How to Create a Programming Language

L4S: DOMAIN SPECIFIC LANGUAGES

1

TOPICS

- WHAT ARE DOMAIN SPECIFIC LANGAUGES?
- WHY DOMAIN SPECIFIC LANGUAGES
- INTERNAL VS EXTERNAL DSLS
- INTERNAL DSLS
  - ADVANTAGES
  - DRAWBACKS
  - DEVELOPING AN INTERNAL DSL
  - EXAMPLES OF INTERNAL DSLS
- EXTERNAL DSLS
  - ADVANTAGES
  - DRAWBACKS
  - DEVELOPING AN EXTERNAL DSL
  - EXAMPLES OF EXTERNAL DSLS

2

WHAT ARE DOMAIN SPECIFIC LANGAUGES?

- LIMITED EXPRESSIVENESS – GENERALLY, A SMALL LANGUAGE
- SIMPLER DATA ENTRY
- FOCUSES ON A PARTICULAR DOMAIN (OR PROBLEM)
- EASY TO USE [EVEN FOR NON PROGRAMMER SUBJECT MATTER EXPERTS]
  - TERMS ARE CLOSER TO SME VOCABULARY
- TYPE SAFE (LIMITED SYNTAX/SEMANTICS HELPS AVOID MISUSING TYPES, ETC)
- TYPICALLY ARE 'DECLARATIVE'

3

WHY DOMAIN SPECIFIC LANGUAGES?

- SMALL – CAN'T BUILD A WHOLE SYSTEM IN IT
- SPECIFIC OPERATIONS TO SOLVE SPECIFIC DOMAIN PROBLEMS
- EASY FOR 'NON-PROGRAMMERS' TO UNDERSTAND
- ENABLES COMMUNICATION

4

INTERNAL VS EXTERNAL

- INTERNAL (OR EMBEDDED) DSLS
  - PART OF OR EXTENSION TO AN EXISTING GENERAL PURPOSE LANGUAGE
  - MAY BE A 'LIBRARY' CALLED FROM THE GPL
- EXTERNAL DSLS
  - SEPARATE LANGUAGE/SEPARATE COMPILER OR INTERPRETER
  - DESIGNED TO STAND APART FROM GPLS

5

INTERNAL DSL ADVANTAGES

- RELATIVELY EASY TO IMPLEMENT
- CAN LEVERAGE WELL SUPPORTED IDES [ECLIPSE, INTELLIJ, ETC]
- HOST LANGUAGE CAN HELP REINFORCE 'CORRECTNESS'

6

DRAWBACKS OF INTERNAL DSLS

- STUCK WITH A LOT OF 'SYNTAX' FROM THE HOST LANGUAGE
- LESS FLEXIBLE THAN EXTERNAL DSLS

7

DEVELOPING AN INTERNAL DSL

- SOME TECHNIQUES THAT HELP WITH EMBEDDING A DSL
  - 'FLUENT' INTERFACE
  - METHOD CHAINING
    - RETURNING 'THIS'
    - RETURNING INTERMEDIATE OBJECTS
  - STATIC FACTORY METHODS AND IMPORTS

8

INTERNAL DSL EXAMPLES

- SIMPLE VACATION PLANNING SYSTEM
- 2 VERSIONS
  - ONE USES METHOD CHAINING [METHODS RETURN 'SELF']
  - ONE USES 'INTERMEDIATE OBJECTS'

9

VACATION

- VERSION 1:

```
class Vacation:
    def __init__(self):
        self.startDate = "70/01/01"
        self.endDate = "70/01.07"
        self.bookedCity = ""
        self.bookedHotel = ""
        self.bookedAirline = ""
        self.bookedFlight = ""

    def start(self, newDate):
        self.startDate = newDate
        return self

    def end(self, newDate):
        self.endDate = newDate
        return self

    def city(self, newCity):
        self.bookedCity = newCity
        return self

    def hotel(self, newHotel):
        self.bookedHotel = newHotel
        return self

    def airline(self, newAirline):
        self.bookedAirline = newAirline
        return self

    def flight(self, newFlight):
        self.bookedFlight = newFlight
        return self

    def toString(self):
        return "From: " + self.startDate + ", Until: " +
self.endDate + ", City: " + self.bookedCity + ",
Hotel: " + self.bookedHotel + ", Flying on: " +
self.bookedAirline + " Flight#: " +
self.bookedFlight
```


10

VACATION [USAGE]

```
import vacation

myVaca = vacation.Vacation().start("08/04/20").end("08/07/20").city("Sejong City").hotel("Sejong
Academy of Sciences and Arts").airline("TheBus Airlines").flight("1234")

alsoMyVaca = vacation.Vacation().end("08/07/20").start("08/04/20").airline("TheBus
Airlines").flight("2345").city("Sejong City").hotel("Sejong Academy of Sciences and Arts")

print("Vacation info:")
print(myVaca.toString())
print("vaca 2!")
print(alsoMyVaca.toString())
```

```
testVacation.py
Vacation info:
From: 08/04/20, Until: 08/07/20, City: Sejong City, Hotel: Sejong Academy of Sciences and Arts,
Flying on: TheBus Airlines Flight#: 1234
vaca 2!
From: 08/04/20, Until: 08/07/20, City: Sejong City, Hotel: Sejong Academy of Sciences and Arts,
Flying on: TheBus Airlines Flight#: 2345
```


11

VACATION 2 [INTERMEDIATE OBJECTS]

```
class Vacation:
    def __init__(self):
        self.startDate = "70/01/01"
        self.endDate = "70/01.07"
        self.location = ""
        self.transport = ""

    def start(self, newDate):
        self.startDate = newDate
        return self

    def end(self, newDate):
        self.endDate = newDate
        return self

    def destination(self, theLocation):
        self.location = theLocation

    def transportation(self, theTransport):
        self.transport = theTransport

    def toString(self):
        return "From: " + self.startDate + ", Until: " + self.endDate + "\n   Location: " +
self.location.toString() + "\n   Transportation: " + self.transport.toString()
```


12

VACATION 2 [INTERMEDIATE OBJECTS]

```
class Destination():
    def __init__(self):
        self.bookedCity = ""
        self.bookedHotel = ""

    def city(self, newCity):
        self.bookedCity = newCity
        return self

    def hotel(self, newHotel):
        self.bookedHotel = newHotel
        return self

    def toString(self):
        return " City: " + self.bookedCity + ", Hotel: " + self.bookedHotel
```


13

VACATION 2 [INTERMEDIATE OBJECTS]

```
class Transportation():
    def __init__(self):
        self.bookedAirline = ""
        self.bookedFlight = ""

    def airline(self, newAirline):
        self.bookedAirline = newAirline
        return self

    def flight(self, newFlight):
        self.bookedFlight = newFlight
        return self

    def toString(self):
        return " Flying on: " + self.bookedAirline + " Flight#: " + self.bookedFlight
```


14

VACATION 2 USAGE

```
import vacation

myVaca = vacation.Vacation().start("08/04/20").end("08/07/20")

myLocation = vacation.Destination().city("Sejong City").hotel("Sejong Academy of Sciences and Arts")

myTransportation = vacation.Transportation().airline("TheBus Airlines").flight("1234")

myVaca.destination(myLocation)
myVaca.transportation(myTransportation)

print("Vacation info:")

print(myVaca.toString())
```

```
Vacation info:
From: 08/04/20, Until: 08/07/20
  Location:  City: Sejong City, Hotel: Sejong Academy of Sciences and Arts
  Transportation:  Flying on: TheBus Airlines Flight#: 1234
```


15

EXTERNAL DSL ADVANTAGES

- NOT TIED TO A GPL AND ITS SYNTAX/STRUCTURES
  - GREAT FREEDOM IN DESIGNING AN INTUITIVE SYNTACTIC AND SEMANTIC STRUCTURE
- CAN EASILY BE USED FOR 'COMMUNICATION' BETWEEN DEVELOEPRS ARE USERS
  - INTERNAL DSLS ARE HARDER TO SEGREGATE FROM COMPLEX GPL SYNTAX

16

EXTERNAL DSL DRAWBACKS

- COST OF LEARNING A NEW LANGUAGE
  - MINIMIZED BY DESIGNING LANGUAGE FOR EASY ACQUISITION BY NON-PROGRAMMERS
- NEED TO WRITE COMPLETELY DIFFERENT COMPILER/TRANSLATION SYSTEM
- DIFFICULT TO BALANCE TRADEOFFS BETWEEN 'SPECIFICITY' AND NEED FOR GENERAL LANGUAGE CONSTRUCTS
- LESS EFFICIENT THAN CUSTOM CODE IN GPL
- DIFFICULTY IN INTEGRATING AN 'EXTERNAL' DSL WITH OTHER PARTS OF SYSTEM

17

DEVELOPING AN EXTERNAL DSL

- SPEAK TO SUBJECT MATTER EXPERTS (SMES) ON THE DOMAIN
- DECIDE ON ESSENTIAL OPERATIONS
- CREATE A SYNTAX THAT IS
  - EASY FOR NON-TECHNICAL SMES TO WORK WITH
    - MAY TAKE DISCUSSION WITH SME
  - IS PARSEABLE BY AN LL(1) OR LR(1) PARSER
    - LOOK AT LEFT RECURSION IN THE GRAMMAR
    - LOOK AT COMMON PREFIXES, ETC
- DECIDE IF LANGUAGE WILL BE COMPILED OR INTERPRETED
- BUILD COMPILER/INTERPRETER
- RELEASE TO SMES AND GET FEEDBACK ON USABILITY

18

EXAMPLE EXTERNAL DSLS

- **AWK** – FOR TEXT PROCESSING/EDITING
- **DOT** – FOR DRAWING GRAPHS
- **GHERKIN** – FOR SPECIFYING SYSTEM TESTS
- **PAINTUML** – FOR CREATING UML DIAGRAMS
- **SQL** – DATABASE QUERRIES
- **HTML** – WEB LAYOUT
- **CSS** – WEB PAGE STYLE DESCRIPTION
- **XML** – DATA ENCODING
- **VHDL** – HARDWARE DESIGN
- **ANTLR** – LEXER AND PARSE DESCRIPTION [ALSO, PLY, LEX, YACC, FLEX, BISON, ETC]
- **MAKE** – SPECIFYING BUILD SYSTEMS/PROCEDURES

19

SQL – STRUCTURED QUERY LANGUAGE

- LANGUAGE FOR QUERYING DATABASES
- ENGLISH-LIKE SO EASY TO UNDERSTAND
- RELATIVELY EASY TO SPECIFY DESIRED DATA AND RELATIONSHIPS

```
SELECT partNumber, quantityInStock, price FROM Inventory
    WHERE price > 50.00 AND quantityInStock > 10
    ORDER by price;

-- Reads records from the inventory table. Reports on parts where there
are more than 10 in stock and the prices is greater than $50.00. It orders
the results ascending by the value of price.
```


20

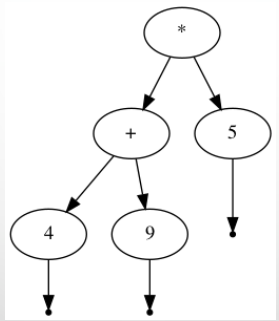
EXTERNAL DSL EXAMPLES

Dot:

```
digraph AST {
"*" -> "+";
"*" -> 5;
"+" -> 4;
"+" -> 9;
null1 [shape=point];
5 -> null1;
null2 [shape=point];
4 -> null2;
null3 [shape=point];
9 -> null3;
}
```


21

EXTERNAL DSL EXAMPLES



22

QUESTIONS?

23