```perl
#! perl

########################
# GreetingApplication.pl
# Demonstration Greeting Generation Application
########################
#
# CVS Revision History
# $Author: swheat-cvsapp $
# $Revision: 1.1 $
# $Date: 2004/06/09 00:49:56 $
# $Source: /cvs/repositories/openii3/project/distributions/materials/openeai-examples/GreetingApplication.pl,v $
#
# This file is part of the OpenEAI sample, reference implementation,
# and deployment management suite created by Tod Jackson
# (tod@openeai.org) and Steve Wheat (steve@openeai.org) at
# the University of Illinois Urbana-Champaign.
#
# Copyright (C) 2003 The OpenEAI Software Foundation
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# For specific licensing details and examples of how this software
# can be used to implement integrations for your enterprise, visit
# http://www.OpenEai.org/licensing.
#
#########################
#
# This is a little script for use in demonstrating messaging from PERL.
# This script reads a file with names of people or things to greet
# and performs Greeting.Generate-Requests with that data and reports the
# results.
#
########################

use strict;

# Implementing messaging requires the ability to instatiate Java objects.
# Here we have used the PERL Java module available on CPAN.  This module
# requires "no strict 'subs'".
use Java;
no strict 'subs';

```

```
use vars qw($startTime $processingStartTime $endTime
            $java $docUri $appName $INFILE $aConfig $i $j $p2p
            $processingTime $totalTime);
#use subs qw(&process_input_line);

# GLOBAL CONFIGURATION
$docUri = "configs/messaging/Environments/Examples/Deployments/".
          "AnyOpeneaiEnterprise.xml";

$appName = "org.any-openeai-enterprise.PerlGreetingApplication";

$INFILE = "configs/messaging/Environments/Examples/".
          "InputFiles/GreetingApplication/greetees.txt";

# main program flow

# get the start time
$startTime = time;

$java = new Java();
eval {
  # create an AppConfig object
  $aConfig = $java->create_object("org.openeai.config.AppConfig", $docUri,
                                  $appName);

  # get a p2p producer from AppConfig to use
  $p2p = $aConfig->getObject("P2pEgsProducer");

};
 if ($@) {
  # An exception was thrown!!
  $@ =~ s/ERROR: //;      # Gets rid of 'ERROR: '
  $@ =~ s/at $0.*$//;     # Gets rid of 'croak' generated stuff
  # Print just the Java stuff
  print "$@\n";
};

print "\n";
print localtime(time)." Opening file $INFILE.\n";
open INFILE or die "Can't open file $INFILE: $!\n";

# get the start of processing time
$processingStartTime = time;

$i=1;
while(<INFILE>) {
  &process_input_line($i,$_,$p2p);
  $i++;
}

close INFILE;

# get the end time
$endTime = time;

# summarize and exit
```

```
# calculate total time and processing time
$processingTime = $endTime - $processingStartTime;
$totalTime = $endTime - $startTime;

print "\n";
print localtime(time)." Summarizing...\n";
print localtime(time)." Application start time: ".localtime($startTime)."\n";
print localtime(time)."  Processing start time: ".localtime($processingStartTime)."\n";
print localtime(time)."    Processing end time: ".localtime($endTime)."\n";
print localtime(time)."        Processing time: $processingTime seconds\n";
print localtime(time)."   Application run time: $totalTime seconds\n";
print localtime(time)." Exiting.\n\n";

exit;

##################################
# subroutine process_input_line
# args: input file line
##################################
# more comments here
##################################

sub process_input_line ($$) {

  my $inputLineNum = $_[0];
  my $inputLine = $_[1];
  my $p2p = $_[2];
  my $i = 0;

  #print "Processing line $inputLineNum: $inputLine\n";

  # get rid of the line break
  chop($inputLine) if $inputLine =~ /\n$/;

  eval {
    my $greeting = $aConfig->getObject("Greeting");
    my $greetee = $aConfig->getObject("Greetee");
    my $v = $java->create_object("java.util.Vector");

    $greetee->setFullName($inputLine);

    print "\n";
    print localtime(time)." Sending Greeting.Generate-Request for: ".
          $inputLine."\n";

    # generate the greeting
    $v = $greeting->generate($greetee, $p2p);

    # display the greeting that was generated by EGS
     $i=0;
     while ($i < $v->size->get_value) {
       $greeting = $v->get($i);
        print localtime(time)." Greeting returned for '$inputLine' is: ".
              $greeting->getText->get_value;
        $i++;
    }
```

```
    };
      if ($@) {
      # An exception was thrown!!
      $@ =~ s/ERROR: //;      # Gets rid of 'ERROR: '
      $@ =~ s/at $0.*$//;     # Gets rid of 'croak' generated stuff
      # Print just the Java stuff
      print "$@\n";
    };
  }

```