基于.NET Core 的客户分数排名服务实现的测试用例

一、功能测试用例（核心业务场景）

1. TC-F-001：Update Score 接口 - 新增客户（初始无数据，分数为正）

前置条件：服务已启动，内存中无任何客户数据。

测试步骤：1. 发送 POST 请求：`/customer/15514665/score/124`；2. 查看接口响应结果；3. 验证内存中客户数据是否新增。

预期结果：1. 响应状态码为 200；2. 响应体返回 `{"currentScore": 124}`3. 内存中成功新增客户 15514665，分数记录为 124。

2. TC-F-002：Update Score 接口 - 已有客户分数增加（分数仍为正）

前置条件：服务已启动，客户 3333333 已存在，当前分数为 80。

测试步骤：1. 发送 POST 请求：`/customer/3333333/score/30`；2. 查看响应结果；3. 验证客户分数是否更新。

预期结果：1. 响应状态码为 200；2. 响应体返回 `{"currentScore": 110}`；3. 内存中客户 3333333 的分数更新为 110。

3. TC-F-003：Update Score 接口 - 已有客户分数减少（分数仍为正）

前置条件：服务已启动，客户 3333333 已存在，当前分数为 80。

测试步骤：1. 发送 POST 请求：`/customer/3333333/score/-20`；2. 查看响应结果；3. 验证客户分数是否更新。

预期结果：1. 响应状态码为 200；2. 响应体返回 `{"currentScore": 60}`；3. 内存中客户 3333333 的分数更新为 60。

4. TC-F-004：Update Score 接口 - 客户分数更新后≤0（移除出排行榜）

前置条件：服务已启动，客户 254814111 已存在，当前分数为 50。

测试步骤：1. 发送 POST 请求：`/customer/254814111/score/-100`；2. 查看响应结果；3. 尝试查询该客户排名。

预期结果：1. 响应状态码为 200，返回 `{"currentScore": -50}`；2. 内存中无客户 254814111 的数据；3. 查询该客户返回 404 响应，提示“客户不存在或分数≤0，未参与排行榜”。

5. TC-F-005：Update Score 接口 - 参数校验（customerid 为负）

前置条件：服务已启动。

测试步骤：1. 发送 POST 请求：`/customer/-123/score/50`；2. 查看响应结果。

预期结果：1. 响应状态码为 400；2. 响应体提示“CustomerID必须为正整数”。

6. TC-F-006：Update Score 接口 - 参数校验（score 超范围+1001）

前置条件：服务已启动。

测试步骤：1. 发送 POST 请求：`/customer/15514665/score/1001`；2. 查看响应结果。

预期结果：1. 响应状态码为 400；2. 响应体提示“分数调整范围需在[-1000,1000]内”。

7. TC-F-007：Get by rank 接口 - 正常查询排名区间（start≤end，范围有效）

前置条件：服务已启动，内存中存在排名 1-10 的客户（数据参考文档示例：15514665 分数124排名1、81546541分数113排名2等）。

测试步骤：1. 发送 GET 请求：`/leaderboard?start=3&end=5`；2. 查看响应结果。

预期结果：1. 响应状态码为 200；2. 响应数据包含排名 3-5 的客户：客户 1745431（Score:100，Rank:3）、客户 76786448（Score:100，Rank:4）、客户 254814111（Score:96，Rank:5）。

8. TC-F-008：Get by rank 接口 - 查询排名区间（start>end，参数无效）

前置条件：服务已启动。

测试步骤：1. 发送 GET 请求：`/leaderboard?start=5&end=3`；2. 查看响应结果。

预期结果：1. 响应状态码为 400；2. 响应体提示“start排名需≤end排名”。

9. TC-F-009：Get by rank 接口 - 查询排名区间（范围超出实际排名）

前置条件：服务已启动，当前仅存在排名 1-10 的客户。

测试步骤：1. 发送 GET 请求：`/leaderboard?start=11&end=15`；2. 查看响应结果。

预期结果：1. 响应状态码为 200；2. 响应数据为空列表（`{"data": []}`）。

10. TC-F-010：Get by customerid 接口 - 正常查询（含上下游排名，high=2，low=3）

前置条件：服务已启动，客户 7777777 存在，排名 91（数据参考文档示例：7786448 分数313排名89、54814111分数301排名90等）。

测试步骤：1. 发送 GET 请求：`/leaderboard/7777777?high=2&low=3`；2. 查看响应结果。

预期结果：1. 响应状态码为 200；2. 响应数据包含排名 89-94 的客户：7786448（Score:313，Rank:89）、54814111（Score:301，Rank:90）、7777777（Score:298，Rank:91）、96144320（Score:298，Rank:92）、16144320（Score:270，Rank:93）、2000437（Score:239，Rank:94）。

11. TC-F-011：Get by customerid 接口 - 查询不存在的客户

前置条件：服务已启动，内存中无客户 9999999 数据。

测试步骤：1. 发送 GET 请求：`/leaderboard/9999999?high=1&low=1`；2. 查看响应结果。

预期结果：1. 响应状态码为 404；2. 响应体提示“客户不存在或分数≤0，未参与排行榜”。

12. TC-F-012：Get by customerid 接口 - 参数校验（high 为负）

前置条件：服务已启动，客户 15514665 存在。

测试步骤：1. 发送 GET 请求：`/leaderboard/15514665?high=-1&low=1`；2. 查看响应结果。

预期结果：1. 响应状态码为 400；2. 响应体提示“high/low需为非负整数”。

13. TC-F-013：排名逻辑 - 分数相同，按 CustomerID 升序排

前置条件：服务已启动，内存中新增 3 个客户：客户 8009471（Score:93）、客户 6144320（Score:93）、客户 11028481（Score:93）。

测试步骤：1. 发送 GET 请求：`/leaderboard?start=7&end=9`；2. 查看响应中 3 个客户的排名。

预期结果：1. 响应状态码为 200；2. 排名顺序为：客户 6144320（Score:93，Rank:7）、客户 8009471（Score:93，Rank:8）、客户 11028481（Score:93，Rank:9）（按 CustomerID 升序排列）。

二、非功能测试用例（性能、兼容性）

1. TC-NF-002：服务性能 - 大量客户数据下 Get 接口响应

前置条件：服务已启动，内存中导入 10 万客户数据（分数随机，排名逻辑符合“分数降序→ID升序”）。

测试步骤：1. 发送 GET 请求：`/leaderboard?start=1000&end=1010`；2. 发送 GET 请求：`/leaderboard/{随机客户ID}?high=5&low=5`；3. 记录两次请求的响应时间。

预期结果：1. 两次请求响应时间均≤300ms；2. 响应数据排名与分数逻辑正确（无排名颠倒、分数错误）。

2. TC-NF-003：兼容性 - .NET Core 版本兼容性（.NET 7/.NET 8）

前置条件：分别在安装 .NET 7 和 .NET 8的服务器上部署服务。

测试步骤：1. 在 .NET 7 环境：执行 `dotnet run` 启动服务，调用 Update Score 和 Get 接口各 10 次；2. 在 .NET 8环境：重复步骤 1。

预期结果：1. 两个环境下服务均能正常启动，无启动报错；2. 所有接口调用均成功，响应结果与预期一致（无版本导致的参数解析错误、逻辑异常）。

3. TC-NF-004：服务稳定性 - 长时间运行（24 小时）

前置条件：服务已启动，内存中初始存在 5 万客户数据，每小时触发 100 次随机 Update Score 请求。

测试步骤：1. 持续监控服务运行 24 小时；2. 期间每 4 小时调用一次 Get 接口验证数据；3. 检查内存使用情况。

预期结果：1. 服务无崩溃、无卡死，可正常接收并处理请求；2. 数据一致性正常（无分数/排名异常，如客户分数未更新、排名与分数不匹配）。

三、异常测试用例（边界与错误场景）

1. TC-E-001：Update Score 接口 - score 为边界值（+1000/-1000）

前置条件：服务已启动，客户 15514665 存在，当前分数为 0。

测试步骤：1. 发送 POST 请求：`/customer/15514665/score/1000`；2. 发送 POST 请求：`/customer/15514665/score/-1000`；3. 查看两次响应。

预期结果：1. 第一次响应：状态码 200，返回 `{"currentScore":1000}`；2. 第二次响应：状态码 200，返回 `{"currentScore":0}`；3. 客户 15514665 分数为 0，仍保留在内存（未移除，因仅分数>0参与排行榜，分数=0不参与但无需移除）。

2. TC-E-002：Update Score 接口 - customerid 为 int64 最大值

前置条件：服务已启动，内存中无客户 9223372036854775807 数据。

测试步骤：1. 发送 POST 请求：`/customer/9223372036854775807/score/50`；2. 查看响应与内存数据。

预期结果：1. 响应状态码为 200，返回 `{"currentScore":50}`；2. 内存中成功新增该客户数据，无ID超限报错。

3. TC-E-003：Get by rank 接口 - start/end 为 0 或负数

前置条件：服务已启动，存在排名 1-10 的客户。

测试步骤：1. 发送 GET 请求：`/leaderboard?start=0&end=5`；2. 发送 GET 请求：`/leaderboard?start=-2&end=3`。

预期结果：1. 两次请求均返回状态码 400；2. 响应体提示“start/end排名需为正整数”。

4. TC-E-004：服务重启 - 重启后内存数据丢失

前置条件：服务已启动，内存中存在 10 个客户数据（含不同分数、排名）。

测试步骤：1. 停止服务；2. 重新启动服务；3. 调用 Get 接口查询原客户数据。

预期结果：1. 服务重启成功，无启动异常；2. 查询原客户数据返回 404 响应（内存数据已丢失，符合“无持久化，仅内存存储”要求）