implementation of the 'Swiss Simple' engine in JavaPairing

(rules changed respect to the document '**Swiss System Based on Rating**
Approved by the 1992, 1997 and 1998, 2001, 2006 General Assemblies')

A.2 criteria c) 'FIDE Title' at the moment is ignored by JavaPairing. I supposed it be not so frequent to require writing specific code, which is difficult because players may be loaded from text files where the field 'Title' may have all kind of code inside! To overcome this lack, three workarounds at your choice are offered:
1. insert players in the right order from the beginning or fine tune the order in the main grid by using menu options offered by right mouse click before making the 1st round
2. alter the Elo of one player in such a way it can be correctly sorted by decreasing Elo. After 1st round generation, reset the value to the original one. Order then is by Elo and ID, so it doesn't change any more during the tournament
3. (advanced user) do 1st round so IDs are wrongly assigned, save the tournament file, open it with a text editor and exchange the IDs of the players out of order, save it and reopen with JavaPairing. Output the round that now has correct pairings

A.3 Score brackets. If odd score group, JavaPairing moves down one player (downfloater) preferably of the dominating colour that can be paired in the/a following group. The procedure works in two steps. First, a possible upfloater is selected (like Dubov system does), then a group member that can be paired with the upfloater is found. Second, these two players are discharged to be paired later. The remainder of the score group is now even so it is tried to be paired with the standard Swiss mechanism. If pairing results impossible, because game already played or color constrain, perhaps very infrequent event, the two discharged players are called back and eventually the score group in incremented by two more upfloaters. The resulting group is treated as homogeneous and paired with the standard Swiss mechanism

A.4 Floats. For an easier code flow, JavaPairing registers and takes care of only upfloaters. Those are by definition players paired with a player having higher score. JavaPairing ensures that the same player will not be upfloated again in the next two rounds. I evaluated an accidental repeated downfloat less penalizing than upfloat.

A.5 Byes. The player who receives the BYE is selected in advance. It is the last ranked not having had a BYE or not played games for any reason in the tournament. This resembles Dubov and Lim systems. It may sound strange, but the original rule is better useful on manual pairing than on automation! This implementation requires even groups and it is better not to try to introduce not strongly needed exceptions. In fact exceptions may result in dirty code and bugs

B.4 As many players as possible receive their colour preference, without regard of x. This resembles the Lim system. I had specific requests to stress “paranoid” colour allocation!

B.5 & B.6. Only upfloaters are needed to be handled (see notes on A.3 and A.4)

C.6 p pairings are obtained in compliance with B1, B2 and *Relative Criteria* if possible. It sounds better to me. For heterogeneous score groups, the downfloater is not simply paired with the highest Rating, but an attempt is made to find a player with the correct colour due

C.12 is not needed any more, because the algorithm grants pairing at this step

C.13 JavaPairing breaks the last done pair at a time, then considers all the remaining players as a homogeneous group. As precaution, more probably to happen at last rounds of a few players tournament, JavaPairing suggests to switch to Amalfi Rating if a huge score group is so selected, because in this situation the usual S1 vs S2 rule alters the tournament and probably leads to unfair secondary prizes win. Amalfi Rating is suggested here because it performs pairing in strict ranking order. It you agree, remove the round, go to the setup window and change the play system, then redo the round and compare calculations