Task 6 – Interactive Selection/User-Centric Fitness

This task adds selection and creates the user interface for ranking melody pairs and basslines. The Music class was also updated to include fields for bassline rank and melody pair rank. Individuals are selected via the same means as RGB individuals. The *interactive-selection* method displays each of the selected individuals to the user using EasyABC notation and prompts the user for rankings of the melody pairs and basslines for each music sample. There is error-handling for incorrect user inputs. At the moment, the rankings only support natural numbers from 0 to 10. Once all rankings have been input by the user, the music sample object gets updated with the new rankings.

Demo

```
[2]> ( setf pop ( initial-population ) )
#<POPULATION #x1AB82501>
[3]> ( setf selection ( select-individuals pop ) )
(#<MUSIC #x1AB35F8D> #<MUSIC #x1AB2A73D> #<MUSIC #x1AB2F1C9>
#<MUSIC #x1AB32C19>)
[4]> ( interactive-selection selection )
----------------Individual 99----------------
X:1
T:Individual 99
C:Dystopian Tuesday
M:4/4
L:1/4
Q:1/4=120
V:S clef=treble name=S
V:A clef=treble name=A
V:T clef=treble name=T
%%score [ ( S ) ( A ) ( T ) ]
K:C
V:S
%%MIDI program 0
E A2 D2 A D2
V:A
G' A' G2' A' G2' A/2' B/2'
V:T
D, D4, G2, G,
Melody 1 & 2 ranking (out of 10)? a
```

```
[ERROR] A ranking must be a natural number x such that -1 < x <
11.
Melody 1 & 2 ranking (out of 10)? -1

[ERROR] A ranking must be a natural number x such that -1 < x <
11.
Melody 1 & 2 ranking (out of 10)? 11

[ERROR] A ranking must be a natural number x such that -1 < x <
11.
Melody 1 & 2 ranking (out of 10)? 4.5

[ERROR] A ranking must be a natural number x such that -1 < x <
11.
Melody 1 & 2 ranking (out of 10)? 4
Bassline ranking (out of 10)? 9

----------------Individual 26----------------
X:1
T:Individual 26
C:Dystopian Tuesday
M:4/4
L:1/4
Q:1/4=120
V:S clef=treble name=S
V:A clef=treble name=A
V:T clef=treble name=T
%%score [ ( S ) ( A ) ( T ) ]
K:C
V:S
%%MIDI program 0
G/2 D/2 G2 A/2 B G B/2 D2
V:A
B/2 F/2 B2 E/2 F B F/2 F2
V:T
E2, E2, D2, E2,
Melody 1 & 2 ranking (out of 10)? 2
Bassline ranking (out of 10)? 6

----------------Individual 57----------------
```

```
X:1
T:Individual 57
C:Dystopian Tuesday
M:4/4
L:1/4
Q:1/4=120
V:S clef=treble name=S
V:A clef=treble name=A
V:T clef=treble name=T
%%score [ ( S ) ( A ) ( T ) ]
K:C
V:S
%%MIDI program 0
D' C/2' D2' D/2' D2' B2'
V:A
D' D2' C/2' D/2' B2' D2'
V:T
G2, E, G2, B2, G,
Melody 1 & 2 ranking (out of 10)? 7
Bassline ranking (out of 10)? 3

----------------Individual 79----------------
X:1
T:Individual 79
C:Dystopian Tuesday
M:4/4
L:1/4
Q:1/4=120
V:S clef=treble name=S
V:A clef=treble name=A
V:T clef=treble name=T
%%score [ ( S ) ( A ) ( T ) ]
K:C
V:S
%%MIDI program 0
C/2 B B/2 A2 F/2 G C/2 D2
V:A
E/2 F F/2 E2 A/2 B E/2 F2
V:T
E2, D4, F2,
Melody 1 & 2 ranking (out of 10)? 1
```

```
Bassline ranking (out of 10)? 0

NIL
[5]> ( mapcar #'music-melodies-rank selection )
(4 2 7 1)
[6]> ( mapcar #'music-bassline-rank selection )
(9 6 3 0)
[7]> ( mapcar #'music-rank selection )
(13 8 10 1)
[8]>
```

Updated Music Class

```
( defclass music ()
    (
        ( melody1 :accessor music-melody1 :initarg :melody1 )
        ( melody2 :accessor music-melody2 :initarg :melody2 )
        ( melody3 :accessor music-melody3 :initarg :melody3 )
        ( str-representation :accessor music-str-representation :initarg
:str-representation )
        ( rank :accessor music-rank :initarg :rank )
        ( bassline-rank :accessor music-bassline-rank :initarg
:bassline-rank )
        ( melodies-rank :accessor music-melodies-rank :initarg
:melodies-rank )
        ( num :accessor music-num :initarg :num )
    )
)
```

Code

```
; Global variable that sets the number of individuals selected from a
population.
( defconstant *selection-size* 4 )

; Method that selects the most-fit-individual from a list of candidates.
( defmethod select-individual ( ( p population )
    &aux i candidates rn )
    ( setf candidates ( select-individuals p ) )
```

```
    ( setf mfi ( most-fit-individual candidates ) )
    mfi
)

; Method that randomly selects a number of individuals from a population
; using *selection-size*.
( defmethod select-individuals ( ( p population )
    &aux individuals candidates rn )
    ( setf individuals ( population-individuals p ) )
    ( setf candidates () )
    ( dotimes ( i *selection-size* )
        ( setf rn ( random *population-size* ) )
        ( push ( nth rn individuals ) candidates )
    )
    candidates
)

; Method that handles the user-facing interactive selection process.
; -Lets the user rank the melody pairs and basslines of the number of
;  music samples defined by *selection-size*.
; -Includes error-handling for user input.
( defmethod interactive-selection ( ( selected-samples list ) &aux
melody1&2-rank bassline-rank)

    (cond
        (( null selected-samples )
            nil
        )
        (t
            ( setf current-sample ( car selected-samples ) )
            ( easyabc-display current-sample )
            ( format *query-io* "Melody 1 & 2 ranking (out of 10)? " )
            ( setf melody1&2-rank ( read-line *query-io* ) )

            ; ERROR-HANDLING for melody1&2-rank user input. Re-displays prompt
if
            ;  1. Input is not a number.
            ;  2. Input is less than 0.
            ;  3. Input is greater than 10.
            ( loop while ( or ( not ( ignore-errors ( parse-integer
melody1&2-rank ) ) )
                              ( < (parse-integer melody1&2-rank) 0 )
                              ( > (parse-integer melody1&2-rank) 10 ) )
```

```
                do ( format t "~%[ERROR] A ranking must be a number x such that
-1 < x < 11.~%")
                    ( format *query-io* "Melody 1 & 2 ranking (out of 10)? " )
                    ( setf melody1&2-rank ( read-line *query-io* ) )
            )


            ( format *query-io* "Bassline ranking (out of 10)? " )
            ( setf bassline-rank ( read-line *query-io* ) )

            ; ERROR-HANDLING for bassline-rank user input. Re-displays prompt
if
            ;  1. Input is not a number.
            ;  2. Input is less than 0.
            ;  3. Input is greater than 10.
            ( loop while ( or ( not ( ignore-errors ( parse-integer
bassline-rank ) ) )
                              ( < ( parse-integer bassline-rank ) 0 )
                              ( > ( parse-integer bassline-rank ) 10 ) )

                do ( format t "~%[ERROR] A ranking must be a number x such that
-1 < x < 11.~%")
                    ( format *query-io* "Bassline ranking (out of 10)? " )
                    ( setf bassline-rank ( read-line *query-io* ) )
            )

            ; Updates the ranks of the current sample.
            ( setf ( music-bassline-rank current-sample ) ( parse-integer
bassline-rank ) )
            ( setf ( music-melodies-rank current-sample ) ( parse-integer
melody1&2-rank ) )
            ( setf ( music-rank current-sample ) ( + ( parse-integer
bassline-rank ) ( parse-integer melody1&2-rank ) ) )

            ( terpri )
            ( interactive-selection ( cdr selected-samples ) )

        )
    )

)
```