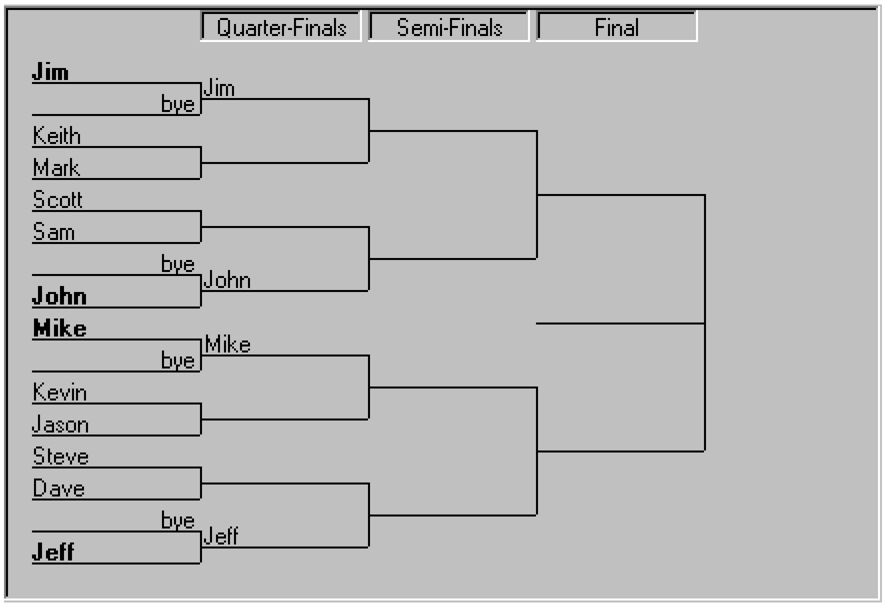
Overview



DrawSE is an ActiveX control which enables Windows developers to build applications which utilize common Single-elimination, or "knockout" competition brackets. It incorporates all the logic required to perform contestant Seeding, proper Bye placement, random placement of unseeded contestants, and input of match results to advance winners to the next round.   It is ideal for software developers who want to create tournament management applications which utilize the single-elimination draw format.   It can accomodate any competition size between 2 and 128 contestants.

DrawSE is not sport-specific.   It can be used for individual competitions such as tennis tournaments, wrestling meets, drag racing events, and spelling bees.   It can just as easily be used for team events like basketball tournaments, baseball tournaments, and school academic competitions.   As long as the competition follows the single-elimination format, DrawSE is up to the task.

Properties

Methods

Events

Using the DrawSE control

About DrawSE

Properties

BackColor
Byes
CompetitionStarted
Complete
ContestantColor
ContestantFont
Contestants
Drawn
DrawSize
MaxContestants
PercentToSeed
PreventAssocConflicts
Rounds
ScoreColor
ScoreFont
ScoresRequired
SeedingType
Seeds
StrictSeeding
UIEnabled

BackColor

Data type: OLE_COLOR.

Default value: COLOR_BACKGROUND.

Run-time usage: Read/write.

Summary:
This property is used to set the background color of the control (on which the brackets are drawn).

Byes

Data type: Integer.

Run-time usage: Read-only.

Summary:
This property indicates the number of byes in the draw. This value gets set when the draw is made.

CompetitionStarted

Data type: Boolean.

Run-time usage: Read-only.

Summary:

This property is indicates whether any match results have been recorded.

Complete

Data type: Boolean.

Run-time usage: Read-only.

Summary:

This property indicates whether all of the matches in the draw have been completed.

ContestantColor

Data type: OLE_COLOR.

Default value: COLOR_WINDOWTEXT.

Run-time usage: Read/write.

Summary:
This property is used to set the color in which the contestant's names are displayed in the draw brackets.

ContestantFont

Data type: System font.

Default value: MS Sans Serif, 8 pt.

Run-time usage: Read/write.

Summary:
This property is used to set the font in which the contestant's names are displayed in the draw brackets.

Contestants

Data type: Integer.

Valid range of values: 0 to MaxContestants.

Run-time usage: Read-only.

Summary:
This property is used to limit the number of contestants who can enter the draw. Setting this property at run-time after the draw has been made will result in a trappable error.

Drawn

Data type: Boolean.

Run-time usage: Read-only.

Summary:

This property is indicates the draw has been made.

DrawName

Data type: String.

Run-time usage: Read/write.

Summary:

This property assigns a name to the draw.

DrawSize

Data type: Integer.

Run-time usage: Read-only.

Summary:
This property is indicates the number of first-round contestant slots in the draw. Equal to Contestants + Byes. This value gets set when the draw is made.

MaxContestants

Data type: Integer.

Valid range of values: 2, 4, 8, 16, 32, 64, 128 (stops at 16 in free version of control).

Default value: 32.

Run-time usage: Read/write.

Summary:
This property is used to limit the number of contestants who can enter the draw. Setting this property at run-time after the draw has been made will result in a trappable error.

PercentToSeed

Data type: Integer.

Valid range of values: 0, 25, 50, or 100.

Default value: 25.

Run-time usage: Read/write.

Summary:

This property is used to determine approximately what percentage of the contestants will be seeded when making the draw. A value of 0 is equivalent to a SeedingMethod value of Random. Setting this property at run-time after the draw has been made will result in a trappable error.

PreventAssocConflicts

Data type: Boolean.

Default value: False.

Run-time usage: Read/write.

Summary:

If this property is True, special consideration will be given to contestants' associations when making the draw. The idea behind this is to avoid pairing contestants from the same association in the first round of competition. Setting this property at run-time after the draw has been made will result in a trappable error.

Rounds

Data type: Integer.

Run-time usage: Read-only.

Summary:
This property indicates the number of rounds of competition in the draw. This value gets set when the draw is made.

ScoreColor

Data type: OLE_COLOR.

Default value: COLOR_WINDOWTEXT.

Run-time usage: Read/write.

Summary:
This property is used to set the color in which the match scores are displayed in the draw brackets.

ScoreFont

Data type: System font.

Default value: Small Fonts, 6 pt.

Run-time usage: Read/write.

Summary:

This property is used to set the font in which the match scores are displayed in the draw brackets.

ScoresRequired

Data type: Boolean.

Default value: True.

Run-time usage: Read/write.

Summary:

If this property is True, scores must be provided when entering a match result. This is not applicable to players advancing due to a first-round bye. Setting this property at run-time after match results have been recorded (CompetitionStarted = True) will result in a trappable error.

SeedingMethod

Data type: Integer.

Valid range of values: 0, 1, 2

- 0 = Ascending
  - Using their seed values, contestants will be seeded from low to high, i.e. as in numeric ranking order.
- 1 = Descending
  - Using their seed values, contestants will be seeded from high to low, meaning the higher the number, the stronger the contestant.
- 2 = Random
  - Draw will be made randomly, with no consideration given to seed values.

Default value: 0.

Run-time usage: Read/write.

Summary:

This property is used to determine the method of seeding to be used when making the draw. A value of Random is equivalent to a PercentToSeed value of 0. Setting this property at run-time after the draw has been made will result in a trappable error.

Seeds

Data type: Integer.

Run-time usage: Read-only.

Summary:
This property indicates the number of seeds in the draw. This value gets set when the draw is made.

StrictSeeding

Data type: Boolean.

Default value: True.

Run-time usage: Read/write.

Summary:
If this property is False, then all seeds within a given Seeding Tier are treated as equivalent. This means that these seed slots will be populated randomly using the contestants who belong to the tier, offering more seeding flexibility. *Example*: Seed slots in the 5-8 tier could be filled by seeds 7, 6, 8, 5 in this order.

UIEnabled

Data type: Boolean.

Default value: True.

Run-time usage: Read/write.

Summary:
This property determines whether the user is allowed to enter match results by dragging contestants into the next round slot.

Methods

AddContestant
CancelDraw
EnterResult
GetContestant
GetContestantsList
GetDrawData
GetSeed
MakeDraw
MatchesPerRound
RemoveContestant
RestoreDraw
SwitchContestants
UndoResult

AddContestant

Parameters:

ContestantName (string):

The name of the contestant being added to draw.

SeedValue (long integer):

A numeric value associated with the contestant to be used for determining contestant seedings. Values of zero will cause the contestant to be treated as unseeded.

Association (string, optional):

A name or code which is used to identify a contestant's association or the geographic the contestant represents. This is useful when it is desirable to separate contestants in the draw based on such association. This value is ignored if PreventAssocConflicts is set to False.

Return data type: None.

Summary:

This method is invoked whenever a contestant is to be added to the draw. This method will raise a trappable error when the following conditions exist:

1) Contestants = MaxContestants

2) PreventAssocConflicts = True and *Association* = ""

3) The *ContestantName* has already been entered

4) Drawn = True and Byes = 0

5) Drawn = True and SeedingMethod <> Random and *SeedValue* > seed value of lowest seeded contestant

6) CompetitionStarted = True

CancelDraw

Parameters: None.

Return data type: Boolean.

Summary:
This method is used to cancel a draw which has already been made. If the competition has already begun, the user will be prompted on whether to proceed or not. This method has no effect on the contestants entered. The function returns True if successful, else False.

EnterResult

Parameters:

Round (integer):
The round number of the completed match.

Match (integer):
The match's relative position within its round.

WinningSlot (integer):
The winning contestant's match slot (either 1 or 2).

Scores (string):
Match scores from the winner's perspective.

Return data type: None.

Summary:

This method is called whenever the results of a match are determined so that the draw can be updated. It is also called automatically when using "drag-and-drop" to advance a winning contestant to the next round in the draw.

GetConstestant

Parameters:

Round (integer):
: The round number of the match.

Match (integer):
: A relative number of the match within the round.

Slot (integer):
: The slot number, 1 or 2.

Return data type: String.

Summary:

This method returns the the name of a contestant in a particular match slot.

GetContestantsList

Parameters: None.

Return data type: String.

Summary:

This method is used to build a pipe-delimited string of contestant names.

GetDrawData

Parameters: None.

Return data type: String.

Summary:
This method is used to build a pipe-delimited string representation of the current state of the draw. Working in conjunction with the RestoreDraw method, this enables the developer of the host application to save the draw information in a database so that the control can be restored to this state in the future. The string will be formatted as follows:

```
Round|Match|Contestant1|Contestant2|Winner|Scores|~ ...repeat...
```

The pipe character, **|**, serves as a field delimiter while the tilde, ~, signifies end of record.

GetSeed

Parameters:

Seed (integer): The seed number to search for.

Return data type: String.

Summary:

This method is used to retrieve the name of a contestant based on seed number. Valid range of seed numbers is 1 through Seeds.

MakeDraw

Parameters: None.

Return data type: None.

Summary:
This method is used to make the draw after all of the contestants have been entered. This method will raise a trappable error if Contestants < 2 or if Drawn = True.

MatchesPerRound

Parameters:

Round (integer): The round number.

Return data type: Integer.

Summary:

This method is used to determine how many matches are in a given round. This method will raise a trappable error if Drawn = False or the Round is invalid.

RemoveContestant

Parameters:

ContestantName (string, optional):

The name of the contestant being removed from the draw.

Return data type: Boolean.

Returns True if the contestant was successfully removed, else False. The False condition occurs if there is no contestant entered matching the ContestantName parameter.

Summary:

This method is used to remove a contestant from the draw. If ContestantName is not specified, then all contestants will be removed. This method will raise a trappable error if CompetitionStarted = True or if Drawn = True and the contestant to be removed happens to be seeded.

RestoreDraw

Parameters:

DrawData (string):

A pipe-delimited string containing data from a saved draw.

Return data type: None.

Summary:

This method is used to restore the draw conrol to a previous state. Working in conjunction with the GetDrawData method, this enables the developer of the host application to retrieve draw data from a database and redisplay it. The string will be formatted as follows:

Round|Match|Contestant1|Contestant2|Winner|Scores|~ ...repeat...

The pipe character, |, serves as a field delimiter while the tilde, ~, signifies end of record.

SwitchContestants

Parameters:

Contestant1 (string):
The name of the first contestant to be switched.

Contestant2 (string):
The name of the second contestant to be switched.

Return data type: None.

Summary:

This method is used to switch places in the draw between two contestants. This method will raise a trappable error under the following conditions:

1) Drawn = False.

2) CompetitionStarted = True.

3) Either of the two contestant names are not entered in the competition.

4) PercentToSeed = 100 and the seeding value assigned to each of the two contestants when are not equal.

5) Either of the two contestants is the #1 or #2 seed.

6) StrictSeeding = True and both contestants are in the same seeding tier.

7) Both contestants are seeded, but not in the same seeding tier.

UndoResult

Parameters:

Round (integer):

The round number of the completed match.

Match (integer):

The match's relative position within its round.

Return data type: None.

Summary:

This method is used to undo a match result. This method will raise a trappable error if the Round/Match values do not refer to a valid completed match.

Events

Advance
Final
ScoreChanged

Advance

Parameters:

Round (Integer):
The round number of the completed match.

Match (Integer):
The match number of the completed match.

Winner (String):
The name of the winning contestant.

Loser (String):
The name of the losing contestant.

Scores (String:)
The match scores, if provided.

Summary:

This event occurs whenever a contestant advances in the draw, either by dragging a contestant to the next round or as a result of the EnterResult method.

Final

Parameters: None.

Summary:

This event occurs whenever the final match of the draw has been completed.

ScoreChanged

Parameters:

Round (Integer):

The round number of the match.

Match (Integer):

The match number of the match.

PreviousScore (String:)

The match score being changed.

NewScore (String:)

The new match score.

Summary:

This event occurs whenever a completed match score is changed either by using the popup menu when right-clicking on a score or immediately after a score is provided when entering a match result.

Glossary

Bye

Depending on how many contestants are entered in a competition, there may be some vacant match slots which are known as byes. Contestants paired opposite of a bye slot will automatically advance to the second round.

Seeding Tier

This is a concept where seed positions can be grouped into equivalent levels. When making the draw, this concept will only be used if StrictSeeding is set to False. The tiers exist as follows:

Tier	Seeds
1	1, 2
2	3, 4
3	5 - 8
4	9 - 16
5	17 - 32
6	33 - 64

Single-elimination

This is perhaps the most common of competition formats. The basic premise is, "lose once and your out." Starting with a group of competitors, one-on-one pairings are made. The winner of each pairing advances to the next round while the losers are removed from the competition, thereby trimming the number of remaining competitors in half each round. The pairing and advancing process continues until only one contestant remains, the champion.

Seeding

The concept of seeding pertains to structuring the draw such that the stronger contestants are placed in specific, advantageous slots in the draw. This strategy is designed to eliminate the weaker contestants early in the competition, leaving only the best to battle it out the rest of the way.

About DrawSE

DrawSE ActiveX Control:
Demo version 1.0.0 (limited to 16 contestants)
Full version 2.0.0

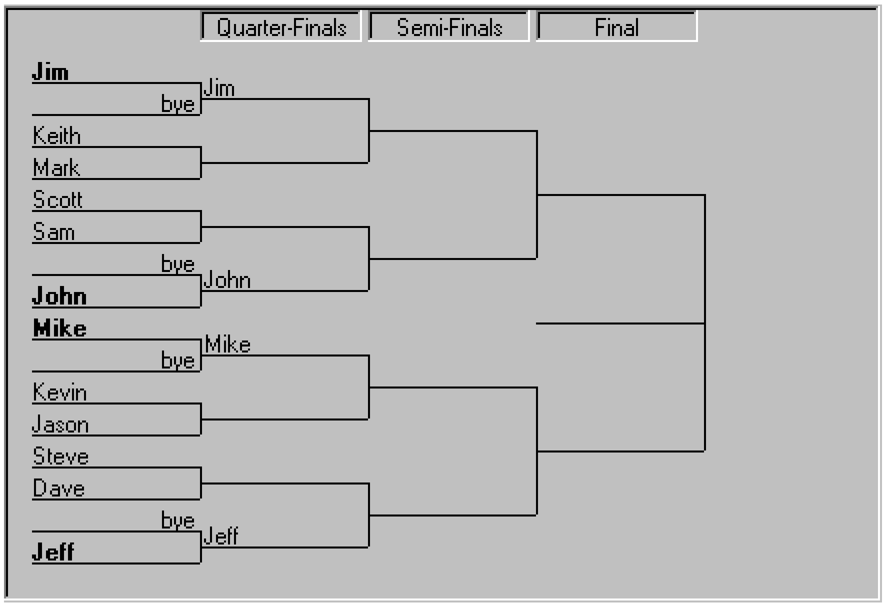
Author
Robert Mayer

Delos Software
http://software.delos.net

Registration
The price for the full version is **$35.00**. This includes *free lifetime* upgrades. Contact the author via email for ordering details. The demo version is always free, of course. All product support is also handled via email. Bug reports, functionality requests, and miscellaneous questions/comments are highly encouraged.

Credits
Special thanks go out to Robert Swilley and Mike Yarbrough who assisted in developing the complex draw-making algorithms.

Using the DrawSE control



In order to use the control, first you must understand how the rounds, matches, and slots in the draw are indentified.   Identifying the rounds is quite obvious.   The leftmost "column" of slots make up round 1 and the round number increments as you move to the right.   Within each round, the matches are identified from 1 to *n*.   Each match consists of two slots.   In the example above, the *Keith vs. Mark* match would be Round 1, Match 2, with Keith occupying Slot 1 and Mark in Slot 2.

Setting up a draw

Adding contestants

Making the draw

Switching contestants in the draw

Advancing contestants

Undoing a match result

Saving a draw

Restoring a draw

Setting up a draw

Before you begin entering contestants in a draw, you must define several configuration parameters:

1) Set the MaxContestants property. This establishes a limit to how many contestants can enter. The valid range of values is 2, 4, 8, 16, 32 (default), 64, and 128.

2) Set the SeedingMethod property. This determines whether contestants will be seeded in Ascending, Descending, or Random fashion. A professional tennis tournament might use the Ascending method by seeding players based on their current world rankings. In table tennis and chess, sanctioned tournament players have a numeric ratings whereby the higher the number, the better the player. The Descending method would be appropriate for such competitions. A handicap bowling tournament could use the Random method since the premise is that all contestants have been equalized.

3) Set the PercentToSeed property. This controls approximately what percentage of the contestants will be seeded, with the remainder being placed randomly throughout the draw. Valid values are 0, 25 (default), 50, and 100.

4) Set the PreventAssocConflicts property. Set this to True if you would like to avoid pairing contestants from the same association (see AddContestant property) in the first round. Default value is False.

5) Set the StrictSeeding property. Setting this to False will enable more flexibility in the seeding by treating contestants within the same seeding tier as equivalent.

6) Set the ScoresRequired property. Setting this to True means that you will be automatically prompted for match scores after advancing a contestant in the draw.

Adding contestants

Obviously, contestants must be entered before a competition can take place. This is accomplished by calling the AddContestant method. Most single-elimination competitions utilize a seeding process whereby the stronger contestants are placed in favorable positions in the draw. Contestants are seeded based on comparison of a numeric *seeding value* assigned to them at the time they are added. Contestants may optionally be added with an *association*, which can be used to separate those of like *association* in the draw by setting the PreventAssocConflicts property to True. See the following code sample:

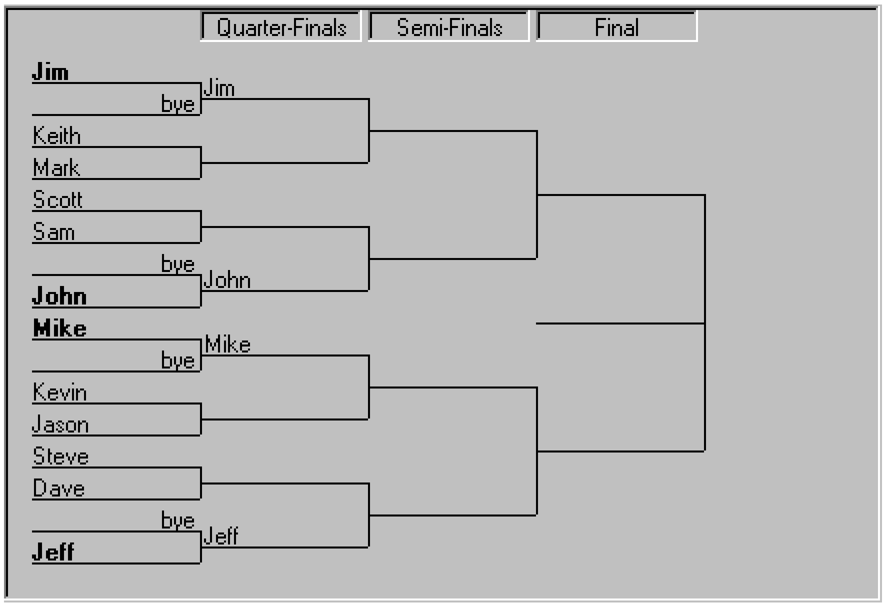
```
DrawSE1.AddContestant "Joe Blow", 2, "Chicago"
```

Making the draw

Once the properties are set and all of the contestants have been added, you are ready to make the draw. Simply call the MakeDraw method to do so:

```
DrawSE1.MakeDraw
```

Switching contestants in the draw

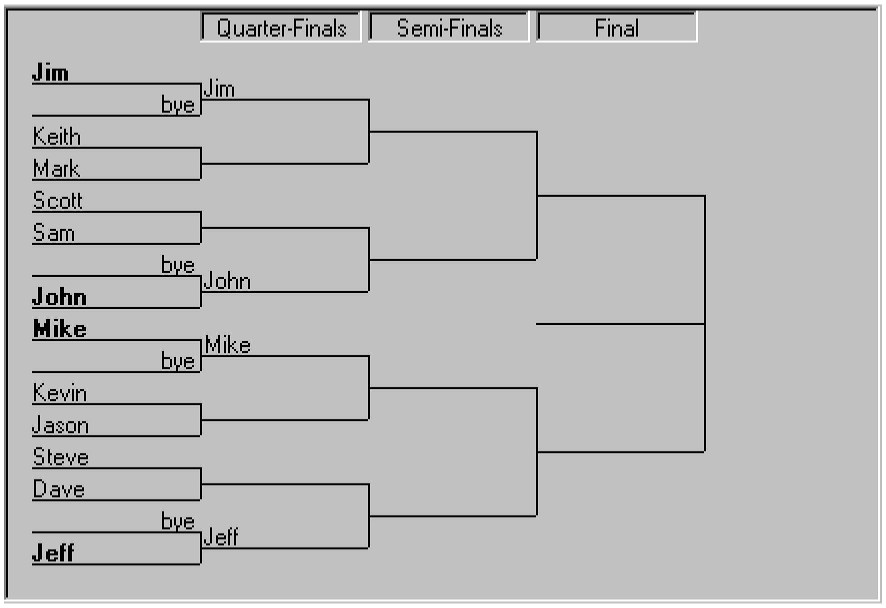


After the draw is made, you still have some flexibility to make changes.   Call the SwitchContestants method to allow two contestants trade places in the draw:

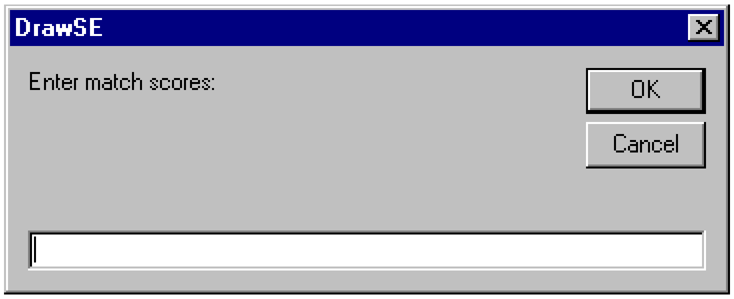
```
DrawSE1.SwitchContestants "Kevin", "Steve"
```

Advancing contestants

In a single-elimination draw two contestants will face one another resulting in the winner advancing and the loser being eliminated. There are basically three ways to update the DrawSE control to reflect match results. The first two are handled by working directly with the user interface of the control, and the other is strictly programatic:



1) *Drag-and-drop.* This is perhaps the most intuitive way to accomplish this, as it is consistent with how many operations are handled in Windows applications. Using the mouse, simply click on the winning contestant in a first-round match and drag the name over to the appropriate next round slot and release. This will cause the winning contestant to now appear in that slot. If the ScoresRequired property is set to True, you will first be prompted to enter the scores before the draw is updated.



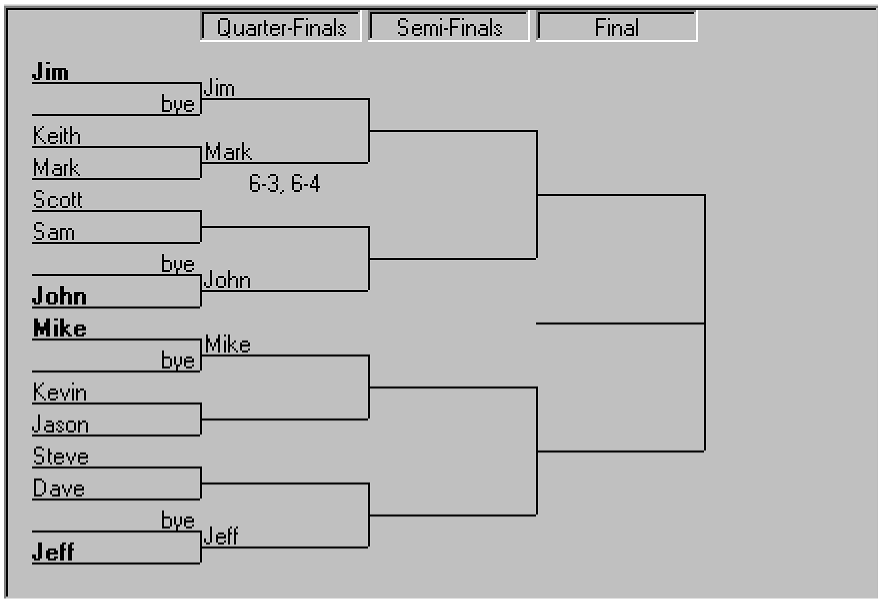
Enter match scores:

If you try to drag a contestant to the incorrect slot, you will get an error message.

2) *Select "Advance" from the popup menu*. If you right-click on a contestant in a match not yet completed, a popup menu will appear. Selecting "Advance" will cause the contestant to be advanced to the next round slot. As mentioned above, you will be prompted for input of scores the ScoresRequired property is set to True.

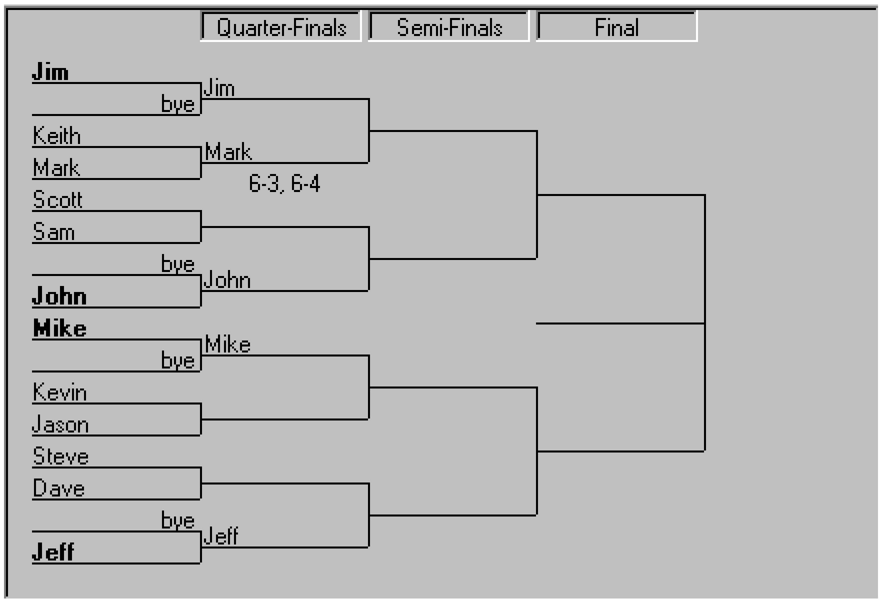
3) *Use the EnterResult method*. To advance Mark as a winner over Keith programatically, follow the code example below:

```
DrawSE1.EnterResult (1, 2, 2, "6-3, 6-4")
```



Undoing a match result

Anyone can make a mistake, so it is necessary that you have the capability to undo a match result. If you mistakenly advance the wrong contestant in a match, you have two ways to go about undoing your error:



1) *Select "Undo Result" from the popup menu*. If you right-click on an advanced contestant in a completed match, a popup menu will appear. Selecting "Undo Result" will restore the draw to the state it was in prior to advancing this contestant.

2) *Use the UndoResult method*. To undo the match result between Keith and Mark programatically, use the following code:

```
DrawSE1.UndoResult 1, 2
```

Saving a draw

If you are using DrawSE as part of a tournament application, it is very likely that you will have a need to save the draw data into a database for future retrieval. This is accomplished by using the GetDrawData method. This method returns a single string representation of the entire draw in its current state. Although, saving the whole string into a database or flat file could suffice, it would probably be more useful to parse the separate elements out of the string and store the data in a structured database table. See the sample VB project included (SAMPLE.VBP) for a detailed code example.

Restoring a draw

After using GetDrawData to save a draw, you can recall it later using the RestoreDraw method.   See the sample VB project included (SAMPLE.VBP) for a detailed code example.