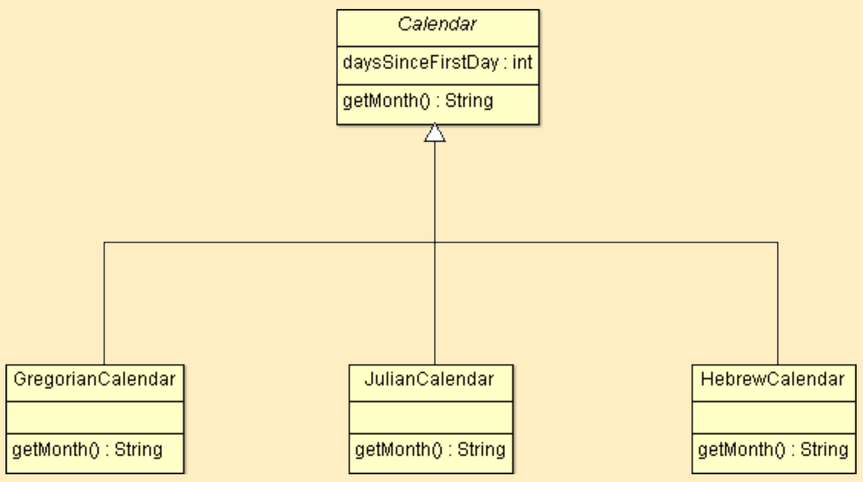
SWNE
Uma Linguagem Baseada em Predicados

Alex Abate Biral
Orientação: Prof. Marco Dimas Gubitoso

Despacho Dinâmico

O conceito de despacho dinâmico simples foi introduzido pela linguagem Simula 67. Ele consiste em associar funções ou procedimentos a classe de um dos argumentos da mesma. Diferentes implementações desta função podem ser criadas, cada uma associada a uma sub-classe diferente do objeto. Quando o procedimento é chamado, o programa determina qual versão deste usar através da classe do objeto que possui o método.
Por exemplo, suponha que queremos implementar uma classe "Calendar" para representar dias do ano. As datas são identificados de maneira inequívoca através de um número inteiro representando quantos dias se passaram dês de 1 de Janeiro de 1970 (podendo ser negativo). As subclasses de Calendar implementam todas os mesmos métodos, como getMonth, que retorna o mês da data específica no calendário da classe.

Figura: Sub-classes com despacho dinâmico para o método "getMonth".



Usando classes como as descritas na figura acima, o código para a implementação dos métodos getMonth se torna muito mais fácil de ser lido, pois não precisa se preocupar com três casos diferentes.
O resultado disto é que linguagens com despacho dinâmico permitem que o código seja escrito de maneira bem mais modular e simples. Isto permite tanto que o desenho do programa seja mais claro e direto e que o código seja entendido mais facilmente.

Outros Despachos

Porém, o despacho dinâmico mais comum é o despacho simples que acabamos de explicar. Porém, este despacho é bem limitado. Suponha que no nosso exemplo de calendários, queiramos implementar um método de igualdade "equals".
A linguagem que trabalhamos já provem um método equals, então queremos sobrecarregar este método para retornar false caso o objeto comparado não seja um calendário. E caso seja, comparamos o campo daysSinceFirstDay.
Com o despacho simples, o sistema acima nem sempre funciona. Se o programa não souber quando está compilando que o argumento de equals é um calendário, o método mais genérico, com argumento Object será usado! O despacho simples só pode ocorrer sobre a classe do dono do método, não os outros argumentos.
Para resolver situações como esta, uma variedade de outros tipos de despacho foram criados, como o despacho múltiplo, que resolve exatamente este problema, permitindo que as classes de todos os argumentos sejam levadas em conta. Outro tipo de despacho é o despacho por padrão, que usa algum tipo de padrão, como expressões regulares, para classificar os argumentos da função. Também existe o despacho por classes predicativas, onde a classe dos argumentos não é pré-definida, mas calculada de acordo com algum critério das mesmas.

Despacho por Predicados

Generalizando essas várias ideias, podemos pensar em um despacho por predicados. Isto é, a especialização de métodos poderia ser feita simplesmente em cima de qualquer teste sobre um ou mais argumentos. Suponhamos que queremos testar a Conjectura de Collatz. Então criamos a função *collatz* onde:

$$\begin{aligned} \mathit{Collatz}(X) &= \frac{X}{2}\ (2|X) \\ &= 3X+1\ (2\not|X) \\ &= 1\ (X=1) \end{aligned}$$

O Despacho por predicados permite implementar funções como este de maneira mais similar a como pensamos sobre ela.

SWNE

O SWNE é uma linguagem que implementa o conceito de despacho por predicados. Diferente das implementações atuais, o SWNE não implementa outro conceito, como classes, para representar seus objetos. Um predicado pode servir como classe no sentido que ele provém estrutura a um objeto e é exclusivo a outros predicados, mas um objeto pode não ter "classe" nenhuma.
O objetivo do SWNE é servir como prova de conceito de que é possível ter uma linguagem focada apenas em predicados, e no futuro funcionar como arcabouço para o estudo de que possibilidades o uso de predicados dentro do desenho da linguagem pode oferecer.

Objetos

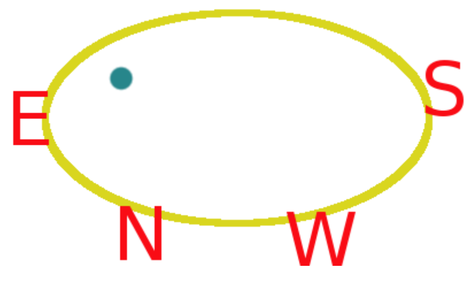
Todos os elementos substantivos do programa, todos os valores, são chamados de objetos. Cada objeto pode ter vários "campos", que são objetos associados ao primeiro através de uma variável. Um objeto também pode ter um valor intrínseco. No momento isto só é possível para objetos nativos da linguagem, mas o trabalho desenvolvido mostra como seria possível que o programador criasse novos objetos com valores próprios.

Predicados

O coração do SWNE são os predicados, que provém o sistema básico de despacho. No SWNE, dividimos os predicados em predicados dinâmicos e estáticos.
Os predicados dinâmicos são basicamente uma função simples, sobre um número qualquer de objetos, que retorna um valor Booleano. Para evitar problemas neste estágio, as funções predicativas não possuem nenhum tipo de despacho. O nome dinâmico é usado porque um objeto pode entrar e sair do predicado sem que nenhuma referência seja feita ao mesmo. Como uma conta no banco pode se tornar devedora apenas com uma operação de transferência.
Os predicados estáticos são basicamente conjuntos aos quais objetos podem ser adicionados ou removidos de forma explícita. Por exemplo, num browser, um URL ser ou não favorito depende apenas se ele está na lista de favoritos. Por causa que o predicado é referenciado explicitamente, apenas predicados estáticos podem adicionar estrutura a um objeto (desta forma, eles funcionam como classes).
Os predicados são ordenados entre si através de relações de implicação. Por exemplo, o predicado (potencia de dois) implica no predicado (par). Isto é importante para evitar que um predicado mais genérico sobrescreva um mais específico.

Conclusões & Trabalhos Futuros

O SWNE, apesar de ser uma linguagem bastante limitada, mostra que é possível implementar o despacho por predicados como mecânica central de uma linguagem. Além disso, o trabalho desenvolvido mostra várias possibilidades para o futuro.
O uso de predicados, por exemplo, permitiria que a linguagem possuísse características de linguagens lógicas, como Prolog. O despacho dinâmico poderia ser usado também para tratamento de erros. Usando um sistema de currying, seria possível especializar funções de maneira que os argumentos possam ser passados iterativamente. Com a base do SWNE pronta, há uma vasta gama de possibilidades a serem exploradas.



Para saber mais, acesse:

`https://linux.ime.usp.br/~zaratrus/swne/`