This code defines a package named `PL_PIG_CHESS_INTERFACE` in Oracle's PL/SQL language. The package provides an interface for playing with the `PL_PIG_CHESS_ENGINE` chess engine and running test suites.

Here's a breakdown of the code:

- The package includes examples of how to use the interface to play against the engine at different levels, run test suites, and perform other operations.
- The `NEW_GAME` procedure initializes a new game with specified parameters, such as the engine strength/time usage for white and black players, the starting position, and the theory and interaction modes.
- The `DO_MOVE` procedure allows the user to make a move in the form of a string, e.g., 'e2e4' or 'g1f3'.
- The `DO_BOTMOVE` procedure makes a move for the engine, with an optional parameter to overrule the engine's move.
- The `DO_BOTGAME` procedure plays a full game with the engine, up to a specified maximum number of moves.
- The `SET_White` and `SET_Black` procedures alter the engine/player selection for the white and black players, respectively.
- The `TAKEBACK_MOVE` and `TAKEBACK_MOVES` procedures are not yet implemented and are intended to take back the last move or the last two moves, respectively.
- The package includes several procedures for running test suites, such as `test_BKtest`, `test_MSquickTest`, `test_THmyPosTest`, etc. Each procedure takes parameters for the engine level, the range of positions to test, and other options.
- The `test1` and `test2` procedures are also included, but their specific functionality is not described in the provided code snippet.

Method Signatures and Code Behavior

`NEW_GAME` Procedure

```
PROCEDURE NEW_GAME(
  White INTEGER DEFAULT 2,
  Black INTEGER DEFAULT 0,
  STARTPOSITION VARCHAR2 DEFAULT NULL,
  p_TheoryMode INTEGER DEFAULT 0,
  p_InteractionMode INTEGER DEFAULT 1
);
```

Behavior:

- Initializes a new game with specified parameters.
- Sets the engine strength/time usage for white and black players.
- Allows for a custom starting position to be provided.
- Configures the theory and interaction modes.

`DO_MOVE` Procedure

```
PROCEDURE DO_MOVE(fromto VARCHAR2);
```

Behavior:

- Allows the user to make a move in the form of a string, e.g., 'e2e4' or 'g1f3'.

`DO_BOTMOVE` Procedure

```
PROCEDURE DO_BOTMOVE(OverruleLevel SIMPLE_INTEGER DEFAULT 0);
```

Behavior:

- Makes a move for the engine.
- Allows for an optional parameter to overrule the engine's move.

`DO_BOTGAME` Procedure

```
PROCEDURE DO_BOTGAME(maxmoves SIMPLE_INTEGER DEFAULT 200);
```

Behavior:

- Plays a full game with the engine.
- Up to a specified maximum number of moves.

`SET_White` Procedure

```
PROCEDURE SET_White(White INTEGER DEFAULT 0);
```

Behavior:

- Alters the engine/player selection for the white player.

`SET_Black` Procedure

```
PROCEDURE SET_Black(Black INTEGER DEFAULT 0);
```

Behavior:

- Alters the engine/player selection for the black player.

`TAKEBACK_MOVE` Procedure

```
PROCEDURE TAKEBACK_MOVE;
```

Behavior:

- Takes back the last move (not yet implemented).

`TAKEBACK_MOVES` Procedure

```
PROCEDURE TAKEBACK_MOVES;
```

Behavior:

- Takes back the last two moves (not yet implemented).

Test Suite Procedures

- `test_BKtest`, `test_MSquickTest`, `test_THmyPosTest`, etc.:
  - Run a test suite with specified parameters.
  - Allow for customization of the engine level, position range, and other options.

`test1` and `test2` Procedures

- Specific functionality is not described in the provided code snippet.