オブジェクト指向哲学
～入門編～

第5回　連結リストを使ったプログラム

～配列以外のデータ構造に挑戦！～

学習目標

- Java における参照の仕組みを説明できる
  - 基本データ型と参照型の違いを説明できる
- 連結リストを使ったプログラムを書ける
  - 連結リストを使う利点を説明できる
  - 連結リストの概念を説明できる
  - 連結リストへの挿入、削除、検索のプログラムを Java を使って書ける

5.1. 参照の仕組み

今回の本題である連結リストを理解するためには、参照の仕組みを理解しておく必要があります。まず、次の問題に挑戦してみましょう。

＜考えよう！＞次の2つのリストがどのような出力をするか考えなさい。

参照練習問題1

```
public static void main(String args[]){
  int x;
  int y;
  x = 3;
  y = x;
  y = 15;
  System.out.println(x);
  System.out.println(y);
}
```

参照練習問題2

```
public static void main(String args[]){
  TestClass x;
  TestClass y;
  x = new TestClass();
  x.id = 3;
  y = x;
  y.id = 15;
  System.out.println(x.id);
  System.out.println(y.id);
}

class TestClass {
  int id;
}
```

5.1.1.　Java における参照の仕組み

①.メモリと番地

プログラムは、変数をメモリに格納します。

- メモリには、すべての場所に番地がついています。
- この番地のことを指すことを参照といいます。

メモリ

番地	内容
100	
101	
102	
103	
104	

②.参照とメモリの仕組み

変数がどのようにしてメモリに格納されるかトレースしてみましょう。

(1)参照練習問題１の場合

参照練習問題1

```
public static void main(String args[]){
  int x;
  int y;
  x = 3;
  y = x;
  y = 15;
  System.out.println(x);
  System.out.println(y);
}
```

メモリ

番地	内容
100	
101	
102	
103	
104	

メモリ

番地	内容
100	
101	
102	
103	
104	

メモリ

番地	内容
100	
101	
102	
103	
104	

メモリ

番地	内容
100	
101	
102	
103	
104	

メモリ

番地	内容
100	
101	
102	
103	
104	

(2)**参照練習問題２の場合**

参照練習問題2

```
public static void main(String args[]){
  TestClass x;
  TestClass y;
  x = new TestClass();
  x.id = 3;
  y = x;
  y.id = 15;
  System.out.println(x.id);
  System.out.println(y.id);
}

class TestClass {
  int id;
}
```

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

メモリ

番地	内容
100	
101	

オブジェクト用のメモリ

1000	
1001	

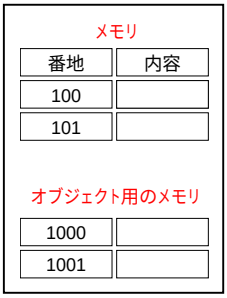
③.基本データ型と参照型

- 基本データ型
  値が直接入るタイプ

メモリ

番地	内容
100	
101	
102	
103	
104	

- 参照型
  オブジェクト用のメモリに生成され、参照が入るタイプ



(1)基本データ型を覚えましょう

基本データ型か参照型か見分けるために、基本データ型を覚えましょう。
Java の基本データ型 8 種類しかありません。

整数が入るもの	int
	long
	short
実数(浮動小数点数)が入るもの	float
	double
真偽値が入るもの	boolean
文字が入るもの	byte（１バイト文字）
	char（２バイト文字）

(2)その他はすべて参照型

new 演算子でインスタンス化しなければならないものは参照型です。

実は配列も参照型です。

```
int[] idArray = new int[5];
```

メモリ

	番地	内容
noArray	101	1000(番地)

オブジェクト用のメモリ

	番地	内容
[0]	1000	0
[1]	1001	0
[2]	1002	0
[3]	1003	0
[4]	1004	0

5.2.　連結リスト

5.2.1.　　配列にこだわる必要はない

　これまでは、配列を使ってデータを扱ってきました。配列は使いやすいのですが、問題点もあります。どんな問題点があるでしょうか。考えてみましょう。

①.間に割り込んで挿入するのが大変

テキストエディタ（メモ帳、mule...）を作ることを考えましょう。

＜機能＞

・文字を挿入する

・文字を削除する

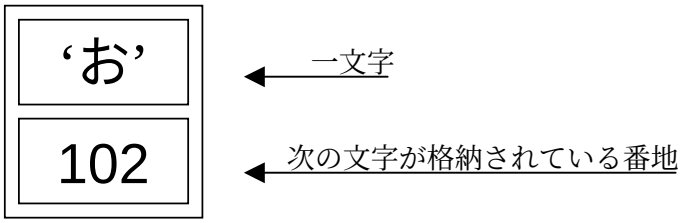
②.配列で作ったとすると...

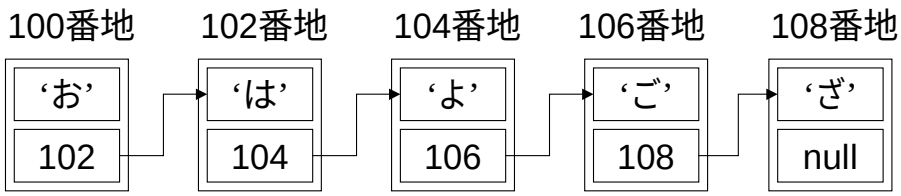
	番地	内容
[0]	1000	‘お’
[1]	1001	‘は’
[2]	1002	‘よ’
[3]	1003	‘う’
[4]	1004	‘ざ’
[5]	1005	‘い’
[6]	1006	‘ま’
[7]	1007	‘す’

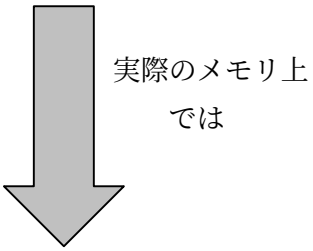
③.配列での実装の問題点

5.2.2.　連結リストとは？

メモリに「一文字」と「次の文字が格納されている番地」を記憶しておく方法です。





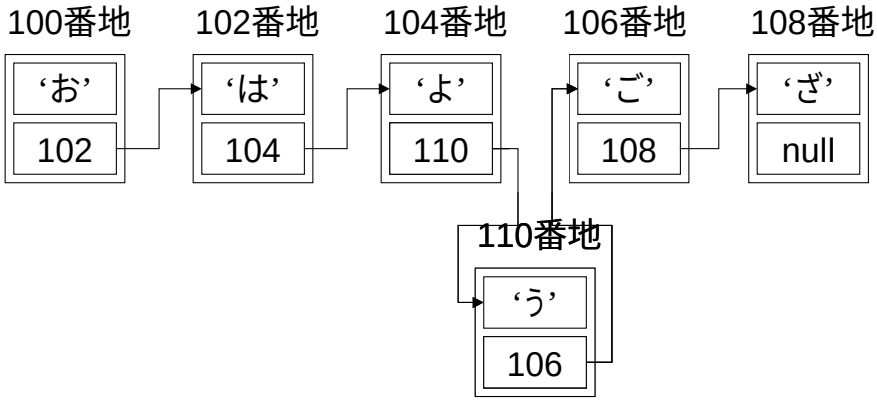


番地	内容
100	‘お’
101	102
102	‘は’
103	104
104	‘よ’
105	106
106	‘う’
107	

①.連結リストの特徴

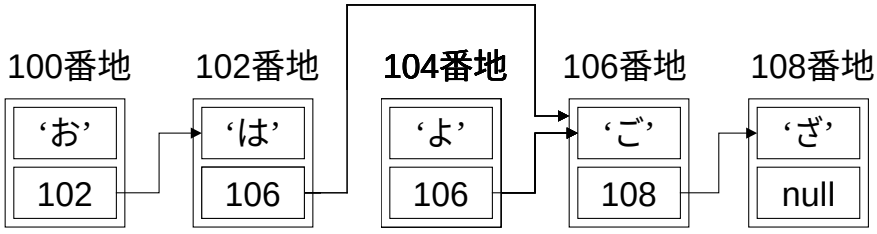
(1)連結リストへの挿入

- 移動しなくても済みます！



(2)連結リストからの削除

- 挿入よりもっと簡単に行えます！



(3)連結リストの利点・欠点

考えてみましょう。

5.3.　連結リストを実装する

商品管理のプログラムを、ItemTypeList クラスと Example クラスに連結リストを使って、作り直してみましょう。

- ItemType クラスは変更しなくてすむはずです

5.3.1.　連結リストの実装

①.連結リストの実装例

例題 5-1：連結リストの実装(Example5_1.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 5-1：連結リストの実装
4:      * 取り扱う商品種類を管理するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example5_1 {
9:
10:       /**
11:       * メイン
12:       * 取り扱う商品種類を管理するプログラム
13:       * コーラ、ソーダ、お茶を追加し、リストを表示する
14:       */
15:       public static void main(String[] args) {
16:
17:         //自動販売機プログラムの開始を知らせる
18:         System.out.println("自動販売機が開始しました。");
19:
20:         //商品種類連結リストを生成する
21:         ItemTypeList itemTypeList = new ItemTypeList();
22:
23:         //商品種類を保存するための連結始点終点を生成する
24:         LinkTerminal linkTerminal = new LinkTerminal();
25:
26:         //商品種類を追加する
27:         itemTypeList.add(linkTerminal,new ItemType(1001,"コーラ",120));
28:         itemTypeList.add(linkTerminal,new ItemType(1002,"ソーダ",120));
29:         itemTypeList.add(linkTerminal,new ItemType(1003,"お茶",120));
30:
31:         //商品種類リストを表示する
32:         itemTypeList.display(linkTerminal);
33:
34:       }
35:     }
```

例題 5-1：連結リストの実装(ItemTypeList.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 5-1：連結リストの実装
4:      * 取り扱う商品種類を管理するプログラム
5:      *
6:      * 商品種類リストクラス
7:      */
8:      public class ItemTypeList {
9:
10:       /**
11:       * 商品種類を追加する
12:       */
13:       public void add(LinkTerminal linkTerminal,ItemType addItemType){
14:         //追加する連結インスタンスを生成する
15:         LinkObject addLink = new LinkObject();
16:         addLink.data = addItemType;
17:
18:         if(linkTerminal.first == null){//連結リストが空のとき
19:           linkTerminal.first = addLink;
20:           linkTerminal.last = addLink;
21:         }else{//連結リストが空でないとき
22:           linkTerminal.last.next = addLink;
23:           linkTerminal.last = addLink;
24:         }
25:       }
26:
27:       /**
28:       * 商品種類リストを表示する
29:       */
30:       public void display(LinkTerminal linkTerminal){
31:         LinkObject current = linkTerminal.first;//今たどっている連結インスタンス
32:         while(current != null){
33:
System.out.println(current.data.id+":"+current.data.name+":"+current.data.price+" は 販 売
中です");
34:           current = current.next;
35:         }
36:       }
37:
38:     }
```

例題 5-1：連結リストの実装(LinkObject.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 5-1：連結リストの実装
4:      * 取り扱う商品種類を管理するプログラム
5:      *
6:      * 連結インスタンスクラス
7:      */
8:      public class LinkObject {
9:        ItemType data;     //商品種類
10:       LinkObject next;   //次の要素への参照
11:     }
```

例題 5-1：連結リストの実装(LinkTerminal.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 5-1：連結リストの実装
4:      * 取り扱う商品種類を管理するプログラム
5:      *
6:      * 連結リストの始点と終点をもつクラス
7:      */
8:      public class LinkTerminal {
9:        LinkObject first;//始点
10:       LinkObject last;//終点
11:     }
```

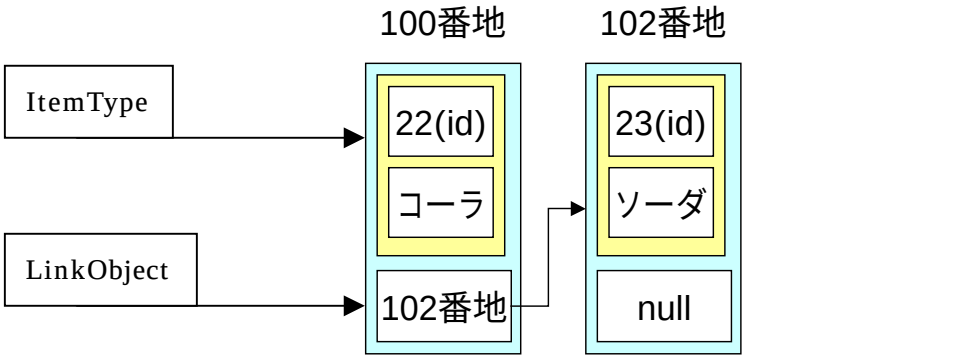
②.クラス設計

(1)LinkObject **クラス**

- 連結クラス LinkObject を作ります

```
//連結クラス
class LinkObject {
  ItemType data;     //商品種類
  LinkObject next;   //次の要素への参照
}
```

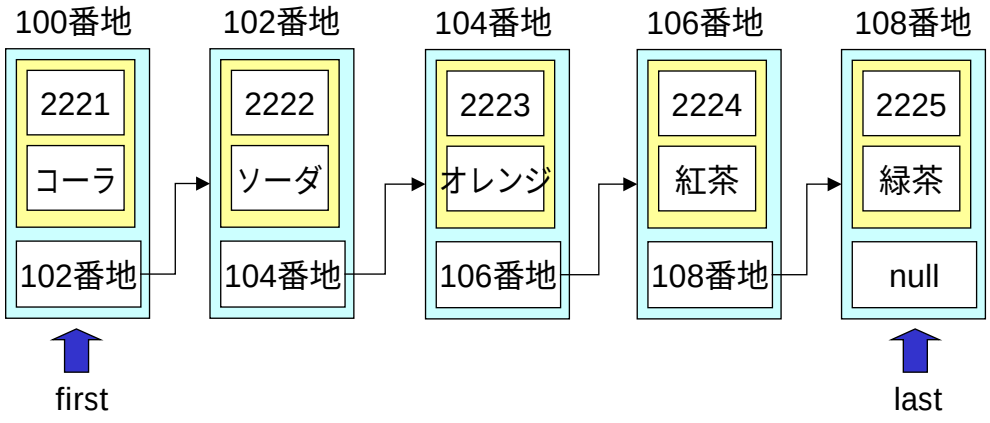
リストは自分で自分を持っているように見えますが､実は参照なので以下のようになります。



（実際には ItemType も参照です）

- 連結リストを使った ItemTypeList クラスの実装方針

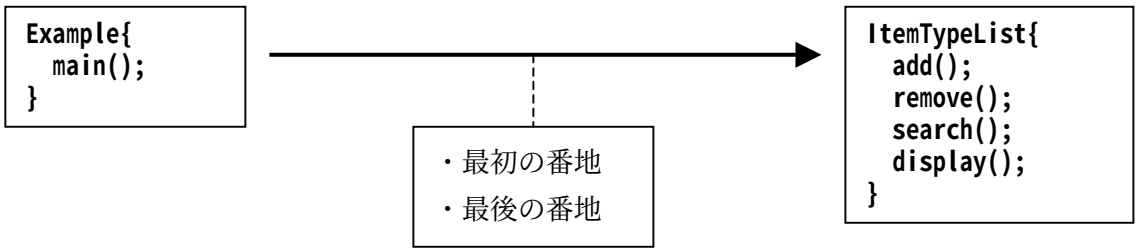
挿入や検索をするために、メインクラスは、常にリストの最初の番地(first)と最後の番地(last)だけは、知っておく必要があります。



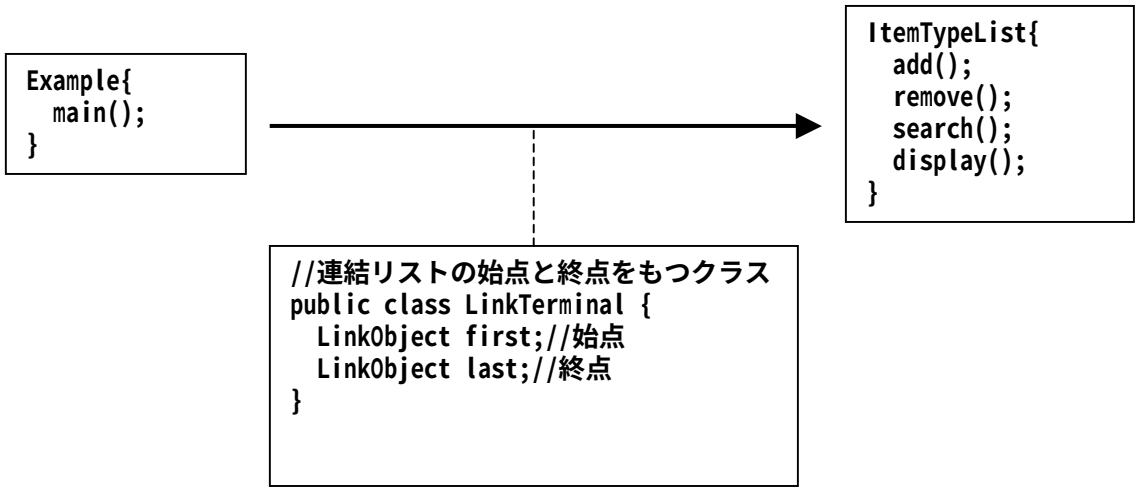
(2)LinkTerminal **クラス**

前回同様、Example クラスと ItemTypeList クラスを分離します

引数として、配列ではなく、最初の番地 (first)と最後の番地(last)を渡す必要があります。



そこで引数用クラス LinkTerminal を作り、それを渡すことにします。



5.3.2.　連結リストのアルゴリズム

①.追加アルゴリズム

(1)一般的な追加の場合

（追加メソッドから抜粋）

```
LinkObject regLink = new LinkObject();
regLink.data = regItem;

last.next = regLink;
last = regLink;
```

(2)一番最初の追加をする場合

（追加メソッドから抜粋）

```
LinkObject regLink = new LinkObject();
regLink.data = regItem;

if(first == null){
  first = regLink;
  last = regLink;
}else{
  last.next = regLink;
  last = regLink;
}
```

②.検索アルゴリズム

(1)探したい商品種類が見つかる場合

（検索メソッドから抜粋）

```
LinkObject current=first;
while(current!=null){
  if(current.data.id==searchId){
    System.out.println(“見つかりました”) ;
    return current.data;
  }
  current=current.next;
}
System.out.println(“見つかりませんでした”);
return null;
```

(2)探したい商品種類が見つからない場合

（検索メソッドから抜粋）

```
LinkObject current=first;
while(current!=null){
  if(current.data.id==searchId){
    System.out.println(“見つかりました”) ;
    return current.data;
  }
  current=current.next;
}
System.out.println(“見つかりませんでした”);
return null;
```

③.削除アルゴリズム

(1)削除したい要素が、リストの中央で見つかる場合

（削除メソッドから抜粋）

```
LinkObject current=first;
LinkObject previous;
while(current!=null){
  if(current.data.id==removeId){
    previous.next=current.next;
    return current.data;
  }
  previous=current;
  current=current.next;
}
```

(2)削除したい要素が、リストの最初で見つかる場合

（削除メソッド中から抜粋）

```
LinkObject current=first;
LinkObject previous;
while(current!=null){
  if(current.data.id==removeId){
    if(corrent==first){
      first=current.next;
      return current.data;
    }
}
```

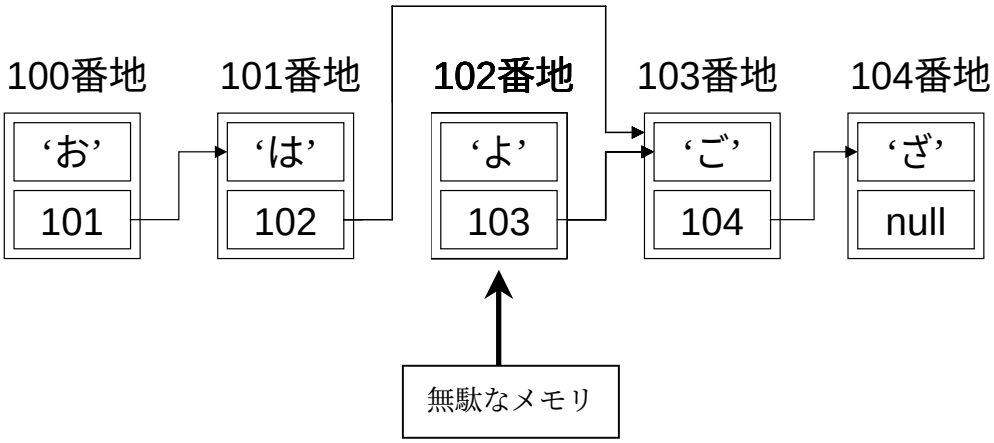
(3)削除したい要素が、リストの最後で見つかる場合

（削除メソッド中から抜粋）

```
LinkObject current=first;
LinkObject previous;
while(current!=null){
  if(current.data.id==removeId){
    if(corrent==last){
      last=previous;
      last.next=null;
      return current.data;
    }
    previous=current;
    current=current.next;
  }
}
```

Java Tips　ガベージ･コレクション

連結リストによるデータ構造で､削除するときに無駄なメモリが残ってしまう気がするのですが、心配いりません。Java では、いらないメモリを常に見張っている「ガベージコレクタ」というすばらしい機能が標準でついています。誰からも参照されていないインスタンスを自動的にメモリから消します。



練習問題

＜記述問題＞

☆記述問題 5-1

連結リストの利点･欠点を配列と比べながら説明せよ。

☆記述問題 5-2

何故 Java において 2 種類の参照の仕組みがあるのか考えよ。

＜プログラム問題＞

☆プログラム問題 5-1

プログラム問題 4-2 で作ったプログラムを仕様は変えずに連結リストで作り変えよ。なお、このプログラムは以下の 5 つのクラスから構成される。

```
public class Exercise5_1{
  main()
}
```

Exercise5_1.java

```
public class ItemType{
  int id
  String name
  int price
```

ItemType.java

```
public class ItemTypeList{
  add();
  remove();
  search();
  display();
}
```

ItemTypeList.java

```
public class LinkTerminal{
  LinkObject first
  LinkObject last
}
```

LinkTerminal.java

```
public class LinkObject{
  ItemType data
  LinkObject next
}
```

LinkObject.java