Tennis Scoring

Application Kata

Important: Take the iterations like you would in a real project. In iteration 1 you don't know about iteration 2 and so on. Please do not peek!

Background

You are an ambitious tennis player who dedicates most of his spare time to his local tennis club. Once in a while your club organizes tournaments. Traditionally the scores of each match are tracked by volunteers on score boards.

Those score boards show the current game score, the set score and the match score. The development of a match result like 3:1 is unfortunately lost after the match.

As everyone knows, that you are a passionate programmer, the management committee of the tennis club asks you to write a scoring software for them.

Iteration 1

Write a console program that tracks the score of a tennis match. The players are labeled as player a and player b.

As a user you should be able to either input an 'a', when player a has scored or 'b' when player b has scored.

As a response the program will show you **two** results.

The first results shows what has happened in the match:

- When no game is won, the game scores of both players will be shown. E.g. Game: Fifteen – Thirty.
- When a game is won, the set scores will be displayed. E.g. Set: 2 – 4.
- When a set is won, the match scores will be displayed. E.g. Match: 1 – 2.

The second result shows the entire **history** of the match. For each game played in the match, output a headline showing the ordinals s and g for the corresponding game and set.

```
Set [s], Game [g]
------
```

Underneath each headline output the result of every single serve.

```
[Player a score] – [Player b score]
```

The following example shows the second part of the output for: "bbaaabbbab"

```
Set 1, Game 1
------
Love – Fifteen
Love – Thirty
Fifteen – Thirty
Thirty – Thirty
Fourty – Thirty
Deuce – Deuce
Deuce – Advance

Set 1, Game 2
-----
Fifteen – Love
Fifteen – Fifteen
```

Output the history underneath the first output. Clear the console before each output.

There is **no limit** to the number of sets, that can be played.

Simplified scoring rules:

A set ends, when a player has won at least six games and the scores deviate from each other by two or more points.

A match is won, when a player has won at least three sets.

Iteration 2

There are some variations to the scoring rules of tennis, which should be handled in the scoring program.

Upon starting the program the settings of the match should be queried. Either by console input or a config file, that's up to you.

The settings are made up of:

- the number of sets, that have to be won in order to win a match
- the number of games, that have to be won to win a set
- a flag that indicates, whether tie breaks are to be played or not

When a tie break[1] is played, the game after the set score of 6:6 will be scored in a different way. In this situation, assuming we are in set 1, the input "aaabbabaabbbaa" will give you:

```
Set: 1 - 0

Set 1, tie break
------
1 - 0
2 - 0
3 - 0
3 - 1
3 - 2
4 - 2
4 - 3
5 - 3
6 - 3
6 - 4
6 - 5
6 - 6
7 - 6
8 - 6
```

The first player to reach a miniumum of 6 points with an advance of two points wins the tie break.

When a match is won, a **match summary** should be the **only** displayed output. E.g.:

```
Winner: Player B with 3 : 1
---------------------------
Set         | 1 | 2 | 3 | 4 |
---------------------------
Player A    | 2 | 6 | 4 | 1 |
---------------------------
Player B    | 6 | 3 | 6 | 6 |
```

[1] Rules: http://de.wikipedia.org/wiki/Tie-Break

Iteration 3

In a tournament you might want to track the scores of multiple matches. The matches are numbered from 0,1,2,… to n.

Introduce new commands to your program:

init	starts a new tennis match. Further input of ‘a’ and ‘b’ lead to the scoring in the new match.
match [i]	Context is switched to match ‘i’. Where ‘i' is the ordinal of the match. Further input of ‘a’ and ‘b’ lead to the scoring in match [i]. Also the current match summary will be displayed.
summary	Displays the match summary of the current match.
matches	Displays all matches with their ordinal ‘i' and the current match scores

Scoring a match, that is **already won**, should have no effect.

Iteration 4

All the matches should be persistent.

Add persistency support for at least two different persistency providers like text files, xml, json, mysql, sql, mongo, couch, raven, Neo4J, etc.

On application startup I want to choose which provider is to be used by inputting a simplified connection string.

Iteration 5

In order to score a tournament, with plenty of matches being played concurrently, we should forget about the console.

Implement RESTful services, that provide the same functionality like the console application. The match command is no longer needed. Instead we will provide a match-id for each command.

Additionally, the names of the opponents in each match should be provided as parameters: /init/foo,bar will initialize a match “foo vs. bar” and return an unique identifier ‘id’ for the match

Example:

Request	Response
init/foo,bar	{id: 123abc}
score/123abc/a	{status : [“Fifteen”, “Love”], event:”GameResult”}
Json/summary/123abc	{players:[“foo”,”bar”], summary:[[0,0]]}
json/matches	{matches:[{ players:[“foo”,”bar”], score:[0,0]}]}
Json/history/123abc	{ players:[“foo”,”bar”], sets: [[[“Fifteen”, “Love”]]]

The details of the implementation are up to you.

Add a html representation for the services summary, matches and history.