

# Diccionarios 2

## Diccionario implementado con listas enlazadas

Cátedra 26 y 27

Otoño 2005

Una lista enlazada es la estructura de datos dinámica más simple. Dinámica quiere decir que puede cambiar su tamaño durante el transcurso del programa, es decir, puede agrandarse o achicarse, a diferencia de los arreglos, que tienen el gran problema de tener un tamaño fijo. Hoy implementaremos un diccionario mediante una lista enlazada.

```
Nodo n=new Nodo("Perro","Dog",null);
Nodo p=new Nodo("Gato","Cat",n);
Nodo q=new Nodo("Caballo","Horse",p);
```

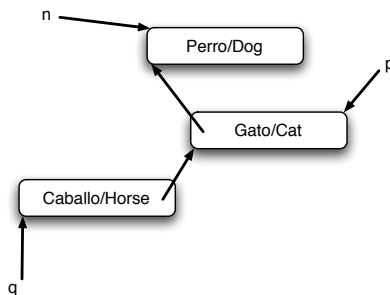
## 1 Nodo

Un nodo es un objeto donde se guarda la definición de un elemento del diccionario y una **referencia** al un nodo que le sigue:

```
class Nodo {
    String llave,valor;
    Nodo prox;
    Nodo(String l, String v, Nodo n) {
        llave=l;
        valor=v;
        prox=n;
    }
}
```

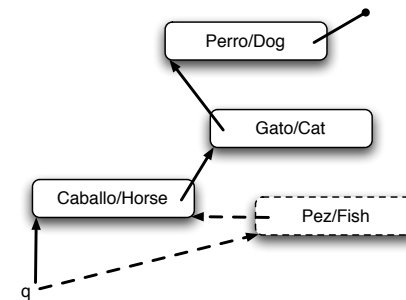
## 2 Listas

Una lista es un conjunto de nodos enlazados. Por ejemplo, para crear la lista de la figura, el código es



## 2.1 Inserción de un nodo

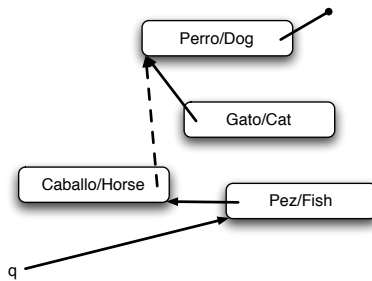
La inserción de un nodo al **inicio** de la lista anterior, se puede hacer en forma sencilla



```
Nodo s=new Nodo("Pez","Fish",q);
q=s;
```

## 2.2 Eliminación de un nodo

Eliminar el nodo n-ésimo de la lista es un poco más complejo. Hay que recorrer n-1 nodos, y hacer que el nodo anterior al n-ésimo apunte al que le sigue:



Es decir,

```

if (n==0)
    q=q.prox;
else {
    Nodo p=q;
    int i=0;
    while (i<n-1)
        p=p.prox;
    p.prox=p.prox.prox;
}
  
```

Nótese que en caso de que n valga 0, se modifica q directamente.

## 4 Un diccionario con listas enlazadas

La siguiente clase Diccionario usa una lista enlazada para permitir mantener un diccionario que crezca:

```

class Diccionario {
    Nodo definicion;
    Diccionario() {
        definicion=null;
    }
    void put(String l, String v) {
        Nodo n=new Nodo(l,v,definicion);
        definicion=n;
    }
    String get(String l) {
        Nodo n=definicion;
        while (n!=null) {
            if (n.llave.equals(l))
                return n.valor;
            n=n.prox;
        }
        return null;
    }
}
  
```

```

void remove(String l) {
    if (definicion==null)
        return;
    if (definicion.llave.equals(l)) {
        definicion=definicion.prox;
        return;
    }
    Nodo q=definicion;
    while (!q.prox.equals(l))
        q=q.prox;
    q.prox=q.prox.prox;
}
  
```