```
/*
   Copyright 1993, General Magic.  All Rights Reserved.

   DESCRIPTION

     simple factorial: computes factorial(n) using each of the Telescript
     loop constructs plus a recursive solution

     Recall:  factorial(6) = 6! = 6 * 5 * 4 * 3 * 2 * 1 = 720
              factorial(5) = 5! = 5 * 4 * 3 * 2 * 1 = 120
              etc.

*/

/*  Using the for looping construct */

do {
   n: Integer = 6;
   fact: Integer = 1;

   for i: Integer to n {fact = fact*i};
   n.dump(); fact.dump();
};

/*  Using the while looping construct */

do {

};

/*  Using the 'loop' looping construct */

do {

};

/*  Using the 'repeat' looping construct */

do {

};


// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```

```
/*
   Copyright 1993, General Magic.  All Rights Reserved.

   DESCRIPTION

     list reversal: reverses the order of elements in a Telescript List object

*/

do {
  // initialization of a List object
  a_list: List[Integer,Equal] = List[Integer,Equal](3, 1, 2, 4, 5);

  a_list.dump();  // dump the contents of the unsorted List object

  // reverse the elements of the List here

  a_list.dump();  // dump the contents of the sorted List object
};


// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```

```
/*
  Copyright 1993, General Magic.  All Rights Reserved.

  DESCRIPTION

  integerCalc:  calculates simple arithmetic functions for integer values.

*/

*@Place.publicPackages.include(IntegerCalc: module = (

  Calculator: class (Object) = (
   public
    result: Integer;                  // a public attribute
   private
    op1, op2: Integer;
    func: String;
   public                          // integer arithmetic functions

    initialize: op(v: Integer; ...) = {result = v; ^};  // initial value for result

    specialdump: op() = {
      String (String("For the calculator:\n\tresult is "), result.asString(), String(".\n"),
              String("\tfirst operand is "), op1.asString(), String(".\n"),
              String("\toperation is "), func, String(".\n"),
              String("\tsecond operand is "), op2.asString(), String(".") ).dump();
    };
    plus:  op(x, y: Integer) Integer = {        // integer addition
      op1 = x; op2 = y; func = "plus";
      result = x+y
    };
    /*  Here are some other ways to return a result
      plus:  op(x, y: Integer) Integer = {
        result = x+y;
        return(result);
      };
      Or
      plus:  op(x, y: Integer) Integer = {
        x+y;
      };
      Or
      plus:  op(x, y: Integer) Integer = {
        return(x+y);
      };
      Or
      plus:  op(x, y: Integer) Integer = {
        return(result = x+y)
      };
    */
    minus: op(x: Integer; y: Integer) Integer = {   // integer subtraction
      op1 = x; op2 = y; func = "minus";
      result = x-y
    };
    times: op(x: Integer; y: Integer) Integer = {   // integer multiplication
      op1 = x; op2 = y; func = "times";
      result = x*y
    };
    divide: op(x: Integer; y: Integer) Integer = {  // a PROMISE for integer division
      op1 = x; op2 = y; func = "divide";
      result = (x/y)@;  // force compiler to accept a possible
                // real number value assigned to the integer
                // result variable.  Note that at runtime,
                // it is up to the programmer to insure that
                // the value is an integer, else a runtime
                // ResultInvalid error will occur.

                // the engine DOES return a Real for that division
                // hence we will see InvalidResult
    };
    quotient: op(x: Integer; y: Integer) Integer = {  // integer division
      result = x;
    };
    modulus: op(x: Integer; y: Integer) Integer = {   // modulus
      result = x;
    };
    divideFloor: op(x: Integer; y: Integer) Integer = { // divide and floor to an integer value
      result = x;
    };
    plusminus: op(x: Integer) Integer = {     // integer negation
      result = x;
    };
  );

  Test: class (Object) = {

  public
    initialize: op(...) = {^; the_calc = Calculator(0)};
```

```
    the_calc: Calculator;
    test: op() = {
      the_calc.plus(5,6).dump(); the_calc.specialdump();
      the_calc.minus(3,7).dump(); the_calc.specialdump();
      the_calc.times(5,6).dump(); the_calc.specialdump();
      the_calc.plusminus(8).dump(); the_calc.specialdump();

      try {                     // exception handling
        the_calc.divide(6,0).dump(); the_calc.specialdump();
      } catch DivisionByZero {            // catch division by zero
        "Can't divide by zero, remember?".dump();
      } catch error: Exception {            // catch any other exceptions
        error.dump();
      };

      try {
        the_calc.divide(6,2).dump(); the_calc.specialdump();
              // this generates an ResultInvalid error
              // since an integer divide returns a Real, not an Integer
              // The Language reference implies otherwise but that
              // is a bug in the language reference.
      } catch DivisionByZero {
        "Can't divide by zero, remember?".dump();
      } catch error: Exception {
        error.dump();
      };
      try {
        the_calc.divideFloor(6,4).dump(); the_calc.specialdump();
      } catch DivisionByZero {
        "Can't divide by zero, remember?".dump();
      } catch error: Exception {
        error.dump();
      };

      the_calc.quotient(6,2).dump(); the_calc.specialdump();

      the_calc.quotient(6,4).dump();
              // generates message and result is set to 0
      the_calc.specialdump();

      the_calc.modulus(6,4).dump(); the_calc.specialdump();
    };
  );
)     // end of module
);      // end of module include into publicPackages of BootPlace

do {
  << [:Test] test >>
}


// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```

```
/*
  Copyright 1993, General Magic.  All Rights Reserved.

  DESCRIPTION

  stringConvert:  converts values of many different types to strings.

  StrVal class:  makes formatted output possible and easy!

           uses dump2Chars() to make better formatted output

*/


/*
  Study this program.  Note especially how the StrValDemo class uses the
  StrVal class to create formatted output.  You don't have to study the
  code for StrVal but do make sure that you can use the StrVal class to
  create formatted output in your own programs.

  StrVal can be used to create String objects from strings and from the
  values in other types of objects (like integers, reals, lists, telenames
  and more).

  For performance you might consider using dump2Chars() directly
  for those types which are already formatted nicely by the engine.

*/

*@Place.publicPackages.include(StringConversion: module = (

  StrVal: class (Object) = (

//  For performance you might consider using dump2Chars() directly
//  for those types which are already formatted nicely by the engine.


      public
          value:  String;


          initialize: op (object:  protected Object) = (
              theFinding : List[Integer,Equal] | Nil;
              startPos, theLength : Integer;

              ^;

              if object is Number {
                  value = object@Number.asString();
              }
              else if object is String {
                  value = object@String.copy();
              }
              else if object is Identifier {
                  value = object@Identifier.asString();
              }
              else if object is ClassName {
                  value = object@ClassName.classDigest.asString();
              }
              else if object is Class {
                  value = object@Class.name.classDigest.asString();
              }
              else if object is Time {
                  value = object.dump2Chars();
              }
              else  {
                  value = object.dump2Chars();

                  // For anything but primitives return now.

                  if !(object is Primitive) && !(object is OctetString!)
                          && !(object is Identifier) {
                      return;
                  };
                  theFinding = Pattern("<").find(value,nil);
                  if theFinding != nil {
                      startPos = theFinding[1] + 1;
                      theLength = value.length;
                      if startPos < theLength {
                          value = value.substring(startPos,theLength);
                      };
                  };
              };
          );
  );


  StrValDemo: class (Object) = (
```

```
    public
      initialize: op() = {
        ^;
        myTelename = Telename ($"4A6F686E204164656C7573");
        myInt = 5;
      };

    private
      myTelename: Telename!;      // the ! syntax means that only an instance of class Telename
                    // is allowed and NOT an instance of any *subclass* of Telename
      myInt: Integer!;

    public
      dumpThenFormat: op() = {
        "----------- begin dump -----------".dump();
        self.dumpStuff();
        "----------- end of dump -----------".dump();
        "----------- begin format -----------".dump();
        self.formatStuff();
        "----------- end format -----------".dump();
      };

      generateError: op() = {
        "----------- begin error -----------".dump();
        self.errorStuff();
        "-----------  end error  -----------".dump();
      };

      otherStuff: op() = {
        "----------- begin other -----------".dump();
        here.dump();          // the current place
        process.dump();         // the current process
        self.dump();          // the current object
        client.dump();          // the object that invoked the current method
        "-----------  end other  -----------".dump();
      };

    private
      dumpStuff: op() = {
        "hello world!".dump();
        2.dump();
        'a'.dump();
        $ff.dump();
        $"234897".dump();
        %"01001001".dump();
        String("hello ", "again ", "world").dump();

        do {
          the_str:String = "hello world";

          String("the_str = ", the_str, ".").dump();

        }
      };

      formatStuff: op() = {
        String("The string is ", "hello world".dump2Chars(), ".").dump();
        String("The integer is ", 2.dump2Chars(), ".").dump();
        String("The char is ",  'a'.dump2Chars(), ".").dump();
        String("The octet is ", $ff.dump2Chars(), ".").dump();
        String("The octet string is ", $"234897".dump2Chars(), ".").dump();
        String("The bit string is ", %"01001001".dump2Chars(), ".").dump();

        String("myInt is ", myInt.dump2Chars(), ".").dump();
        String("myTelename is ", myTelename.dump2Chars(), ".").dump();

// dump2Chars formats most things pretty nicely but otherwise can use StrVal

// StrVal will format many more things nicely but there is some execution overhead
// which you may want to avoid

        String("The char is ", StrVal('a').value, ".").dump();
        String("The octet is ", StrVal($ff).value, ".").dump();
        self.dump();
      };

      errorStuff: op() = {
        try {
          String("The integer is ", 2, ".").dump();
        } catch error:Exception {
          error.dump();   // This should generate ArgumentInvalid
        };
        /* all these will get errors too
        String("The char is ", 'a', ".").dump();
        String("The octet is ", $ff, ".").dump();
        String("The octet string is ", $"234897", ".").dump();
        String("The bit string is ", %"01001001", ".").dump();
        String("myInt is ", myInt, ".").dump();
```

```
        String("myTelename is ", myTelename, ".").dump();
        */
      };


  ); // end of Demo class definition

)  // end of StringConversion module
);    // end of module include into publicPackages of BootPlace

do {

  << [ :StrValDemo ] ref ref dumpThenFormat generateError otherStuff>>;

};


/*
** Result:
**


String: <------------ begin dump ------------>
String: <hello world!>
Integer: <2>
Character: <97(a)>
Octet: <ff>
OctetString: <234897>
BitString: <01001001>
String: <hello again world>
String: <the_str = hello world.>
String: <------------ end of dump ------------>
String: <------------ begin format ------------>
String: <The string is hello world.>
String: <The integer is 2.>
String: <The char is 97(a).>
String: <The octet is ff.>
String: <The octet string is 234897.>
String: <The bit string is 01001001.>
String: <myInt is 5.>
String: <myTelename is 4A6F686E204164656C7573, Nil.>
String: <The char is 97(a).>
String: <The octet is ff.>
StrValDemo
String: <------------ end format ------------>
String: <------------ begin error ------------>
ArgumentInvalid
String: <------------  end error  ------------>
String: <------------ begin other ------------>
Nil
BootPlace: <Telename: <47656E6572616C204D6167696320496E63, F245643EC47B13A1139589961AD7596E>(0 occupants)>
StrValDemo
BootPlace: <Telename: <47656E6572616C204D6167696320496E63, F245643EC47B13A1139589961AD7596E>(0 occupants)>
String: <------------  end other  ------------>


*/


// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```

```
/*
  Copyright 1993, General Magic.  All Rights Reserved.

  DESCRIPTION

  simpleAgent:  a very SimpleAgent object goes to a DestinationPlace object.  Once there,
    it does very little and then returns to the OriginPlace object whence it came.

*/


*@Place.publicPackages.include(SimpleModule: module = (   // module definition, all classes defined here

  StrVal: class (Object) = (

//  For performance you might consider using dump2Chars() directly
//  for those types which are already formatted nicely by the engine.


      public
          value:  String;

          initialize: op (object:  protected Object) = {
              theFinding : List[Integer,Equal] | Nil;
              startPos, theLength : Integer;

              ^;

              if object is Number {
                  value = object@Number.asString();
              }
              else if object is String {
                  value = object@String.copy();
              }
              else if object is Identifier {
                  value = object@Identifier.asString();
              }
              else if object is ClassName {
                  value = object@ClassName.classDigest.asString();
              }
              else if object is Class {
                  value = object@Class.name.classDigest.asString();
              }
              else if object is Time {
                  value = object.dump2Chars();
              }
              else  {
                  value = object.dump2Chars();

                  // For anything but primitives return now.

                  if !(object is Primitive) && !(object is OctetString!)
                          && !(object is Identifier) {
                      return;
                  };
                  theFinding = Pattern("<").find(value,nil);
                  if theFinding != nil {
                      startPos = theFinding[1] + 1;
                      theLength = value.length;
                      if startPos < theLength {
                          value = value.substring(startPos,theLength);
                      };
                  };
              };
          };
  );


  Pair:  class[classOne,classTwo:  Class] (Object) = (  // Pair is a handy TAL class
              // Here the class of the attributes are parameterized
              // so that when a Pair is created, the class of object1 and object2
              // are specified - see usage of Pair later in this script

              // The HTS compiler enforces the constraints on the objects stored
              // in those attributes, the Telescript engine only requires the
              // stored objects to be a member of class Object.
      public
          object1:  classOne;
          object2:  classTwo;


          initialize: op(arg1: classOne;  arg2:  classTwo) = {
              ^;
              object1 = arg1;
              object2 = arg2;
          };
  );
```

```
OriginPlace: class (EnginePlace) = {  // class definition for OriginPlace
                        // An OriginPlace object creates a
                        // DestinationPlace object and a
                        // SimpleAgent object.
  public
    initialize: op(bootPlacePkgs:List[Package[Identifier,Class,Equal], Equal];
      Telename; Teleaddress; Integer|Nil; Integer|Nil) = {
      for pkgIndex: Integer to bootPlacePkgs.length {   // Here we take each package in the list passed
                                // into initialize and include it in the list
                                // of packages in the place's publicPackages attribute.
                                // See PackageProcess class definition.
                                // include is an operation on Collections and
                                // therefore inherited by List.
                  *.publicPackages.include(bootPlacePkgs[pkgIndex]);

        ^;        // escalate AFTER including the packages into the publicPackages of
              // the Engine Place
      };

    };
    live: sponsored op(Exception|Nil) = {
      DestinationPlace(Permit(1000), nil, nil, nil, "destination");   // Create a destination place within
                                          // the OriginPlace
      SimpleAgent(Permit(1000), nil, nil);  // Create the agent

      loop { }; // Loop, waiting for something to happen...
    };
    entering: sponsored op(subjectName: protected Telename; subjectClass: protected ClassName!; unprotected P
      String("Something entering the OriginPlace: ",
        subjectName.dump2Chars()).dump() ;
      return Pair[Telename,ClassName](subjectName.copy(), subjectClass.copy());
      // entering requires an object returned
      // here we return a Pair containing the Telename and Classname
      // of the object entering the OriginPlace
      // These are copied so that the name and class in the Pair don't go
      // Void when the object which entered goes away
    };

);                        // end of OriginPlace class definition

SimpleAgent: class (Agent) = {          // class definition for SimpleAgent class
                        // A SimpleAgent object goes to the
                        // DestinationPlace object and then
                        // returns to the OriginPlace object.
  public
    initialize: op(Permit; Permit|Nil; Integer|Nil) = {
      ^;
    };

    live: sponsored op(Exception|Nil) = {
      try {
        *.go(Ticket( Telename("destination".asOctetString(),nil)));
      } catch err: Exception {
        String("Exception when SimpleAgent object tries to go ",
            "to the DestinationPlace object.\n",
            StrVal(err).value);
      };

      here.dump();            // where am i now?

      try {
        // return to the OriginPlace object
        *.go(Ticket( Telename("origin".asOctetString(),nil)));
      } catch err: Exception {
        String("Exception when SimpleAgent object tries to go ",
            "to the OriginPlace object.\n",
            StrVal(err).value);
      };

      here.dump();            // where am i now?

      loop { };             // stay here forever
    };
);                        // end of class SimpleAgent

DestinationPlace: class (Place) = {       // class definition for the DestinationPlace class
                        // A DestinationPlace object just loops
                        // around forever
  public
    initialize: op(Permit; Permit|Nil; Integer|Nil; String|Nil; String|Nil;...) = {
      ^;
    };

    live: sponsored op(Exception|Nil) = {
      loop {  };
    };
```

```
      entering: sponsored op(subjectName: protected Telename; subjectClass:protected ClassName! ;unprotected Pe
        String("Agent entered the Destination: ",
          subjectName.dump2Chars()).dump();
        return Pair[Telename,ClassName](subjectName.copy(), subjectClass.copy());
        // entering requires an object returned
        // here we return a Pair containing the Telename and Classname
        // of the object entering the OriginPlace
        // These are copied so that the name and class in the Pair don't go
        // Void when the object which entered goes away
      );

  );                        // end of class DestinationPlace
)                         // end of module SimpleModule
);    // end of module include into publicPackages of BootPlace
do {
  << [ #NIL #NIL [ :Teleaddress ] [ #NIL "origin" asOctetString :Telename]
  #self       // this is the low telescript equivalent of 'self'
            // which is the boot place in this case
  publicPackages    // get the public packages list from the boot place
  :OriginPlace] discard >>
}


/*
** Result:
**

String: <Something entering the OriginPlace: 64657374696E6174696F6E, 4F559EA9B30516936BBFAA60027545D6>
String: <Something entering the OriginPlace: 6167656E74, 1FC544E0CD30A32CAB6CCA3AE501ABB3>
String: <Agent entered the Destination: 6167656E74, 1FC544E0CD30A32CAB6CCA3AE501ABB3>
DestinationPlace: <Telename: <64657374696E6174696F6E, 4F559EA9B30516936BBFAA60027545D6>(1 occupant)>
        SimpleAgent: <Telename: <6167656E74, 1FC544E0CD30A32CAB6CCA3AE501ABB3>>
String: <Something entering the OriginPlace: 6167656E74, 1FC544E0CD30A32CAB6CCA3AE501ABB3>
OriginPlace: <Telename: <6F726967696E, A241709DADCD87D0EB811366568C9D53>(2 occupants)>
        DestinationPlace: <Telename: <64657374696E6174696F6E, 4F559EA9B30516936BBFAA60027545D6>(0 occupants)>
        SimpleAgent: <Telename: <6167656E74, 1FC544E0CD30A32CAB6CCA3AE501ABB3>>


The first line is the DestinationPlace entering the OriginPlace.
The second line is the agent entering the OriginPlace.
Next the agent enters the destination and dumps the current place.
Then the agent goes on its trip and returns to the OriginPlace and
dumps the current place.

*/

// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```

```
/*
   Copyright 1993, General Magic.  All Rights Reserved.

  DESCRIPTION

  stringDelivery:  StringBearer objects carry strings from the StringStart place to the
    StringDump place in order to meet with a StringDumpster object there and deliver
    the string to it.

*/

*@Place.publicPackages.include(StringDelivery: module = (     // module definition

// Traveled is a mixin for Place
// when an exception occurs when attempting to 'go' to a destination
// the agent is sent to the closest enclosing Traveled place. The
// Traveled place is responsible for redirecting the agent appropriately.
// In a real service, it would be redirected to purgatory.

// Agents are sent into Traveled places in a ShippingBox, hence the need
// ShippingBox interface definitions. These are not formally
// part of the Telescript language so their interfaces need to be described
// to the High Telescript compiler. They are built into the Telescript
// engine, however.

ShippingBox: sealed interface (Object, Unmoved) = (
     public
       destinations: readonly protected List[Ticket, Equal];

       exception: readonly TripException|Nil;

       origin: readonly protected Way;

       time: readonly Time;

       peek: op(identifier: Identifier) copied Object
       throws FeatureUnavailable;

       poke: op(identifier: Identifier; attribute: copied Object)
       throws ArgumentInvalid, FeatureUnavailable;

       redirect: op(exception: TripException|Nil; newDestination: copied Ticket|Nil);
     public
       initialize: op()
       throws FeatureUnavailable;
   );


Traveled: mixin interface (Place) = (
     private
       transferOut: op(
         box: unprotected ShippingBox;
         nextHop: protected Way|Nil;
         deliverBefore: Integer|Nil;
                 travelPackages: List[Package, Equal]|Nil);

        private
                travelPackages: List[Package, Equal];


     public
       transferredIn: sponsored op(
         box: unprotected ShippingBox;
         previousHop: protected Way|Nil;
         outboundTransfer: Boolean);
   );

  StrVal: class (Object) = (

//  For performance you might consider using dump2Chars() directly
//  for those types which are already formatted nicely by the engine.


      public
          value:  String;


          initialize: op (object:  protected Object) = {
              theFinding : List[Integer,Equal] | Nil;
              startPos, theLength : Integer;

              ^;

              if object is Number {
                  value = object@Number.asString();
              }
              else if object is String {
                  value = object@String.copy();
```

```
            }
            else if object is Identifier {
                value = object@Identifier.asString();
            }
            else if object is ClassName {
                value = object@ClassName.classDigest.asString();
            }
            else if object is Class {
                value = object@Class.name.classDigest.asString();
            }
            else if object is Time {
                value = object.dump2Chars();
            }
            else  {
                value = object.dump2Chars();

                // For anything but primitives return now.

                if !(object is Primitive) && !(object is OctetString!)
                        && !(object is Identifier) {
                    return;
                };
                theFinding = Pattern("<").find(value,nil);
                if theFinding != nil {
                    startPos = theFinding[1] + 1;
                    theLength = value.length;
                    if startPos < theLength {
                        value = value.substring(startPos,theLength);
                    };
                };
            };
        };
);

StringEnginePlace: class (EnginePlace,Traveled) = (

  public
    initialize: op(bootPlacePkgs:List[Package[Identifier,Class,Equal], Equal]; Telename; Teleaddress; Integer
      for pkgIndex: Integer to bootPlacePkgs.length {
                *.publicPackages.include(bootPlacePkgs[pkgIndex]);
      };
      ^;        // escalate AFTER including the packages into the publicPackages of
              // the Engine Place

  };

/*
  When an agent goes to a place through a Traveled place, the
  transferredIn method of the Traveled place is invoked.

  When an exception occurs while attempting to 'go' to a destination
  the agent is sent to the closest enclosing Traveled place. The
  Traveled place is responsible for redirecting the agent appropriately.

  In a real service, it would be redirected to purgatory. In this case, when
  transferredIn is called and there is an exception, the shipping box is redirected
  to THIS traveled place. That way the agent will live and can take alternative action.

*/

    transferredIn: sponsored op(box: unprotected ShippingBox; hop: protected Way|Nil; transfer: Boolean) = {

      if (box.exception != nil)
      {
        box.redirect(box.exception,Ticket(*.name));   // make the agent go to here
      };
        *.transferOut(box, nil, nil,nil);
    };


    live: sponsored op(Exception|Nil) = {
      StringDump(Permit(2000), nil, nil, nil, "THE STRINGDUMP");

      StringStart(Permit(2000), nil, nil, nil, "THE STRINGSTART");
          // process is a reserved word that returns the current process
      String("The current process for StringEnginePlace's live:\n",
          StrVal(process).value,
          "\n").dump();
      loop { };
    };

    entering: sponsored op(the_Telename: protected Telename; protected ClassName!; unprotected Permit; protec
      String("Agent entering the StringEnginePlace: ", the_Telename.dump2Chars() ).dump();
      return nil;
    };

);                      // end of StringEnginePlace class definition
```

```
StringBearer: class (Agent, MeetingAgent) = {   // class definition
  public
    initialize: op(str: owned String; Permit; Permit|Nil; Integer|Nil; String|Nil) = {
      the_string = str;
      ^;
    };

    live: sponsored op(Exception|Nil) = {
      dumpster_petition: Petition;

      *.go(Ticket( Telename("THE STRINGDUMP".asOctetString(),nil)));

      loop {
        try {
          the_contact: Agent;

          dumpster_petition = Petition( Telename("THE STRINGDUMPSTER".asOctetString()),nil,0);
          the_contact = here@StringDump.meet(dumpster_petition, nil);
                // here is a reserved word that returns the current place
          the_contact@StringDumpster.deliver(the_string);
          here@StringDump.part(the_contact.name);
          break;
        } catch MeetingException {
          String("Meeting Exception for:", the_string).dump();
          break;
        };
      };
    };
  private
    the_string: String;
};                          // end of class StringBearer

StringDumpster: class (Agent, MeetingAgent,EventProcess) = {  // class definition
  public
    initialize: op(Permit; Permit|Nil; Integer|Nil; String|Nil) = {
      string_list = List[String,Equal]();
      ^;
      *.enableEvents(PartEvent(nil,nil));   // register interest in these events
    };

    live: sponsored op(Exception|Nil) = {
      loop {

      *.getEvent(nil,nil);    // when PartEvent is received, print the strings,
                    // otherwise block

      *.printStrings("The parting strings:\n\t");

      };  // Loop, waiting for something to happen...
    };

    meeting: sponsored op(protected Telename; protected ClassName!; protected Petition) Object|Nil = {
      self.printStrings("The meeting strings:\n\t");
      return nil;
    };

    deliver: op(str: copied String) = {
    // this is copied because we want our own copy of the string
      string_list.add(1, str);
    };

  private
    string_list: List[String,Equal];

  printStrings: op(str: String) = {
    String("\n------------------------------\n",
        str,
        string_list.dump2Chars(),
        "\n------------------------------").dump();
  };
};                          // end of class StringDumpster

StringDump: class (Place, MeetingPlace,EventProcess) = {  // class definition
  public
    initialize: op(Permit; Permit|Nil; Integer|Nil; OctetString|Nil; String|Nil) = {
      ^;

      *.enableEvents(ExitEvent(nil,nil));   // register interest in these events

    };

    live: sponsored op(Exception|Nil) = {

      StringDumpster(Permit(500), nil, nil, "THE STRINGDUMPSTER");

      loop {
```

```
          *.reportExit(*.getEvent(nil,nil)@ExitEvent.record);

        };
      };

      entering: sponsored op(the_contactTeleName: protected Telename; the_class: protected ClassName!; unprotec

        String("Agent entered the Dump: ", the_contactTeleName.dump2Chars()).dump();
        return the_contactTeleName.copy() ;
      };

    private

      reportExit: op(the_contactTeleName: Object) = {
        String("Agent exited the Dump: ", the_contactTeleName.dump2Chars()).dump();
      };
  );                        // end of class StringDump

  StringStart: class (Place) = {          // class definition
    public
      initialize: op(Permit; Permit|Nil; Integer|Nil; OctetString|Nil; String|Nil) = {
        ^;
      };

      live: sponsored op(Exception|Nil) = {
        StringBearer("------ West Side Story", Permit(500), nil, nil,  "Alfred").dump();
        StringBearer("To your last dying day", Permit(500), nil, nil,  "Barney").dump();
        StringBearer("From your first cigarette", Permit(500), nil, nil,  "Calvin").dump();
        StringBearer("You're a Jet all the way", Permit(500), nil, nil,  "Donald").dump();
        StringBearer("When you're a Jet", Permit(500), nil, nil,  "Egbert").dump();

        loop { }; // wait forever
      };

      entering: sponsored op(the_name: protected Telename ; the_class: protected ClassName!; the_permit: unprot

        if (the_ticket != nil) {
          String("Agent entering the StringStart: ", the_name.dump2Chars()).dump();
        };
        return nil;
      };


  );            // end of class StringStart
)       // end of module StringDelivery
); // end of module include into publicPackages of BootPlace


do {
  << [ [:Teleaddress]  [:Telename] #self publicPackages :StringEnginePlace] discard >>;
};
                                        get publicPackage

/*
**  Result:
**
String: <Agent entering the StringEnginePlace: 5448452053545249 4E4744554D50, D16D04CEFC1AC0415392CCA98F3DBF05>
String: <Agent entering the StringEnginePlace: 5448452053545249 4E475354415254, E872548DBA7D21DE573B3FF802F75129
String: <The current process for StringEnginePlace's live:
StringEnginePlace: <[T]47656E6572616C204D6167696320496E63, B38080EA74D6DFAA8829792A638B07C8(2 occupants)>
        StringStart: <5448452053545249 4E475354415254, E872548DBA7D21DE573B3FF802F75129(0 occupants)>
        StringDump: <5448452053545249 4E4744554D50, D16D04CEFC1AC0415392CCA98F3DBF05(0 occupants)>>
StringBearer: <Telename: <416C66726564, AD2BAB1D6E91F44DA02C49C7EB70875A>>
StringBearer: <Telename: <4261726E6579, 28E242107E93699CC57D3335EA0B8ACD>>
StringBearer: <Telename: <43616C76696E, 5A4EB14C9EEAA75C93D1F8FF27DD8DC0>>
StringBearer: <Telename: <446F6E616C64, 5C1333557A20A45F05B781DB3159A95E>>
StringBearer: <Telename: <456762657274, FB7293D4B607301E285B5DA0966140D3>>
TwoStringBearer: <Telename: <41726E6F6C64, 0822F4B571104BB7CC6AAC5F69A4E537>>
TwoStringBearer: <Telename: <42656E6A616D696E, B62011FF7CF6BD3AFB044E079445251B>>
TwoStringBearer: <Telename: <436F726579, 9FE2B5D6E1DA4F0E0DC3C25300B4900A>>
String: <Agent entered the Dump: 416C66726564, AD2BAB1D6E91F44DA02C49C7EB70875A>
String: <
------------------------------
The meeting strings:
        List: <0 elements>
------------------------------>
String: <Agent entering the StringStart: 416C66726564, AD2BAB1D6E91F44DA02C49C7EB70875A>
String: <Agent entered the Dump: 4261726E6579, 28E242107E93699CC57D3335EA0B8ACD>
String: <
------------------------------
The meeting strings:
        List: <1 element>
        ------ West Side Story
------------------------------>
String: <Agent entering the StringStart: 4261726E6579, 28E242107E93699CC57D3335EA0B8ACD>
String: <Agent entered the Dump: 43616C76696E, 5A4EB14C9EEAA75C93D1F8FF27DD8DC0>
String: <
------------------------------
```

```
The meeting strings:
        List: <2 elements>
        To your last dying day
        ------ West Side Story
------------------------------>
String: <Agent entering the StringStart: 43616C76696E, 5A4EB14C9EEAA75C93D1F8FF27DD8DC0>
String: <Agent entered the Dump: 446F6E616C64, 5C1333557A20A45F05B781DB3159A95E>
String: <
------------------------------
The meeting strings:
        List: <3 elements>
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <Agent entering the StringStart: 446F6E616C64, 5C1333557A20A45F05B781DB3159A95E>
String: <Agent entered the Dump: 456762657274, FB7293D4B607301E285B5DA0966140D3>
String: <
------------------------------
The meeting strings:
        List: <4 elements>
        You're a Jet all the way
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <Agent entering the StringStart: 456762657274, FB7293D4B607301E285B5DA0966140D3>
# scr (3) Mon 17:26:15: Unable to deliver occupant: OccupancyDenied
String: <Agent entering the StringEnginePlace: 41726E6F6C64, 0822F4B571104BB7CC6AAC5F69A4E537>
String: <Exception on go to StringDump for: 41726E6F6C64, 0822F4B571104BB7CC6AAC5F69A4E537
        OccupancyDenied>
StringEnginePlace: <[T]Telename: <47656E6572616C204D6167696320496E63, B38080EA74D6DFAA8829792A638B07C8>(3 occup
        TwoStringBearer: <Telename: <41726E6F6C64, 0822F4B571104BB7CC6AAC5F69A4E537>>
        StringStart: <Telename: <54484520535452494E475354415254, E872548DBA7D21DE573B3FF802F75129>(2 occupants)
                TwoStringBearer: <Telename: <42656E6A616D696E, B62011FF7CF6BD3AFB044E079445251B>>
                TwoStringBearer: <Telename: <436F726579, 9FE2B5D6E1DA4F0E0DC3C25300B4900A>>
        StringDump: <Telename: <5448452053545249 4E4744554D50, D16D04CEFC1AC0415392CCA98F3DBF05>(1 occupant)>
                StringDumpster: <Telename: <54484520535452494E4744554D5053544552, 24F2401DF8AE68D0D8091C1FE2D65
# scr (3) Mon 17:26:16: Unable to deliver occupant: OccupancyDenied
String: <Agent entering the StringEnginePlace: 42656E6A616D696E, B62011FF7CF6BD3AFB044E079445251B>
String: <Exception on go to StringDump for: 42656E6A616D696E, B62011FF7CF6BD3AFB044E079445251B
        OccupancyDenied>
StringEnginePlace: <[T]Telename: <47656E6572616C204D6167696320496E63, B38080EA74D6DFAA8829792A638B07C8>(3 occup
        TwoStringBearer: <Telename: <42656E6A616D696E, B62011FF7CF6BD3AFB044E079445251B>>
        StringStart: <Telename: <54484520535452494E475354415254, E872548DBA7D21DE573B3FF802F75129>(1 occupant)>
                TwoStringBearer: <Telename: <436F726579, 9FE2B5D6E1DA4F0E0DC3C25300B4900A>>
        StringDump: <Telename: <5448452053545249 4E4744554D50, D16D04CEFC1AC0415392CCA98F3DBF05>(1 occupant)>
                StringDumpster: <Telename: <54484520535452494E4744554D5053544552, 24F2401DF8AE68D0D8091C1FE2D65
# scr (3) Mon 17:26:16: Unable to deliver occupant: OccupancyDenied
String: <Agent entering the StringEnginePlace: 436F726579, 9FE2B5D6E1DA4F0E0DC3C25300B4900A>
String: <Exception on go to StringDump for: 436F726579, 9FE2B5D6E1DA4F0E0DC3C25300B4900A
        OccupancyDenied>
StringEnginePlace: <[T]Telename: <47656E6572616C204D6167696320496E63, B38080EA74D6DFAA8829792A638B07C8>(3 occup
        StringStart: <Telename: <54484520535452494E475354415254, E872548DBA7D21DE573B3FF802F75129>(0 occupants)
        StringDump: <Telename: <5448452053545249 4E4744554D50, D16D04CEFC1AC0415392CCA98F3DBF05>(1 occupant)>
                StringDumpster: <Telename: <54484520535452494E4744554D5053544552, 24F2401DF8AE68D0D8091C1FE2D65
        TwoStringBearer: <Telename: <436F726579, 9FE2B5D6E1DA4F0E0DC3C25300B4900A>>
String: <
------------------------------
The parting strings:
        List: <5 elements>
        When you're a Jet
        You're a Jet all the way
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <
------------------------------
The parting strings:
        List: <5 elements>
        When you're a Jet
        You're a Jet all the way
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <
The parting strings:
        List: <5 elements>
        When you're a Jet
        You're a Jet all the way
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <
------------------------------
```

```
The parting strings:
        List: <5 elements>
        When you're a Jet
        You're a Jet all the way
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <
------------------------------
The parting strings:
        List: <5 elements>
        When you're a Jet
        You're a Jet all the way
        From your first cigarette
        To your last dying day
        ------ West Side Story
------------------------------>
String: <Agent exited the Dump: 416C66726564, AD2BAB1D6E91F44DA02C49C7EB70875A>
String: <Agent exited the Dump: 4261726E6579, 28E242107E93699CC57D3335EA0B8ACD>
String: <Agent exited the Dump: 43616C76696E, 5A4EB14C9EEAA75C93D1F8FF27DD8DC0>
String: <Agent exited the Dump: 446F6E616C64, 5C1333557A20A45F05B781DB3159A95E>
String: <Agent exited the Dump: 456762657274, FB7293D4B607301E285B5DA0966140D3>


*/


// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```

```
/*
  Copyright 1993, General Magic.  All Rights Reserved.

  DESCRIPTION

  nameDirectory:  A DirectoryPlace object maintains a list of ASCII keys for Telename values.
    These key-value pairs are added to the directory by UpdateDirAgent objects on behalf of
    DestinationPlace objects.  TravelAgent objects can then leave from OriginPlace objects,
    carrying an ASCII string name for a DestinationPlace, and travel to the DirectoryPlace
    object.  There, the TravelAgent objects can exchange the string for the DestinationPlace
    telename, create a ticket based on that telename, and go to the DestinationPlace object.


*/


*@Place.publicPackages.include(DirectoryModule: module = (  // module definition for DirectoryModule


  StrVal: class (Object) = (

//  For performance you might consider using dump2Chars() directly
//  for those types which are already formatted nicely by the engine.


      public
          value:  String;


          initialize: op (object:  protected Object) = {
              theFinding : List[Integer,Equal] | Nil;
              startPos, theLength : Integer;

              ^;

              if object is Number {
                  value = object@Number.asString();
              }
              else if object is String {
                  value = object@String.copy();
              }
              else if object is Identifier {
                  value = object@Identifier.asString();
              }
              else if object is ClassName {
                  value = object@ClassName.classDigest.asString();
              }
              else if object is Class {
                  value = object@Class.name.classDigest.asString();
              }
              else if object is Time {
                  value = object.dump2Chars();
              }
              else  {
                  value = object.dump2Chars();

                  // For anything but primitives return now.

                  if !(object is Primitive) && !(object is OctetString!)
                          && !(object is Identifier) {
                      return;
                  };
                  theFinding = Pattern("<").find(value,nil);
                  if theFinding != nil {
                      startPos = theFinding[1] + 1;
                      theLength = value.length;
                      if startPos < theLength {
                          value = value.substring(startPos,theLength);
                      };
                  };
              };
          };
  );


  Pair:  class[classOne,classTwo:  Class] (Object) = (
    public
        object1:  classOne;
        object2:  classTwo;


        initialize: op(arg1: classOne;  arg2:  classTwo) = {
            ^;
            object1 = arg1;
            object2 = arg2;
        };
  );
```

```
ShippingBox: sealed interface (Object, Unmoved) = (
     public
       destinations: readonly protected List[Ticket, Equal];

       exception: readonly TripException|Nil;

       origin: readonly protected Way;

       time: readonly Time;

       peek: op(identifier: Identifier) copied Object
       throws FeatureUnavailable;

       poke: op(identifier: Identifier; attribute: copied Object)
       throws ArgumentInvalid, FeatureUnavailable;

       redirect: op(exception: TripException|Nil; newDestination: copied Ticket|Nil);
     public
       initialize: op()
       throws FeatureUnavailable;
   );


Traveled: mixin interface (Place) = (
     private
       transferOut: op(
         box: unprotected ShippingBox;
         nextHop: protected Way|Nil;
         deliverBefore: Integer|Nil;
                 travelPackages: List[Package, Equal]|Nil);

        private
                travelPackages: List[Package, Equal];


     public
       transferredIn: sponsored op(
         box: unprotected ShippingBox;
         previousHop: protected Way|Nil;
         outboundTransfer: Boolean);
   );


  DirectoryEnginePlace: class (EnginePlace,Traveled) = (  // class definition for the DirectoryEnginePlace clas
    // A DirectoryEnginePlace object creates a DirectoryPlace
    // object, several DestinationPlace objects, and an
    // OriginPlace object.  Each of these places will create
    // the agents that will exist within them or travel among
    // them.

    public
      initialize: op(bootPlacePkgs:List[Package[Identifier,Class,Equal], Equal];
        Telename; Teleaddress; Integer|Nil; Integer|Nil) = {

        for pkgIndex: Integer to bootPlacePkgs.length {
                    *.publicPackages.include(bootPlacePkgs[pkgIndex]);
        };
        ^;        // escalate AFTER including the packages into the publicPackages of
                // the Engine Place

      };

      transferredIn: sponsored op(box: unprotected ShippingBox; hop: protected Way|Nil; transfer: Boolean) = {

        if (box.exception != nil)
        {
          box.redirect(box.exception,Ticket(*.name));   // make the agent go to here
        };
          *.transferOut(box, nil, nil,nil);
      };



      live: sponsored op(Exception|Nil) = {
        // create the place for the directory service
        DirectoryPlace(Permit(5000), nil, nil, nil);

        // create several destinations to be included in the directory
        DestinationPlace("Kauai", Permit(5000), nil, nil);
        // wait for agent originating from DestinationPlaces to register with directory
        self.wait(4);

        DestinationPlace("Maui", Permit(5000), nil, nil, nil);
        // wait to register with directory
        self.wait(4);

        DestinationPlace("Hawaii", Permit(5000), nil, nil, nil);
```

```
        // wait to register with directory
        self.wait(4);

        DestinationPlace("Maui", Permit(5000), nil, nil, nil);
        // wait to register with directory -- note this is a duplicate entry
        self.wait(4);

        // create a place from which agents go to the directory and then the destinations

        OriginPlace(Permit(40000), nil, nil, nil);

        // wait while agents originating from OriginPlace go to destinations via directory
        loop { } ;
      };
);  // end of DirectoryEnginePlace class definition

DirectoryPlace: class (Place,EventProcess) = {  // class definition for the DirectoryPlace class
                    // A DirectoryPlace object maintains a list of
                    // key-value pairs that match an ASCII string name
                    // to a Telescript Telename for a particular
                    // DestinationPlace object
  public
    initialize: op(Permit; Permit|Nil; Integer|Nil; OctetString|Nil; String|Nil) = {

      // key-value pairs are kept in a Telescript Dictionary object
      // The keys are constrained to be String objects
      yellowPages = Dictionary[String,Telename,Equal]();
      ^;
      *.enableEvents(ExitEvent(nil,nil));   // register interest in these events
    };

    live: sponsored op(Exception|Nil) = {
      loop {

        *.reportExit(*.getEvent(nil,nil)@ExitEvent.record);

      };
    };

    entering: sponsored op(subjectName:protected Telename; subjectClass: protected ClassName!; p: unprotected
      if ((subjectClass == TravelAgent.name)
        ) {
        String("A TravelAgent-type object entered the directory: ",
          subjectName.dump2Chars()).dump();
      } else {
        header: String;

        header = String("An agent entered the directory: ",
                subjectName.dump2Chars(),
                "\nThe names in the directory at entry time:\n\t");
        self.printDirectory(header);
      };
      return Pair[Telename,ClassName](subjectName.copy(), subjectClass.copy());
    };


  private

    reportExit: op(theInfo: Object) = {

      subjectClass: ClassName = theInfo@Pair[Telename,ClassName].object2;
      subjectName: Telename = theInfo@Pair[Telename,ClassName].object1;

      if ((subjectClass == TravelAgent.name)
        ) {
        String("A TravelAgent-type object exited the directory: ",
          subjectName.dump2Chars()).dump();
      } else {
        header: String;

        header = String("An agent exited the directory: ",
                subjectName.dump2Chars(),
                "\nThe names in the directory at exit time:\n\t");
        self.printDirectory(header);
      };

    };


    // include another key-value pair in the yellowPages

    /*  We need the data passed into this method to be owned by
      the place. This is because the data is added to the dictionary
      maintained by the place (the directory). Agents bringing data to
      be added to the directory (in this example) own the data they
      are passing into this method so we need to either transfer ownership
      of that data to the place (by using 'owned' as we do here) or we need
      to make a copy of the objects for adding to the dictionary (in that
```

```
      case we would use 'copied').

      In our case, the agent invoking 'addListing' doesn't need to leave with
      the data so it doesn't need its own copy.
    */

  public
    addListing: op(key: owned String; name: owned Telename) = {
      // Add the specified key/name pair to the dictionary. There's one
      // problem with this routine -- you could get duplicate keys in the
      // dictionary (which the lab notes suggest you fix)
      yellowPages.add(key, name);
    };

    // return the Telename from the yellowPages that matches the given key
    // Here, we need to return a copy (owned by the requestor) of the Telename
    // since it is going to be used in a Ticket for 'go'.

    returnValue: op(key: String) copied Telename! = {
      return((yellowPages.get(key))@Telename);
    };

    // print out all key-value pairs in the directory
    printDirectory: op(str: String) = {
      String("\n------------------------------\n",
          str,
          StrVal(yellowPages).value,
          "\n------------------------------").dump();
    };

  private
    yellowPages: Dictionary[String,Telename,Equal];
);                          // end of class DirectoryPlace

DestinationPlace: class (Place) = { // class definition for the DestinationPlace class
      // Each DestinationPlace object creates an
      // UpdateDirAgent object that carries an ASCII
      // name string along with the Telename of the
      // DestinationPlace object to the DirectoryPlace
      // object.  The name string-Telename key-value
      // pair will be added to the directory there

      // These need to be owned by the DestinationPlace since keeps them in
      // its attributes and if this place goes away, so does that data.
  public
    initialize: op(str: owned String; Permit; Permit|Nil; Integer|Nil; OctetString|Nil; String|Nil) = {
      nameKey = str;    // ASCII name string for this place
      ^;
    };

    live: sponsored op(Exception|Nil) = {
      // create an UpdateDirAgent object to put the Telename of this place in the directory
      /*  By looking at the class definition of UpdateDirAgent we can note that
        that ownership of 'nameKey' is passed to the UpdateDirAgent in order to avoid
        since UpdateDirAgent needs it. We don't need to make a copy since we have no further
        need for 'nameKey'. On the other hand, self.name is passed by copy since the place
        cannot transfer ownership of its name.
      */
      UpdateDirAgent(nameKey, self.name, Permit(500), nil, nil, nil);

      // loop forever, waiting for incoming TravelAgent objects
      loop {  };
    };

    entering: sponsored op(subjectName: protected Telename; subjectClass: protected ClassName!; p: unprotecte
      if (subjectClass == TravelAgent.name) {
        String("TravelAgent entered the destination: ", subjectName.dump2Chars()).dump();
      };
      return nil;
    };

  private
    nameKey: String;
);          // end of class DestinationPlace

OriginPlace: class (Place) = {  // class definition for the OriginPlace class
               // An OriginPlace object craete a TravelAgent
               // object to go to a DestinationPlace object
  public
    initialize: op(Permit; Permit|Nil; Integer|Nil; OctetString|Nil; String|Nil) = {
      ^;
    };
    live: sponsored op(Exception|Nil) = {
      a_list: List[String,Equal] = List[String,Equal]("Maui", "Hawaii", "Kauai", "Maui", "Hawaii");

      // create a TravelAgent object that will go to the DestinationPlace whose
      // whose telename corresponds to the ASCII string name (the key in the DirectoryPlace
      // object's yellowPages directory), that is specified in the initialization arguments
```

```
        // here

        TravelAgent("Kauai", Permit(1000), nil, nil, nil);

        loop {  };
      };
  );                          // end of class OriginPlace

  UpdateDirAgent: class (Agent) = {   // class definition for the UpdateDirAgent class
          // An UpdateDirAgent object goes to the DirectoryPlace
          // object to add to its yellowPages the key-value pair
          // for the DestinationPlace object that created it
    public
      initialize: op(key: owned String; tname: copied Telename; Permit; Permit|Nil; Integer|Nil; String|Nil) =
        nameKey = key;
        nameValue = tname;
        ^;
      };
      live: sponsored op(Exception|Nil) = {
        try {
          // go to the DirectoryPlace object
          self.go(Ticket(nil, nil, ClassName("DirectoryPlace".asOctetString()), nil));
          // add the key-value pair to the directory
          here@DirectoryPlace.addListing(nameKey, nameValue);
        } catch error: Exception {
          "error:  UpdateDirAgent going to directory".dump();
          StrVal(error.class);
        };
      };
    private
      nameKey: String;
      nameValue: Telename;
  );                          // end of class UpdateDirAgent

  TravelAgent: class (Agent) = {      // class definition of the TravelAgent class
        // A TravelAgent object goes to the DirectoryPlace
        // object armed with a string.  It gets the
        // Telename object that matches that string from the
        // directory and then goes to THAT place.  Cool!
    public
      // we have the string passed in as copied so that TravelAgent gets its own
      // copy it can take with it.
      initialize: op(key: copied String; Permit; Permit|Nil; Integer|Nil; String|Nil) = {
        nameKey = key;
        ^;
      };

      live: sponsored op(Exception|Nil) = {
        placeName: Telename;

        try {
          // go to the DirectoryPlace object
          self.go(Ticket(nil, nil, ClassName("DirectoryPlace".asOctetString()), nil));

          /*  Next, get the Telename object that matches this nameKey string.
            Note that  the returned value is copied (see returnValue method
            definition) so that we can make our ticket and go with this Telename.
          */
          placeName = here@DirectoryPlace.returnValue(nameKey);

          try {
            // go to the DestinationPlace that has the telename returned by the directory
            self.go(Ticket(placeName, nil, nil, nil));
          } catch error: Exception {
            "error:  TravelAgent going to destination".dump();
            StrVal(error.class);
          };
        } catch error: Exception {
          "error:  TravelAgent going to directory".dump();
        };
      };
    private
      nameKey: String;
  );                          // end of class TravelAgent

)   // end of module DirectoryModule
);      // end of module include into publicPackages of BootPlace

do {
  << [ [:Teleaddress]  [:Telename] #self publicPackages :DirectoryEnginePlace] discard >>;
  }


/*
** Result:
**
```

```
String: <
------------------------------
An agent entered the directory: 47656E6572616C204D6167696320496E63, 6D5D52987DB9FF7B1DFB48A1651B4ADA
The names in the directory at entry time:
        Dictionary: <0 elements>
------------------------------>
String: <
------------------------------
An agent exited the directory: 47656E6572616C204D6167696320496E63, 6D5D52987DB9FF7B1DFB48A1651B4ADA
The names in the directory at exit time:
        Dictionary: <1 element>
        Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <
------------------------------
An agent entered the directory: 47656E6572616C204D6167696320496E63, 3FEFD81C373A1272EE43A8E5BCC432B2
The names in the directory at entry time:
        Dictionary: <1 element>
        Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <
------------------------------
An agent exited the directory: 47656E6572616C204D6167696320496E63, 3FEFD81C373A1272EE43A8E5BCC432B2
The names in the directory at exit time:
        Dictionary: <2 elements>
        Maui 47656E6572616C204D6167696320496E63, 37396A69EF31A5C43BB69FD99503CA03
      Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <
------------------------------
An agent entered the directory: 47656E6572616C204D6167696320496E63, C8F2EC48E666068B844748943CCFE2A3
The names in the directory at entry time:
        Dictionary: <2 elements>
        Maui 47656E6572616C204D6167696320496E63, 37396A69EF31A5C43BB69FD99503CA03
      Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <
------------------------------
An agent exited the directory: 47656E6572616C204D6167696320496E63, C8F2EC48E666068B844748943CCFE2A3
The names in the directory at exit time:
        Dictionary: <3 elements>
        Maui 47656E6572616C204D6167696320496E63, 37396A69EF31A5C43BB69FD99503CA03
      Hawaii 47656E6572616C204D6167696320496E63, 6AACB113C501F6E1EB81EC81DECD57E4
        Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <
------------------------------
An agent entered the directory: 47656E6572616C204D6167696320496E63, 3537C46E933C267785A4263CB9A4633C
The names in the directory at entry time:
        Dictionary: <3 elements>
        Maui 47656E6572616C204D6167696320496E63, 37396A69EF31A5C43BB69FD99503CA03
      Hawaii 47656E6572616C204D6167696320496E63, 6AACB113C501F6E1EB81EC81DECD57E4
        Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <
------------------------------
An agent exited the directory: 47656E6572616C204D6167696320496E63, 3537C46E933C267785A4263CB9A4633C
The names in the directory at exit time:
        Dictionary: <3 elements>
        Maui 47656E6572616C204D6167696320496E63, 37396A69EF31A5C43BB69FD99503CA03
      Hawaii 47656E6572616C204D6167696320496E63, 6AACB113C501F6E1EB81EC81DECD57E4
        Kauai 47656E6572616C204D6167696320496E63, 9FDD630A878AC841DDC8AE600B73B174
------------------------------>
String: <A TravelAgent-type object entered the directory: 47656E6572616C204D6167696320496E63, 6CBD12ED277C90CB8
String: <TravelAgent entered the destination: 47656E6572616C204D6167696320496E63, 6CBD12ED277C90CB84C010A3BE7DI
String: <I went all the way to Kauai
        and all I got was this stinkin' string:
        The Garden Island>
String: <A TravelAgent-type object entered the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D674
String: <I went all the way to Maui
        and all I got was this stinkin' string:
        A Suburb of Los Angeles>
String: <A TravelAgent-type object entered the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D674
String: <I went all the way to Hawaii
        and all I got was this stinkin' string:
        The Big Island>
String: <A TravelAgent-type object entered the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D674
String: <I went all the way to Kauai
        and all I got was this stinkin' string:
        The Garden Island>
String: <A TravelAgent-type object entered the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D674
String: <A TravelAgent-type object exited the directory: 47656E6572616C204D6167696320496E63, 6CBD12ED277C90CB84
String: <A TravelAgent-type object exited the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D6743
String: <A TravelAgent-type object exited the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D6743
String: <A TravelAgent-type object exited the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D6743
String: <I went all the way to Maui
        and all I got was this stinkin' string:
```

```
        A Suburb of Los Angeles>
String: <A TravelAgent-type object entered the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D674
String: <I went all the way to Hawaii
        and all I got was this stinkin' string:
        The Big Island>
String: <A TravelAgent-type object exited the directory: 47656E6572616C204D6167696320496E63, B585E4364CDF8D6743
String: <A TravelAgent-type object exited the directory: 47656E6572616C204D6167696320496E63,' B585E4364CDF8D6743



*/


// Local Variables:
// mode:text
// tab-width:4
// tab-stop-list:(4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80)
// End:
```