client.erl

```erlang
%%%-------------------------------------------------------------------
%%% @author moritzspindelhirn
%%% @doc MOTD Client
%%%
%%% @end
%%% Created : 17. Mai 2014 13:19
%%%-------------------------------------------------------------------
-module(client).
-author("moritzspindelhirn").

%% API
-export([start/1]).

%% Imports
-import(werkzeug, [get_config_value/2]).
-import(timer, [apply_after/4]).

%% Defines
-define(TEAMNR, 99).
-define(PRAKNR, 3).

%%%-------------------------------------------------------------------
%%% Start client
%%% first set lifetime timer
%%% then start loop
%%%-------------------------------------------------------------------
start(Name) ->
  Sendintervall = getSendinterval(),
  SendintervallMs = Sendintervall * 1000,
  Lifetime = getLifetime(),
  LifetimeMs = Lifetime * 1000,
  timer:exit_after(LifetimeMs, "Time to say goodbye."),
  sendloop(Name, SendintervallMs, 0, 0, []).

%%%-------------------------------------------------------------------
%%% Loop for sending messages
%%%-------------------------------------------------------------------
sendloop(Name, Timeout, BatchNum, Count, SendMessageSL) ->
  % send after time
  apply_after(Timeout, client, prepAndSendMsg, [Name, Count, SendMessageSL]),
  BatchNum = BatchNum + 1,
  Count = Count + 1,
  % change timeout time after 5 messages
  if
    BatchNum > 5 ->
      Half = round(Timeout / 2),
      Diff = random:uniform(Half),
      % change must be at least 1 sek
      if
        Diff < 1000 -> Diff = 1000
      end,
      % random add or sub
      UpOrDown = random:uniform(),
      if
        UpOrDown >= 0.5 -> Timeout = Timeout + Diff;
        true -> Timeout = Timeout - Diff
      end,
      % check new timeout is at least 2000
      if
        Timeout < 2000 -> Timeout = 2000
      end,
      log(Name, "New intervall is is ~p", [Timeout]),
      % reset batch counter
      BatchNum = 0,
      % get unique message without send message ( Requirement 11. )
      getMessageId(Name, fun() -> noop end),
      % now read all messages
      getMessages(Name, SendMessageSL)
  end,
  sendloop(Name, Timeout, BatchNum, Count, SendMessageSL).

%%%-------------------------------------------------------------------
%%% Build the real message string and calls sendMessageWithId
%%%-------------------------------------------------------------------
prepAndSendMsg(Name, Nr, SendMessageSL) ->
  {ok, Hostname} = inet:gethostname(),
  SendParams = [Name, Hostname, self(), ?PRAKNR, ?TEAMNR, Nr, date()],
  % Misterious: C 769 (22)
  % 0-client@lab18-<0.1313.0>-C-1-03: 22te_Nachricht. Sendezeit: 16.05 18:01:30,769|(22)
  FormatStr = "~s@~s-~p-C-~d-~d: ~dte_Nachricht. Sendezeit: ~p",
  Msg = io_lib:format(FormatStr, SendParams),
  sendMessageWithId(Name, Msg, SendMessageSL).

%%%-------------------------------------------------------------------
%%% Send a message
%%% First get a new message ID from server
%%%-------------------------------------------------------------------
sendMessageWithId(Name, Msg, SendMessageSL) ->
  Servername = getServerName(),
  Server = global:whereis_name(Servername),
  getMessageId(Name, fun(Number) ->
      Server ! {new_message, {Msg, Number}},
      getMessageId(Name, fun(Number) ->
        Server ! {new_message, {Msg, Number}},
        werkzeug:pushSL(SendMessageSL, {Msg, Number}),
        log(Name, "Sending message ~s ( ID: ~p )", [Msg, Number])
      end)
  end).

%%%-------------------------------------------------------------------
%%% Request a new message ID from teh server
%%%-------------------------------------------------------------------
getMessageId(Name, Callback) ->
  % make sure callback is a alid function
  if
    not is_function(Callback) -> Callback = fun() -> noop end
  end,
  Servername = getServerName(),
  Server = global:whereis_name(Servername),
  % request new message id
  Server ! { query_msgid, self()},
  % receive new message and callback with it
  receive { msgid, Number} ->
    log(Name, "Received message number ~p", [Number]),
    Callback(Number)
  end.

%%%-------------------------------------------------------------------
%%% Get messages
%%%-------------------------------------------------------------------
getMessages(Name, SendMessageSL) ->
  Servername = getServerName(),
  Servername ! { query_messages, self()},
  receive
    { message, Number,Nachricht,Terminated} ->
      % Requirement 12. -> append ***** on messages send by this client
      if
        werkzeug:findSL(SendMessageSL, Number) = {-1, nok} ->
          log(Name, "Received: ~s ( ID: ~p )", [Nachricht, Number]);
        true ->
          append(Nachricht, "*******"),
          log(Name, "Received: ~s ( ID: ~p )", [Nachricht, Number])
      end,
      if
        not Terminated ->
          getMessages(Name, SendMessageSL)
      end
  end.

%%%-------------------------------------------------------------------
%%% Get Server name from config
%%%-------------------------------------------------------------------
```

```
getServerName() ->
  {ok, ConfigListe} = file:consult("client.cfg"),
  {ok, Servername} = get_config_value(servername, ConfigListe),
  Servername.

%%%-------------------------------------------------------------------
%%% Get sendinterval from config
%%%-------------------------------------------------------------------
getSendinterval() ->
  {ok, ConfigListe} = file:consult("client.cfg"),
  {ok, Sendintervall} = get_config_value(sendeintervall, ConfigListe),
  Sendintervall.

%%%-------------------------------------------------------------------
%%% Get livetime from config
%%%-------------------------------------------------------------------
getLifetime() ->
  {ok, ConfigListe} = file:consult("client.cfg"),
  {ok, Lifetime} = get_config_value(lifetime, ConfigListe),
  Lifetime.

%%%-------------------------------------------------------------------
%%% Logging
%%%-------------------------------------------------------------------
log(Name, Msg) ->
  Logfilename = io_lib:format("~s@~p.log", [Name, self()]),
  werkzeug:logging(Logfilename, Msg).
log(Name, Msg, Params) ->
  log(Name, io_lib:format(append(Msg, "\n"), Params)).
```