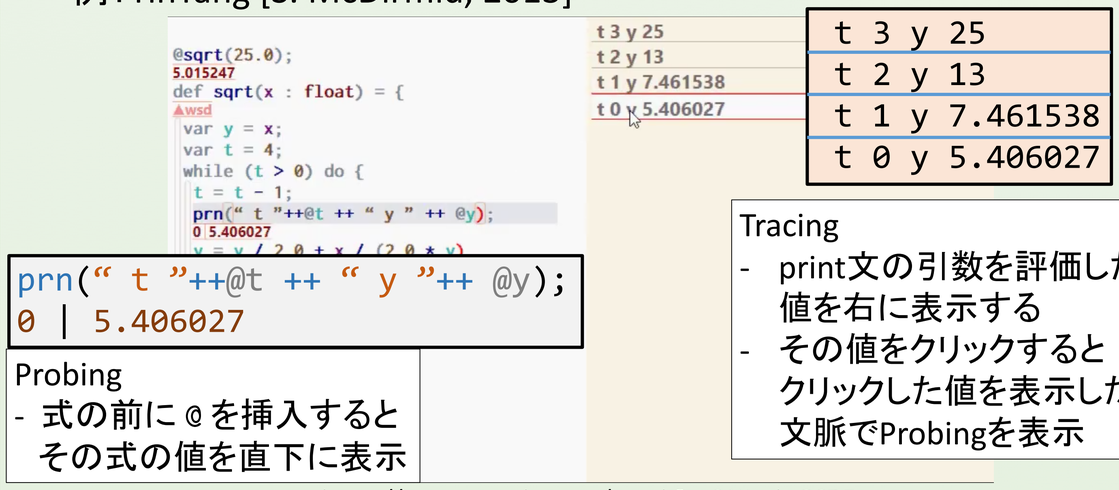
Kanonライブプログラミングを用いたデータ構造のプログラミング

岡 明央　増原 英彦　今井 朝貴　青谷 知幸（東京工業大学）

背景：ライブプログラミング(LP)環境

- **ライブプログラミング(LP)環境**は、プログラムを編集するたびにプログラムの実行結果や実行時情報を**即座に**表示・反映する
- プログラムと出力間の相互関係を教えてくれる機能も備えている
- 例：YinYang [S. McDirmid, 2013]



https://www.youtube.com/watch?v=01Xyoh-G6DE

問題：既存のLP環境はデータ構造を文字列でしか出力しない

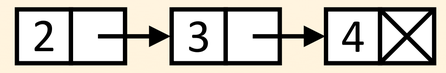
- 例：連結リストを文字列で出力した場合

既存のLP環境で出力される文字列

```
Node(val=2,
     next=Node(val=3,
               next=Node(val=4,next=Nil)))
```

想像しなければならない！

プログラマが想像する構造

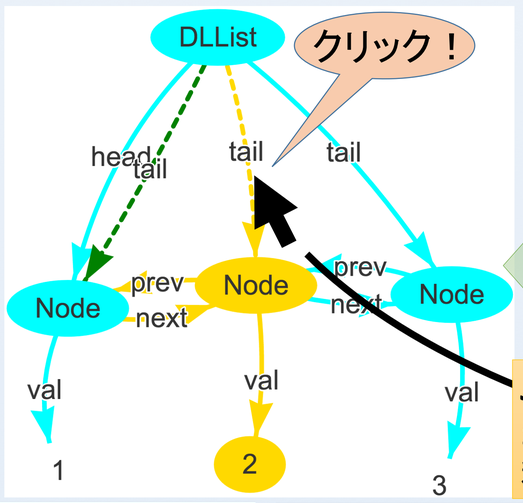


提案：データ構造を可視化するLP環境Kanon

課題
1. データ構造をどのように見せるのか
   - 見せ方にも色々ある
2. プログラムと図との対応がわかりにくい

データ構造はプログラム実行時に変化する！

Kanonの機能



Summarized View Mode
最後まで実行した時点のデータ構造を表示し、変化を1枚にまとめて表示

カーソルがlist.add(2)のとき

Jump to Construction
生成された場所にカーソルが移動し、文脈が移動

矢印の色
オレンジ：「追加」
緑　　：「削除」

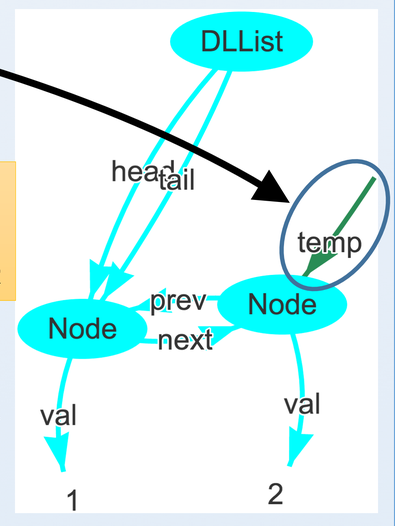
破線矢印
→ 最後まで実行した時点で存在しなかった参照関係

```
1  function DLList () {...};
2  function Node(val) {...};
3  DLList.prototype.add = function(val) {
4      var $temp = new Node(val);
5
6      if (this.head === null) {
7          this.head = temp;
8          this.tail = temp;
9      } else {
10         temp.prev = this.tail;
11         this.tail.next = temp;
12         this.tail = temp;
13     }
14 };
15 var list = new DLList();
16 list.add(1); list.add(2); list.add(3);
```

Probe
変数宣言時に$を挿入した変数をグラフ上に表示

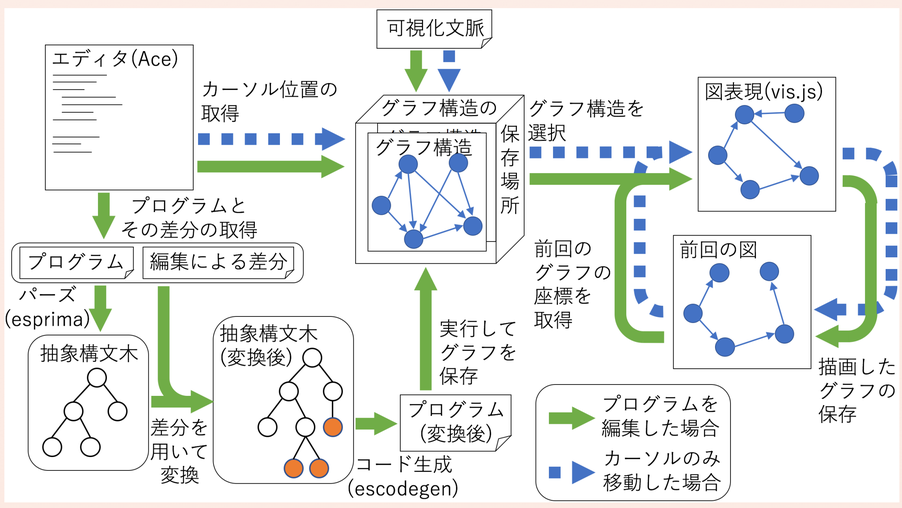
Snapshot View Mode
プログラムをカーソル位置まで実行した時点のデータ構造を表示

list.add(2)の文脈で表示



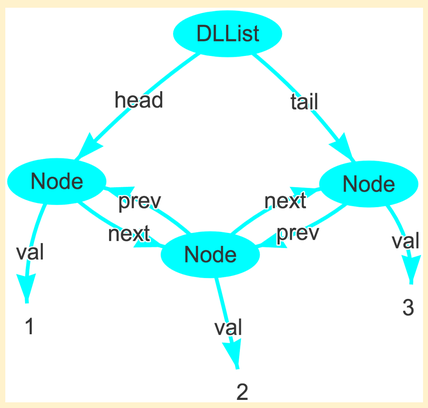
Kanonの実現

- 対象言語はJavaScript
- Khan Academy の Live Editor を参考に作成
- https://github.com/prg-titech/Kanon で公開
  - HTML, CSS, JavaScript の合計約2000行で実装

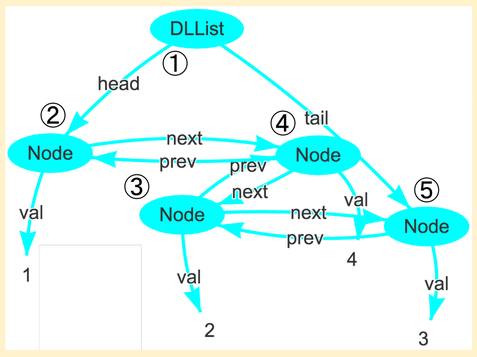


プログラムを変更してもグラフの概形を保つ描画方法

プログラムを実行して得られるオブジェクトは毎回異なる！
→ オブジェクトにIDを与えて識別

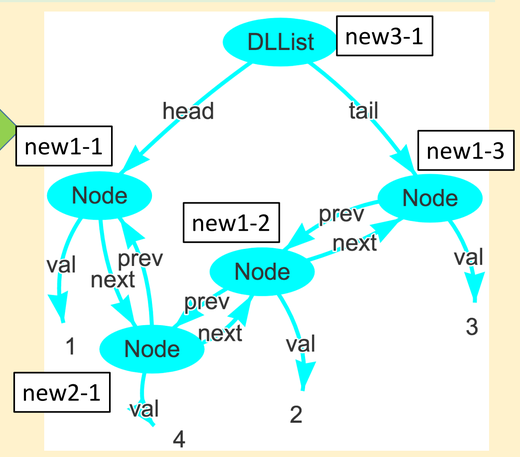



```
add = function(val) {
  var temp = new Node(val);
  ...                          new1
};
insert = function(val,i) {
  var temp = new Node(val);
  ...                          new2
};
var list = new DLList();
list.add(1);                   new3
list.add(2);
list.add(3);
list.insert(4, 1);
```



オブジェクトの生成順にIDを与える場合
→ 期待するオブジェクトがずれて描画される！

オブジェクトの生成場所と実行回数からIDを与える場合
→ オブジェクトを正しく識別できる！



オブジェクトのIDと、座標の組を保存
→ 再描画する際に再利用！

編集してもnew式のIDを維持する方法

new式のIDを編集前と編集後で同一のものを識別したい

```
add = function(val) {
  var t = new Node(val);
  ...                          new1
};

var list = new DLList();       new2
```

差分を用いてIDを計算

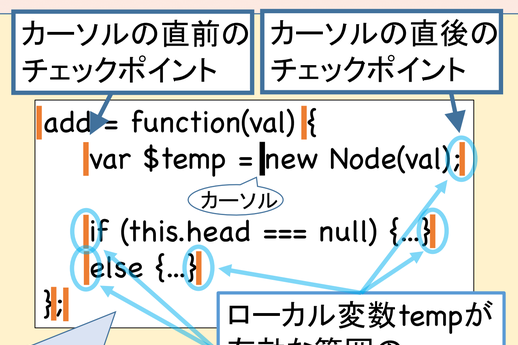
```
add = function(val) {
  var t = new Node(val);
  ...                          new1
};
insert = function(val,i) {
  var t = new Node(val);
  ...                          new3
};

var list = new DLList();       new2
```

各機能の実現

Snapshot View Mode
- カーソルの直前のCPで保存したグラフを描画

Summarized View Mode
- カーソルの前後のCPで差分を計算



Probe
有効な範囲のチェックポイントの引数に「変数名」と「評価した値」を渡す

```
{temp: eval(temp)}
```

Jump to Construction
実行時、各要素が出現した場所を保存し、クリックした要素に対応する場所に移動

関連研究

既存のLP環境(データ構造に非対応)
e.g.) YinYang [S. McDirmid, 2013]
Swift Playground [2014]
Khan Academy's Live Editor [2012]

グラフの概形を保つ方法
- 要素を比較して識別 [T. Zimmermann and A. Zeller, 2002]
- 利点も欠点も存在する

Python Tutor [Philip J. Guo, 2013]
- 最もLP環境に近いデータ構造可視化環境
- 欠点：フィードバックに1秒前後に時間がかかる
  グラフが見にくい

変化する絵の見せ方
- LightTable [2012]：ストロボ写真
- DejaVu [J. Kato, S. McDirmid, X. Cao, 2012]：タイムライン表示

今後の課題

- 多数のオブジェクトを生成するプログラムへの対応
  - グラフの頂点数が多くなる
    - 解決1：フィードバック速度の向上
    - 解決2：グラフの一部だけを表示
- 値の変化を出力に表示
  - ループ変数や、文脈に応じて変化する変数