
 * </p>
 *
 * @author coce@snhu.edu
 *
 */
public class Game {
    long id;
    String name;

    /**
     * Hide the default constructor to prevent creating empty instances.
     */
    private Game() {
    }

    /**
     * Constructor with an identifier and name
     */
    public Game(long id, String name) {
        //deleted this()
        this.id = id;
        this.name = name;
    }

    /**
     * @return the id
     */
    public long getId() {
        return id;
    }

      /**
     * @set the id
     */
    //added setID
    public void setId(long id) {
        this.id = id;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @set the name
     */
    public void SetName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {

        return "Game [id=" + id + ", name=" + name + "]";
    }

}
```

```java
package src.com.gamingroom;

import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;      //Added

/**
 * A singleton service for the game engine
 *
 * @author coce@snhu.edu
 */
public class GameService {

    /**
     * A list of the active games
     */
    private static List<Game> games = new ArrayList<Game>();

    /*
     * Holds the next game identifier
     */
    private static long nextGameId = 1;

    // FIXME: Add missing pieces to turn this class a singleton
    public static GameService GS_Singleton = null;  //declare and set to null

    public static GameService getInstance() {
        if (GS_Singleton == null) {
            GS_Singleton = new GameService();        //if null then singleton is NEW
game service
        }
        return GS_Singleton;
    }

    /**
     * Construct a new game instance
     *
     * @param name the unique name of the game
     * @return the game instance (new or existing)
     */
    public Game addGame(String name) {

        // a local game instance
        Game game = null;

        // FIXME: Use iterator to look for existing game with same name
        // if found, simply return the existing instance
        Iterator iter = games.iterator();
        while (iter.hasNext()) {            //while games has next set 1st game to
iter.next
```

```java
            Game gameOne = (Game)iter.next();
            if (name.equalsIgnoreCase(gameOne.getName())) {
            game = gameOne;
        }
    }
        // if not found, make a new game instance and add to list of games

        if (game == null) {
            game = new Game(nextGameId++, name);
            games.add(game);
        }

        // return the new/existing game instance to the caller
        return game;
    }

    /**
     * Returns the game instance at the specified index.
     * <p>
     * Scope is package/local for testing purposes.
     * </p>
     * @param index index position in the list to return
     * @return requested game instance
     */
    Game getGame(int index) {
        return games.get(index);
    }

    /**
     * Returns the game instance with the specified id.
     *
     * @param id unique identifier of game to search for
     * @return requested game instance
     */
    public Game getGame(long id) {

        // a local game instance
        Game game = null;

        // FIXME: Use iterator to look for existing game with same id
        Iterator iter = games.iterator();
        while (iter.hasNext()) {        //while games has next
            Game gameOne = (Game) iter.next();
        // if found, simply assign that instance to the local variable
            if (gameOne.getId() == id) {
                game = gameOne;
            }
        }
        return game;
    }

    /**
     * Returns the game instance with the specified name.
     *
     * @param name unique name of game to search for
     * @return requested game instance
     */
    public Game getGame(String name) {

        // a local game instance
        Game game = null;

        // FIXME: Use iterator to look for existing game with same name
        Iterator iter = games.iterator();
        while (iter.hasNext()) {
            Game gameOne = (Game) iter.next();
            if (name.equalsIgnoreCase(gameOne.getName())) {
                game = gameOne;
            }
        }
        // if found, simply assign that instance to the local variable

        return game;
    }

    /**
     * Returns the number of games currently active
     *
     * @return the number of games currently active
     */
    public int getGameCount() {
        return games.size();
    }
}

```

```java
package src.com.gamingroom;

/**
 * A simple class to hold information about a team
 * <p>
 * Notice the overloaded constructor that requires
 * an id and name to be passed when creating.
 * Also note that no mutators (setters) defined so
 * these values cannot be changed once a team is
 * created.
 * </p>
 * @author coce@snhu.edu
 *
 */
public class Team {
    long id;
    String name;

    /*
     * Constructor with an identifier and name
     */
    public Team(long id, String name) {
        this.id = id;
        this.name = name;
    }

    /**
     * @return the id
     */
    public long getId() {
        return id;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Team [id=" + id + ", name=" + name + "]";
    }
}

```

```java
package src.com.gamingroom;

/**
 * Application start-up program
 *
 * @author coce@snhu.edu
 */
public class ProgramDriver {

    /**
     * The one-and-only main() method
     *
     * @param args command line arguments
     */
    public static void main(String[] args) {

        // FIXME: obtain reference to the singleton instance
        GameService service = GameService.getInstance(); // replace null with ???

        System.out.println("\nAbout to test initializing game data...");

        // initialize with some game data
        Game game1 = service.addGame("Game #1");
        System.out.println(game1);
        Game game2 = service.addGame("Game #2");
        System.out.println(game2);

        // use another class to prove there is only one instance
        SingletonTester tester = new SingletonTester();
        tester.testSingleton();
    }
}

```

```
package src.com.gamingroom;

/**
 * A class to test a singleton's behavior
 *
 * @author coce@snhu.edu
 */
public class SingletonTester {

	public void testSingleton() {

		System.out.println("\nAbout to test the singleton...");

		// FIXME: obtain local reference to the singleton instance
		GameService service = GameService.getInstance(); // replace null with
???

		// a simple for loop to print the games
		for (int i = 0; i < service.getGameCount(); i++) {
			System.out.println(service.getGame(i));
		}

	}

}

```