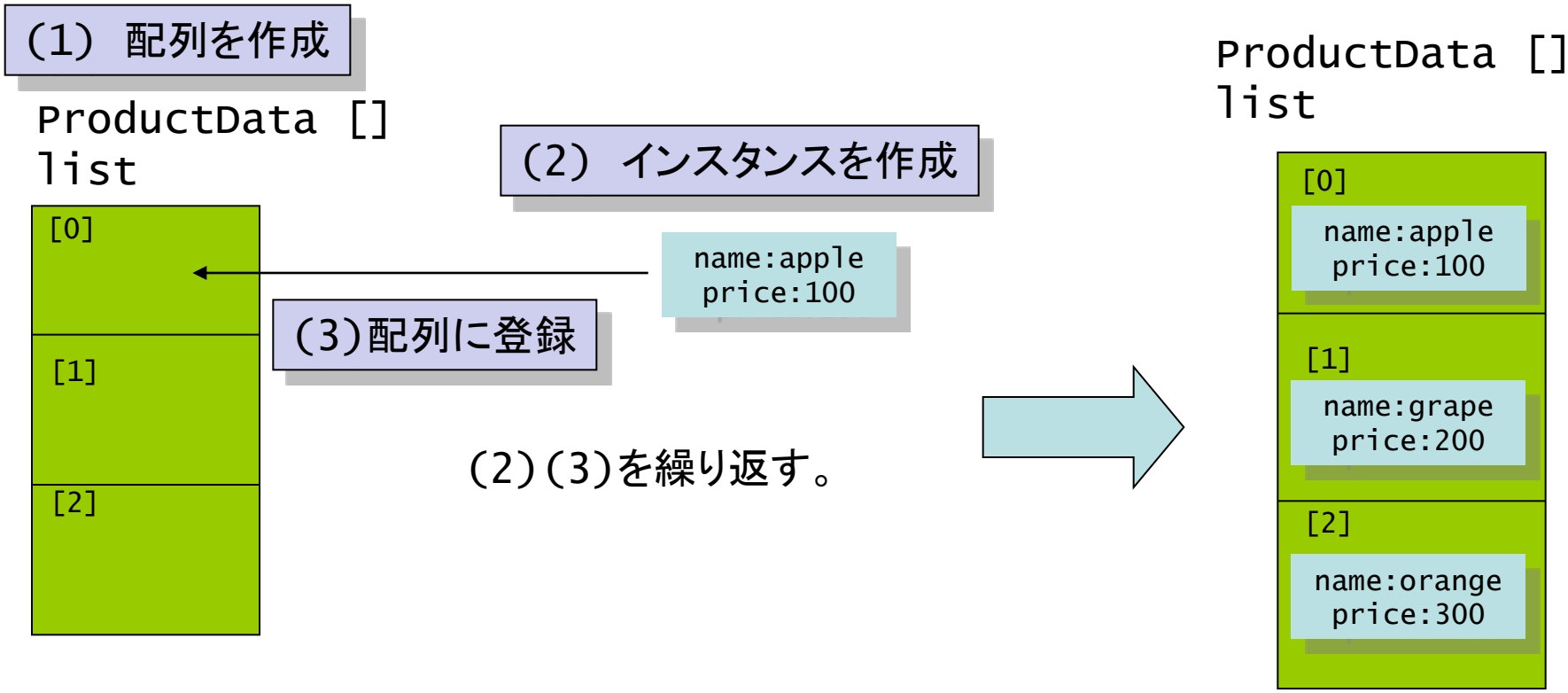
Java 2

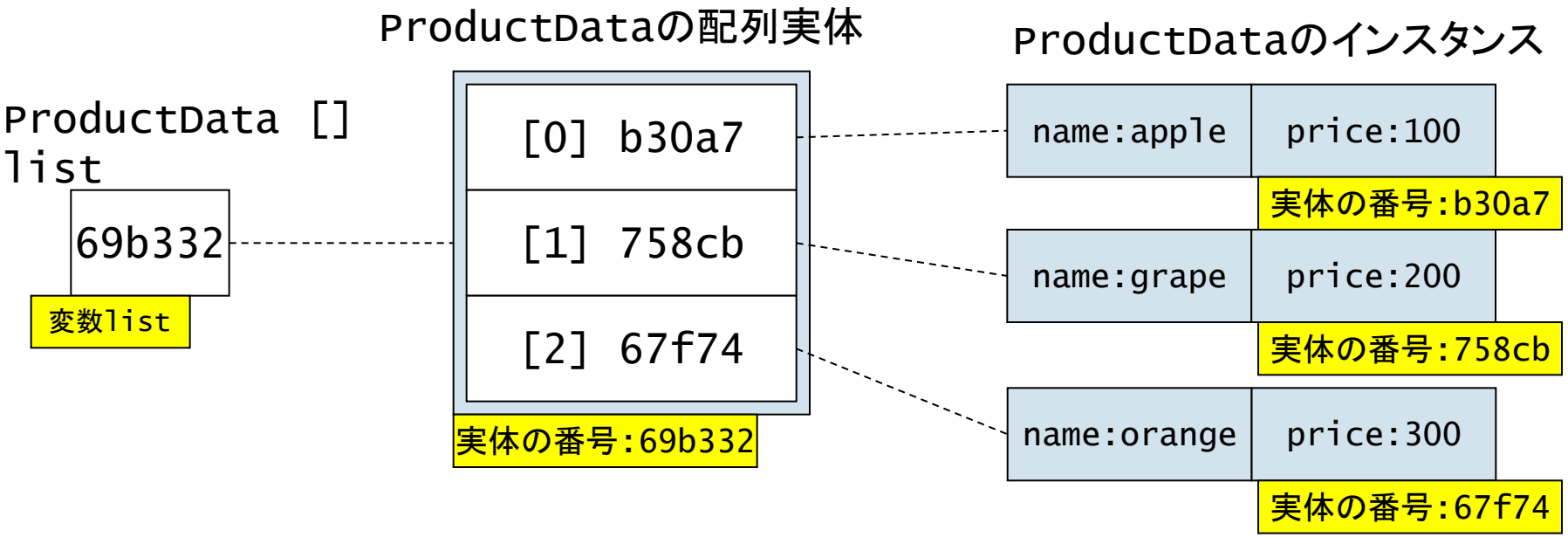
第9回　表形式データ(2)

［復習］ 配列と複合データを用いた表形式データの作成
表へのレコードの登録（考え方）




```
int numProducts = 3;
ProductData [] list = new ProductData[numProducts];
list[0] = new ProductData("apple", 100);
list[1] = new ProductData("grape", 200);
list[2] = new ProductData("orange", 300);
```

［復習］ 配列と複合データを用いた表形式データ（詳細）



実際には、配列変数listには配列実体への参照が格納される。
また、配列実体にはインスタンスの参照が格納される。

```
int numProducts = 3;
ProductData [] list = new ProductData[numProducts];
list[0] = new ProductData("apple", 100);
list[1] = new ProductData("grape", 200);
list[2] = new ProductData("orange", 300);
```

テーマ：表形式データ(2)

配列と複合データを用いた表形式データ

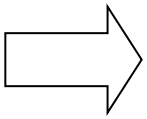
- *前回分*
  - *データの登録*
  - *データの検索*
  - *データの更新*
- **データの追加**

実際的はソフトウェアでは、表形式データの（例えば、データベースのデータ）を利用する場面が非常に多く、とても重要である。そこで、表形式を扱うプログラミングを繰り返しとりあげる。

データの追加

データの追加は、データの「登録」の一種である。すなわち「追加登録」である。

商品名	単価
apple	100
grape	200
orange	300



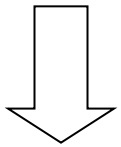
商品名	単価
apple	100
grape	200
orange	300
pear	400

listに商品(pear, 400)を追加する。

追加を行うためには、あらかじめ大きめの配列を確保しておくのが良い。すなわち、最初に登録している部分のあとに、未登録の部分を残す。基本的なアイディアは第7回(csvファイルの読み込みの部分)で扱った。

例1:整数の列を扱う(考え方)

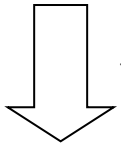
整数
0
2
4



整数
0
2
4
6

```
int numList = 5;
int [] list = new int[numList];
list[0] = 0;
list[1] = 2;
list[2] = 4;
```

5個整数を登録できる列



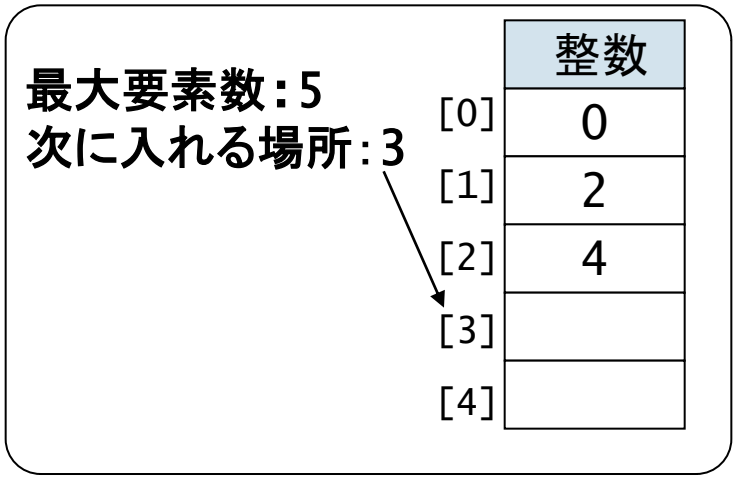
列に整数6を追加する。

すなわち、次に入れるべき場所
(配列の3番目)に6を入れる。

```
int[3] = 6;
```

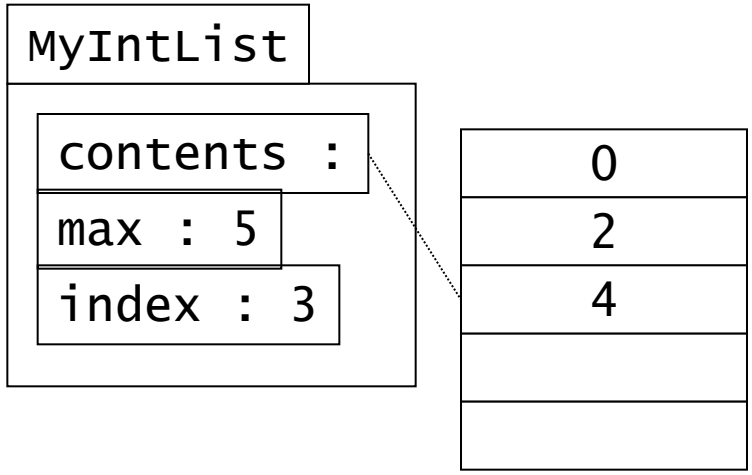
配列を含んだ複合データ

下記3種類の情報が必要であるため、これをまとめて複合データとする。
1： **要素（データ）を蓄積するスペース（配列）**
2： 登録可能な要素の最大の個数（配列の長さ）
3： 次に登録する要素を入れる場所
→ 配列の先頭（0番）から順番に要素が詰まって入っていることが前提




```
public class MyIntList {
   int [] contents;
    int max;
    int index;
}
```

実際には、配列の実体はインスタンスに含まれるのではなく、参照のみを持つので、下図のイメージとなる。



maxは、要素の最大の個数であるから
contents.lengthで、いつでも
計算可能であるが、ここでは
インスタンス変数として持たせている。
（少し冗長である。）

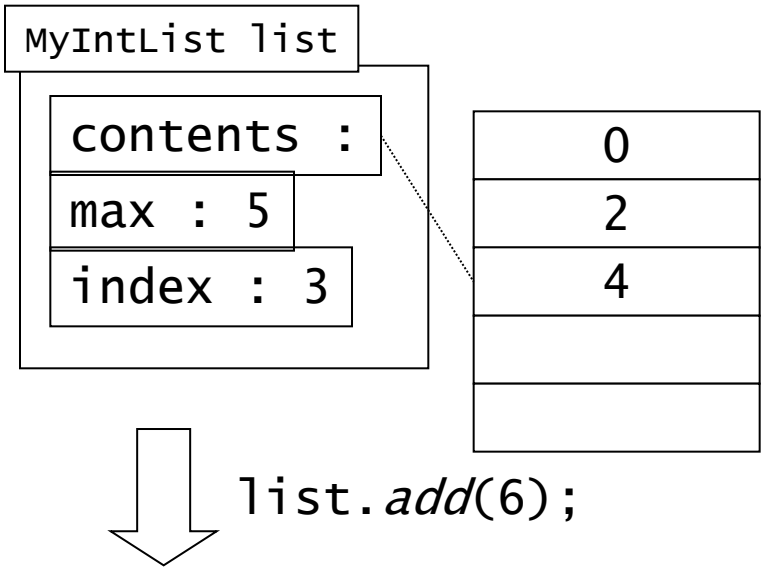
整数の列を保持する複合データ

```
public class MyIntList {
  int [] contents;
  int max;
  int index;

  MyIntList(int max) {
    this.contents = new int[max];
    this.max = max;
    this.index = 0;
  }

  void add(int num) {
    int next = this.index;
    this.contents[next] = num;
    this.index = this.index + 1;
    // this.index++ともかける
  }
}
```

配列の実体を生成することを忘れない。

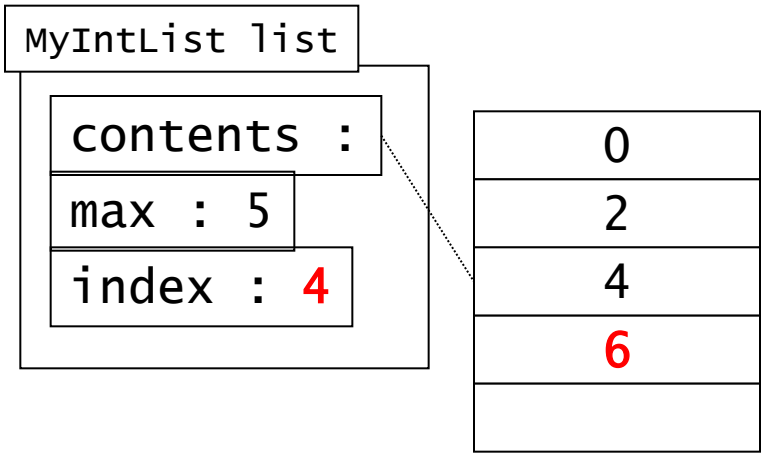
配列の次に登録すべき場所(this.contents[next])にデータnumを追加する。
次に登録すべき場所を更新する(1増やす)。
配列の範囲を超える場合の処理が必要だが、ここでは何も対処していない。

要素の追加(1)　3つあらかじめデータを入れておき、そのあとに整数6を列に追加する。




```
MyIntList list = new MyIntList(5);
list.contents[0] = 0;
list.contents[1] = 2;
list.contents[2] = 4;
list.index = 3; //次は3番目から登録する。
list.add(6);
```

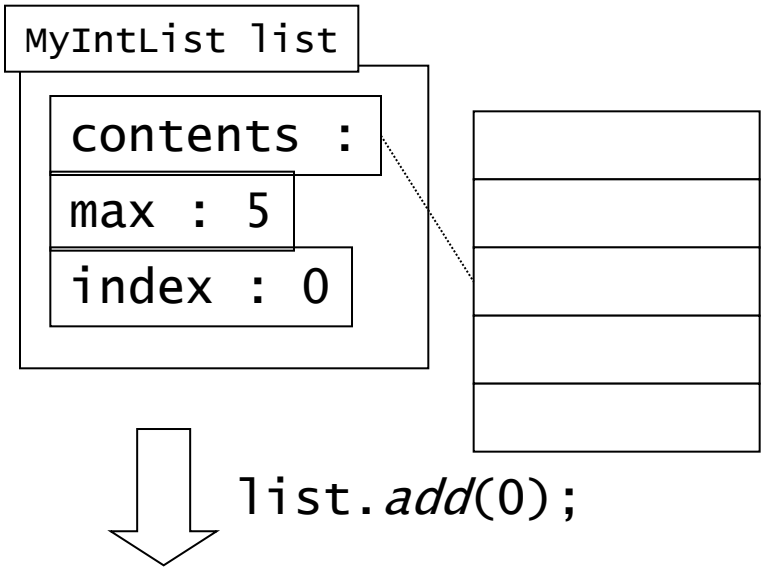
listに6を追加




```
void add(int num) {
  int next = this.index;
  this.contents[next] = num;
  this.index = this.index + 1;
  // this.index++ともかける
}
```

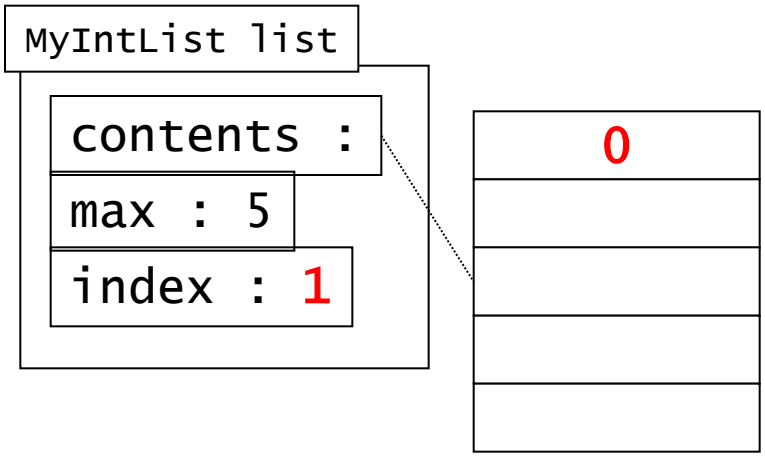
要素の追加(2)

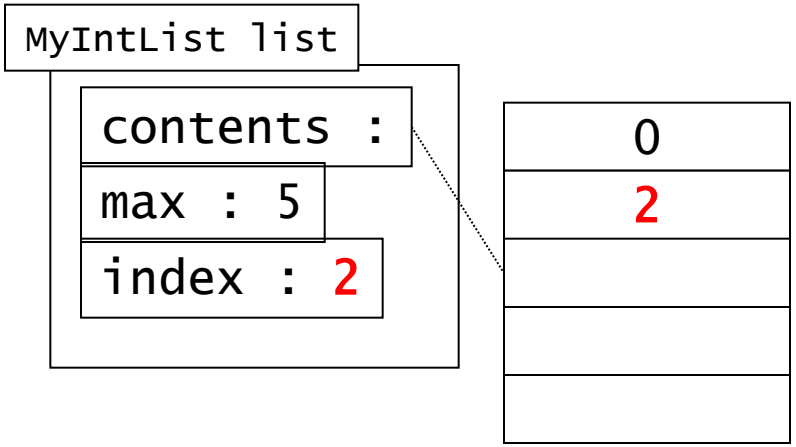
「3つあらかじめデータを入れて」おくのも、
空っぽのリストに3つデータを追加することだから、
addメソッドを用いればよい。




```
MyIntList list = new MyIntList(5);
list.add(0);
list.add(2);
list.add(4);

list.add(6);
```

list.*add*(0);



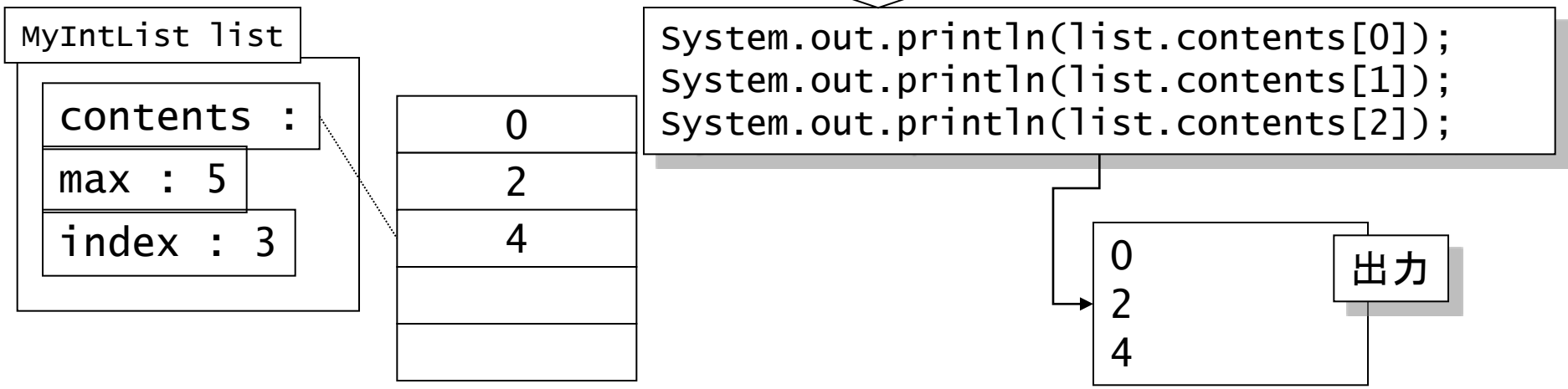


list.*add*(list, 2);

走査(scan)

この構造では、有効なデータが配列全体に格納されているわけではない。登録したデータの部分だけを走査する必要がある。

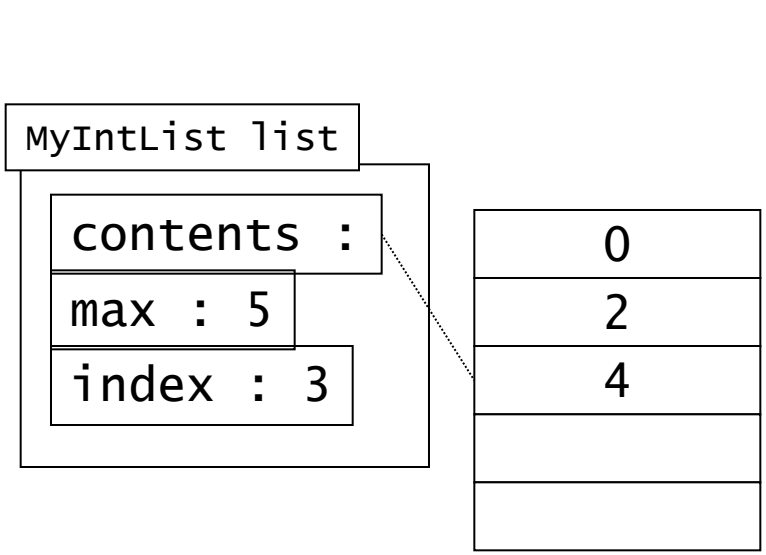
```
int [] array = list.contents;
for (int i = 0; i < list.index; i++ ) {
  System.out.println(array[i]);
}
```



配列全体を走査しない。
(0 <= i < list.index) の範囲を扱う。
(上の絵の例では、iは、0, 1, 2の値をとる。)

配列全体を走査すると…

```
for (int i = 0; i < list.max; i++ ) {
  System.out.println(array[i]);
}
```




```
System.out.println(list.contents[0]);
System.out.println(list.contents[1]);
System.out.println(list.contents[2]);
System.out.println(list.contents[3]);
System.out.println(list.contents[4]);
```

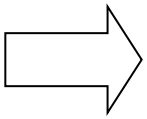
出力

```
0
2
4
0
0
```

最後の二つは、まだ登録されていない値を表示してしまう。
intの初期値は0なので、0が表示されているが、
ここでは意味のないデータである。

例2: 商品データの追加

商品名	単価
apple	100
grape	200
orange	300



商品名	単価
apple	100
grape	200
orange	300
pear	400

listに商品(pear, 400)を追加する。

クラスProductList

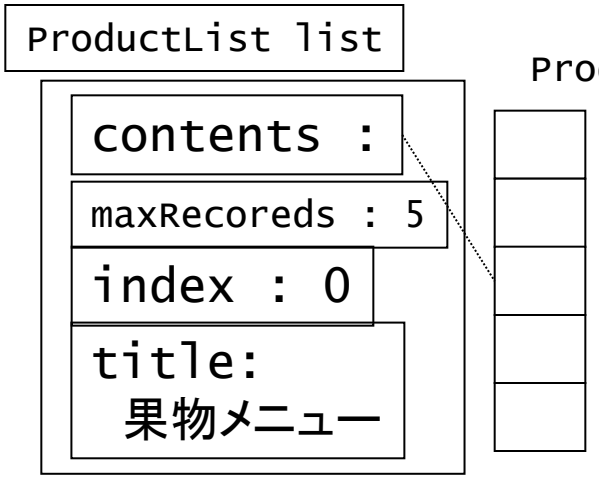
変数の型と名前	初期値	説明
String title	無し	一覧表の名前
int index	0	配列contentsのインデックス
Int maxRecords	無し	配列contentsのサイズ
ProductData [ ] contents	無し	ProductData型の配列

例題11より(一部)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class ProductList {
6      String title;
7      int index = 0;
8      int maxRecords;
9      ProductData [] contents;
10
11     ProductList(int maxRecords, String title) {
12         this.title = title;
13         this.maxRecords = maxRecords;
14         this.contents = new ProductData [maxRecords];
15     }
```

この例では、メニューに名前(タイトル)を付けることができるようになっている。

配列の実体を生成することを忘れない。




```
ProductList list
= new ProductList(5, “果物メニュー”)
```

注意

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class ProductList {
6      String title;
7      int index = 0;
8      int maxRecords;
9      ProductData [] contents;
10
11     ProductList(int maxRecords, String title) {
12         this.title = title;
13         this.maxRecords = maxRecords;
14         this.contents = new ProductData [maxRecords];
15     }
16 }
```

```
this.title = title;
```

こちらはインスタンス変数title

こちらはコンストラクタの引数title

両者は別物である。

引数の名前はインスタンス変数と揃える必要はない

```
ProductList(int max, String t) {
  this.title = t;
  this.maxRecords = max;
  this.contents = new ProductData [max];
}
```

例題13より(一部) 要素を追加するためのメソッド

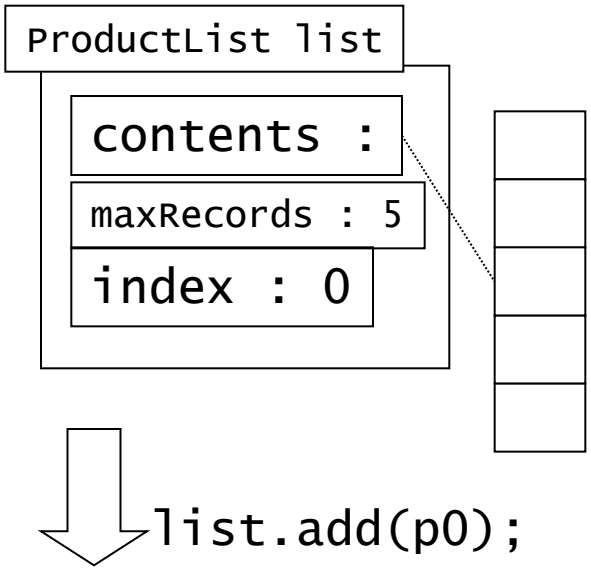
```
//例題13
ProductData add(ProductData p) {
    if (index == maxRecords) {
        return null; // これ以上、商品を登録できない。
    }
    contents[index] = p;
    index++;
    return p;
}
```

配列の次に登録すべき場所(this.contents[this.index])にデータを代入する。

次に登録すべき場所を更新する(1増やす)。

配列の範囲を超える場合nullを返す。追加出来た場合は、追加したデータを返す。

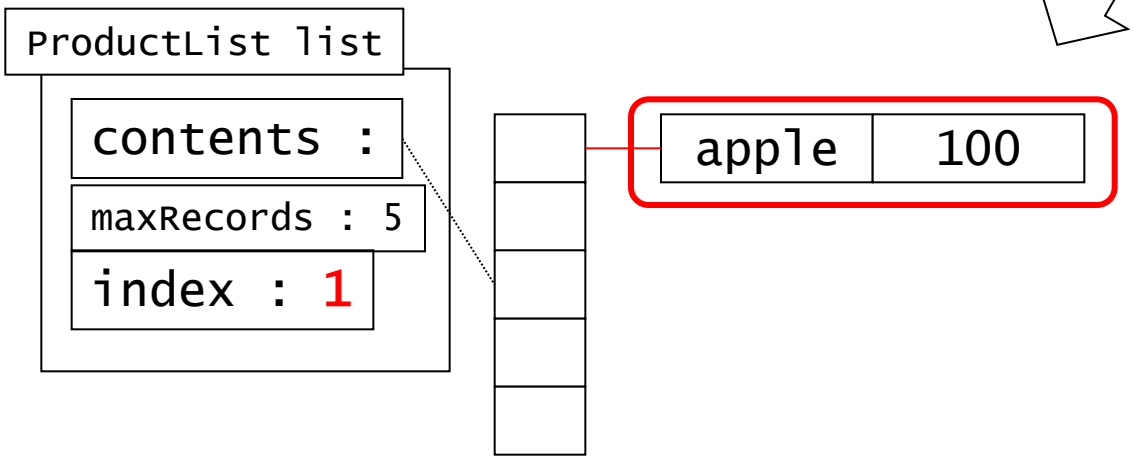
要素の追加:
追加したい複合データを作成してからリストに加える。

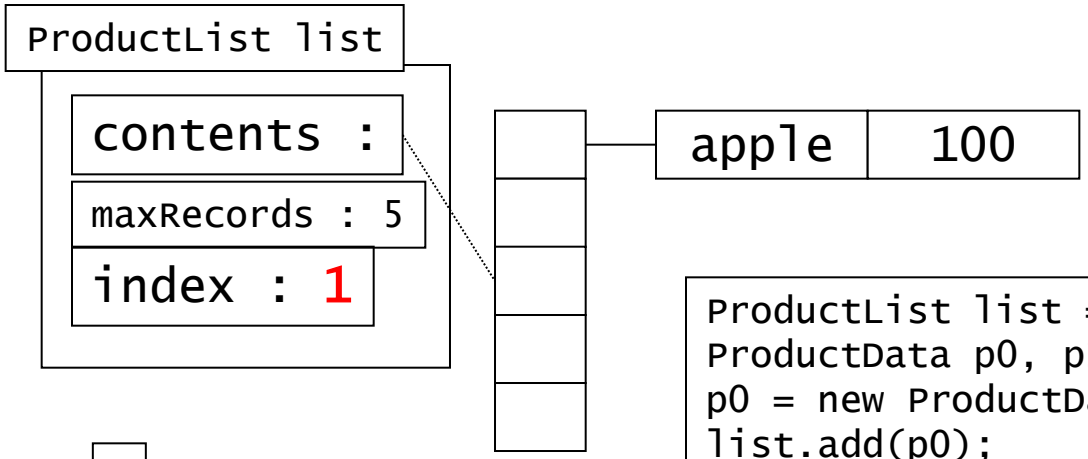



```
ProductList list = ProductList.create(5,“果物リスト”);
ProductData p0, p1;
p0 = new ProductData(“apple”, 100);
list.add(p0);
p1 = new ProductData(“grape”, 200);
list.add(p1);
```

p0 = new ProductData(“apple”,100);

list.add(p0);

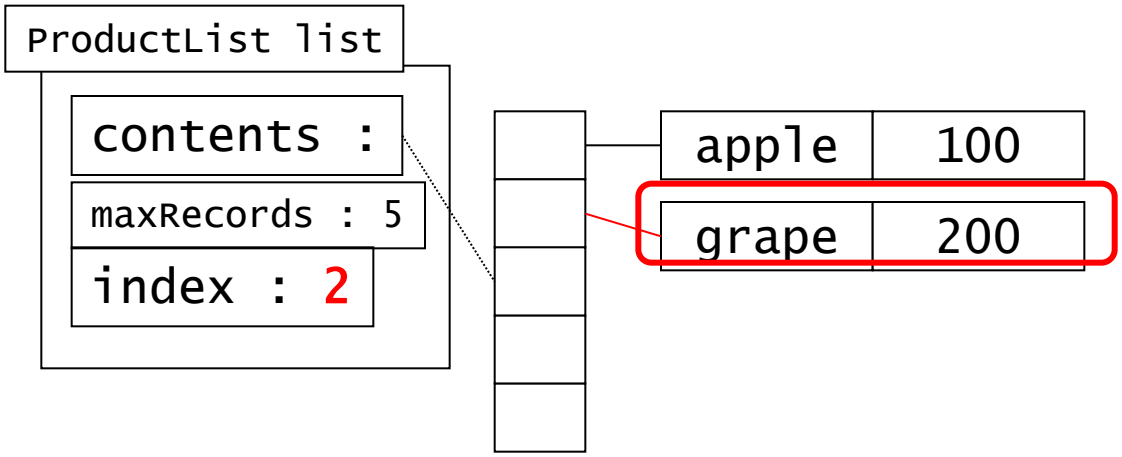





```
ProductList list = new ProductList(5,“果物リスト”);
ProductData p0, p1;
p0 = new ProductData(“apple”, 100);
list.add(p0);
p1 = new ProductData(“grape”, 200);
list.add(p1);
```

p1 = new ProductData(“grape”,200);

list.*add*(p1);



例題14より(一部)

ProductDataのインスタンスを作成してから、リストに追加するメソッドを用いる

```
38    ProductData add(String name, int price) {
39        ProductData p = new ProductData(name, price);
40        return add(p);
41    }
```

```
//例題13
ProductData add(ProductData p) {
    if (index == maxRecords) {
        return null; // これ以上、商品を登録できない。
    }
    contents[index] = p;
    index++;
    return p;
}
```

```
ProductList list = new ProductList(5, “果物リスト”);
list.add(“apple”, 100);
list.add(“grape”, 200);
```

```
ProductList list = new ProductList(5, “果物リスト”);
ProductData p0, p1;
p0 = new ProductData(“apple”, 100);
list.add(p0);
p1 = new ProductData(“grape”, 200);
list.add(p1);
```

まとめ:表形式データ(2)

配列と複合データを用いた表形式データ

- *前回分*
  - *データの登録*
  - *データの検索*
  - *データの更新*
- **データの追加**

実際的はソフトウェアでは、表形式データの（例えば、データベースのデータ）を利用する場面が非常に多く、とても重要である。そこで、表形式を扱うプログラミングを繰り返しとりあげた。

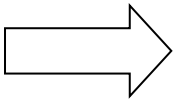
演習：商品オーダリングシステム

商品の注文システムを作成する。ネットショッピングをイメージするとよい。

(1) メニューの提示： 商品メニューを作成し、表示。
(2) 商品の注文：ユーザー（買い物をする人、顧客）は、注文リスト（買い物かご）に、注文を追加していく。その際、商品とその個数を指定する。
(3) 会計： 最後に、合計金額を算出し、表示する。

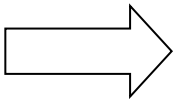
商品メニュー

商品名	単価
apple	100
grape	200
orange	300



買い物かご

商品（名前、単価）		数量
apple	100	2
orange	300	3



1100円

会計

商品をメニューから
選んで買い物かごに追加

セッション（やりとり）の例（問題13）

appleを2個買い物かごに入れる

スプレッドシート

ファイル(F) ウィンドウ(W) ヘルプ(H)

A	B	C
果物メニュー		
商品名	単価	
apple	100	
grape	200	
orange	300	
買い物かご		

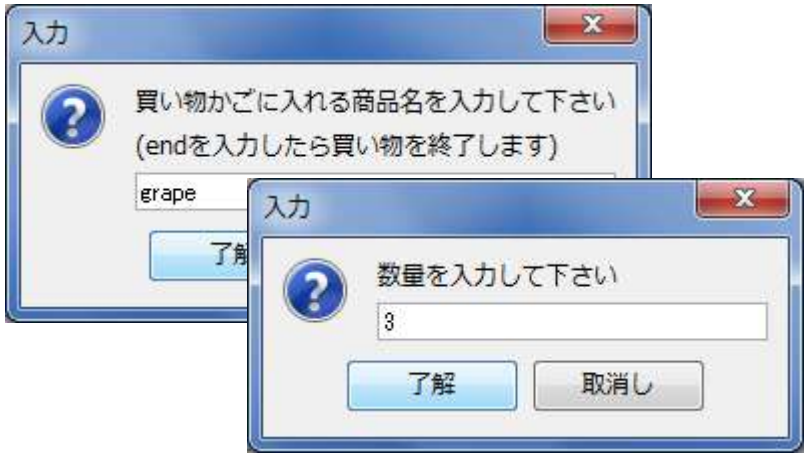
最初は買い物かごの中身は空

スプレッドシート

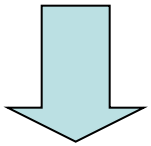
ファイル(F) ウィンドウ(W) ヘルプ(H)

A	B	C	D
果物メニュー			
商品名	単価		
apple	100		
grape	200		
orange	300		
買い物かご			
商品名	単価	個数	注文番号
apple	100	2	0

grapeを3個買い物かごに入れる



次頁へ

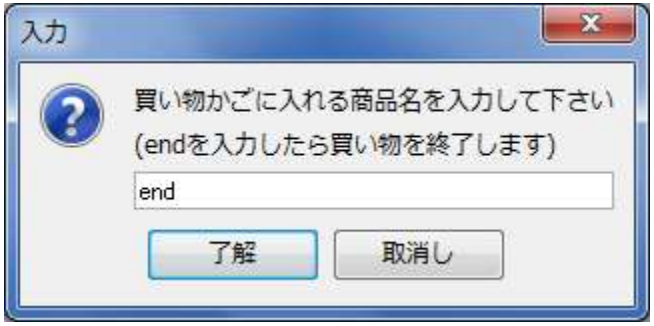


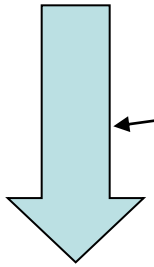
スプレッドシート

ファイル(F)　ウィンドウ(W)　ヘルプ(H)

A	B	C	D
果物メニュー			
商品名	単価		
apple	100		
grape	200		
orange	300		
買い物かご			
商品名	単価	個数	注文番号
apple	100	2	0
grape	200	3	1
合計金額	800円		

終了を指定する





合計金額 800円

本日の例題と問題

- 商品リストの作成
  - Ex11, Ex12, Ex13, Ex14
- 注文リストの作成
  - Ex21, Ex22, Ex23, Ex24
- 発注システムの構築
  - Ex31(1), Ex31(2), Ex31(3), Q11, Q12, Q13*, Q21, Q22, Q23, Q24*, (Q25*)

(Ex:例題, Q:問題, *は少し手間のかかる問題)

各自に適した順番で解けばよいが、上記の順番が自然な流れとなるよう構成されている。

例題集

パッケージ「j2.lesson09」を作成する。

パッケージやクラスの作成，実行の仕方の説明は省略する。
作り方を忘れた場合は過去のスライドや
http://java2010.cis.k.hosei.ac.jp/01/material-01/
を参考にせよ

■例題1

内容: 商品の一覧表を作成することを考える。登録可能な最大数までの範囲内で、任意の件数の商品データを一覧表に登録できるものとする。

商品名	単価
apple	100
grape	200
orange	300

■例題11

問題(a):商品の情報を表すクラスProductDataを作成し、
動作を確認せよ

インスタンス変数

変数の型と名前	初期値	説明
String name	無し	商品の名前
int price	無し	商品の値段

コンストラクタ

コンストラクタ(引数)	機能
ProductData(String name, int price)	ProductDataクラスのインスタンスを作成し、各インスタンス変数にそれぞれ引数の値を代入する。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	showProductData(int row)	ProductDataインスタンスをSpreadsheetのrow行に表示する。
void	header(int row)	Spreadsheetの row行にヘッダを表示する。

（クラス名： ProductData）
動作確認クラス:Ex11TestProductList

■例題11(続き)

問題(b):商品の一覧を表すクラスProductListを作成せよ。

インスタンス変数

変数の型と名前	初期値	説明
String title	無し	一覧表の名前
int index	0	配列contentsのインデックス
Int maxRecords	無し	配列contentsのサイズ
ProductData [ ] contents	無し	ProductData型の配列

コンストラクタ

コンストラクタ(引数)	機能
ProductList(int maxRecords, String title)	ProductListクラスのインスタンスを作成し、各インスタンス変数にそれぞれ引数の値を代入する。

(クラス名： ProductList)

例題11(ProductData)

```
 1 package j2.lesson09;
 2
 3 import gpjava.Spreadsheet;
 4
 5 public class ProductData {
 6     String name;
 7     int price;
 8
 9     ProductData(String name, int price) {
10         this.name = name;
11         this.price = price;
12     }
13
14     void showProductData(int row) {
15         Spreadsheet.setString(row, 0, name);
16         Spreadsheet.setInt(row, 1, price);
17     }
18
19     void header(int row) {
20         Spreadsheet.setString(row, 0, "商品名");
21         Spreadsheet.setString(row, 1, "単価");
22     }
23 }
```

例題11(ProductDataの動作確認)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex11TestProductList {
6      public static void main(String[] args) {
7          new Ex11TestProductList().start();
8      }
9
10     void start() {
11         ProductData p0 = new ProductData("apple", 200);
12         Spreadsheet.show();
13         p0.header(0);
14         p0.showProductData(1);
15     }
16 }
```

例題11(ProductList)

```
1  package j2.lesson09;
2
3  public class ProductList {
4      String title;
5      int index = 0;
6      int maxRecords;
7      ProductData [] contents;
8
9      ProductList(int maxRecords, String title) {
10         this.title = title;
11         this.maxRecords = maxRecords;
12         this.contents = new ProductData [maxRecords];
13     }
14 }
```

■例題12

問題：ProductListクラスに商品メニュー一覧を表示する
　contentsメソッドを作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	contents(int row)	ProductListの内容を Spreadsheet の row行に表示する。

※既存のProductListクラスを編集する。
動作確認クラス：Ex12TestProductList

例題12(ProductList)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class ProductList {
6      String title;
7      int index = 0;
8      int maxRecords;
9      ProductData [] contents;
10
11     ProductList(int maxRecords, String title) {
12         this.title = title;
13         this.maxRecords = maxRecords;
14         this.contents = new ProductData [maxRecords];
15     }
16     //例題12
17     void contents(int row) {
18         Spreadsheet.setString(row, 0, title);
19         if(index > 0) {
20             contents[0].header(row + 1);
21             for (int i = 0; i < index; i++) {
22                 contents[i].showProductData(row + i + 2);
23             }
24         }
25     }
26 }
```

例題12(Ex12TestProductList)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex12TestProductList {
6      public static void main(String[] args) {
7          new Ex12TestProductList().start();
8      }
9
10     void start() {
11         ProductData p0 = new ProductData("apple", 100);
12         ProductData p1 = new ProductData("grape", 200);
13
14         ProductList list = new ProductList(10, "青果メニュー");
15         list.contents[0] = p0;
16         list.contents[1] = p1;
17         list.index = 2;
18
19         Spreadsheet.show();
20         list.contents(0);
21     }
22 }
```

■例題13

問題：ProductListクラスに商品メニューに新たにProductDataのインスタンスを追加するメソッドaddを作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
ProductData	add(ProductData p)	引数pをProductListに追加する。 既に商品リストの要素がいっぱいで追加できない場合はnullを返す。 追加出来た場合は、追加したデータ(つまりp)を返す。
既存インスタンスメソッド(ProductList内)		
void contents(int row)		

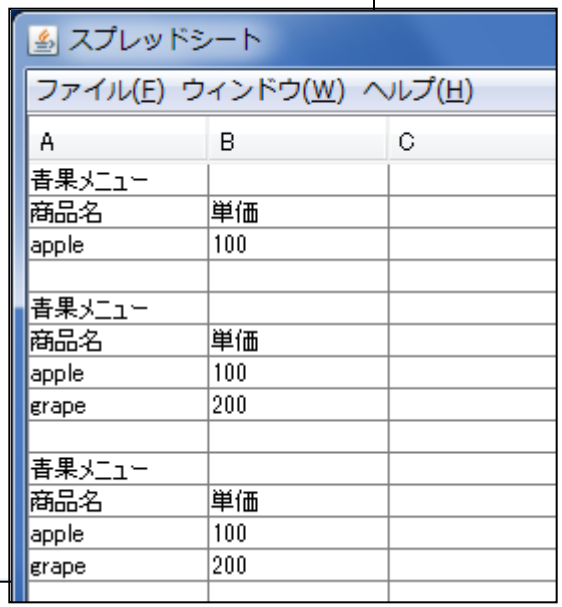
※既存のProductListクラスを編集する。
動作確認クラス：Ex13TestProductList

例題13(ProductList)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class ProductList {
6      String title;
7      int index = 0;
8      int maxRecords;
9      ProductData [] contents;
10
11     ProductList(int maxRecords, String title) {
12         this.title = title;
13         this.maxRecords = maxRecords;
14         this.contents = new ProductData [maxRecords];
15     }
16     //例題12
17     void contents(int row) {..
26
27
28     //例題13
29     ProductData add(ProductData p) {
30         if (index == maxRecords) {
31             return null; // これ以上、商品を登録できない。
32         }
33         contents[index] = p;
34         index++;
35         return p;
36     }
37 }
```

例題13(Ex13TestProductList)

```java
package j2.lesson09;

import gpjava.Spreadsheet;

public class Ex13TestProductList {
    public static void main(String[] args) {
        new Ex13TestProductList().start();
    }

    void start() {
        ProductData p0 = new ProductData("apple", 100);
        ProductData p1 = new ProductData("grape", 200);
        ProductData p2 = new ProductData("grape", 200);

        ProductList list = new ProductList(2, "青果メニュー");

        int row = 0;

        Spreadsheet.show();
        list.add(p0);
        list.contents(row);
        row += list.index + 3;

        list.add(p1);
        list.contents(row);
        row += list.index + 3;

        // 3つ以上は、データは登録されない。
        System.out.println(list.add(p2));
        list.contents(row);
    }
}
```



■例題14

問題：ProductListクラスに次のメソッドaddを作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
ProductData	add(String name, int price)	引数name,priceを基にProductDataインスタンスを作成しProductListに追加する。 ※例題13のaddメソッドとは、引数部が違うことに留意せよ。（メソッドのオーバーロード）。内部で例題13のaddメソッドを呼ぶ。
既存インスタンスメソッド(ProductList内)		
void contents(int row)		
ProductData add(ProductData p) (例題13)		

※既存のProductListクラスを編集する。

動作確認クラス：Ex14TestProductList

例題14(ProductList)

```
 1  package j2.lesson09;
 2
 3  import gpjava.Spreadsheet;
 4
 5  public class ProductList {
 6      String title;
 7      int index = 0;
 8      int maxRecords;
 9      ProductData [] contents;
10
11      ProductList(int maxRecords, String title) {
12          this.title = title;
13          this.maxRecords = maxRecords;
14          this.contents = new ProductData [maxRecords];
15      }
16      //例題12
17      void contents(int row) {..
26
27
28      //例題13
29      ProductData add(ProductData p) {..
37
38      //例題14
39      ProductData add(String name, int price) {
40          ProductData p = new ProductData(name, price);
41          return add(p);
42      }
43  }
```

例題14(Ex14TestProductList)

```java
package j2.lesson09;

import gpjava.Spreadsheet;

public class Ex14TestProductList {
    public static void main(String[] args) {
        new Ex14TestProductList().start();
    }

    void start() {
        ProductList list = new ProductList(2, "青果メニュー");

        int row = 0;

        Spreadsheet.show();
        list.add("apple", 100);
        list.contents(row);
        row += list.index + 3;

        list.add("grape", 200);
        list.contents(row);
        row += list.index + 3;

        // 3つ以上は、データは登録されない。
        System.out.println(list.add("grape", 2002));
        list.contents(row);
    }
}
```

■例題2

内容: 例題2では、商品の注文システムを作成するために、注文内容を記憶するための 注文一覧表を作成する。
例題1と同様、登録可能な最大数までの範囲内で、任意の件数の注文データを一覧表に登録できるものとする。

買い物かご

商品(名前、単価)		数量
apple	100	2
orange	300	3

■例題21

問題(a):一件の注文(注文データ)を表すクラスOrderDataを作成し、動作を確認せよ。

インスタンス変数

変数の型と名前	初期値	説明
ProductData product	無し	商品(ProductDataオブジェクト)
int num	無し	商品の個数

コンストラクタ

コンストラクタ引数)	機能
OrderData(ProductData product, int num)	OrderDataクラスのインスタンスを作成し、各インスタンス変数にそれぞれ引数の値を代入する。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	showOrderData(int row)	OrderDataインスタンスをSpreadsheetのrow行に表示する。
void	header(int row)	Spreadsheetのrow行にヘッダを表示する。

(クラス名: OrderData)
動作確認クラス:Ex21TestOrderData

例題21(OrderData)

```
 1 package j2.lesson09;
 2
 3 import gpjava.Spreadsheet;
 4
 5 public class OrderData {
 6     ProductData product; // product name
 7     int num;
 8
 9     OrderData(ProductData product, int num) {
10         this.product = product;
11         this.num = num;
12     }
13
14     void showOrderData(int row) {
15         product.showProductData(row);
16         Spreadsheet.setInt(row, 2, num);
17     }
18
19     void header(int row) {
20         product.header(row);
21         Spreadsheet.setString(row, 2, "個数");
22     }
23 }
```

例題21(OrderDataの動作確認)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex21TestOrderData {
6      public static void main(String[] args) {
7          new Ex21TestOrderData().start();
8      }
9
10     void start() {
11         ProductData p0 = new ProductData("apple", 200);
12         OrderData o0 = new OrderData(p0, 2);
13
14         Spreadsheet.show();
15         o0.header(0);
16         o0.showOrderData(1);
17     }
18
19 }
```

■例題21(続き)

問題(b):注文内容の一覧表(以下、買い物かご)を表すクラス OrderListを作成せよ。

インスタンス変数

変数の型と名前	初期値	説明
String title	無し	買い物かごの名前
int index	0	配列contentsのインデックス
Int maxRecords	無し	配列contentsのサイズ
OrderData [ ] contents	無し	OrderData型の配列

コンストラクタ

メソッド名(引数)	機能
OrderList(int maxRecords, String title)	OrderListクラスのインスタンスを作成し、各インスタンス変数にそれぞれ引数の値を代入する。

(クラス名: OrderList)

例題21(OrderList)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  import javax.swing.JOptionPane;
6
7  public class OrderList {
8
9      String title;
10     int index = 0;
11     int maxRecords;
12     OrderData[] contents;
13
14     OrderList(int maxRecords, String title) {
15         this.title = title;
16         this.maxRecords = maxRecords;
17         this.contents = new OrderData[maxRecords];
18     }
19 }
```

■例題22

問題：OrderListクラスに買い物カゴ（注文内容一覧）を表示するcontentsメソッドとそのヘッダ情報を表示する headerメソッドを作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	contents(int row)	買い物かごlistをSpreadsheet の row 行に表示する。
void	header(int row)	Spreadsheetのrow行にヘッダを表示する。

※既存のOrderListクラスを編集する。
動作確認クラス：Ex22TestOrderList

例題22(OrderList)

```
 1  package j2.lesson09;
 2
 3  import gpjava.Spreadsheet;
 4
 5  public class OrderList {
 6
 7      String title;
 8      int index = 0;
 9      int maxRecords;
10      OrderData[] contents;
11
12      OrderList(int maxRecords, String title) {
13          this.title = title;
14          this.maxRecords = maxRecords;
15          this.contents = new OrderData[maxRecords];
16      }
17
18      // 例題21->問題21
19      void contents(int row) {
20          Spreadsheet.setString(row, 0, title);
21          if(index > 0) {
22              header(row + 1);
23              for (int i = 0; i < index; i++) {
24                  contents[i].showOrderData(row + i + 2);
25              }
26          }
27      }
28
29      void header(int row) {
30          contents[0].header(row);
31      }
32  }
```

例題22(Ex22TestOrderList)

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex22TestOrderList {
6      public static void main(String[] args) {
7          new Ex22TestOrderList().start();
8      }
9
10     void start() {
11         ProductData p0 = new ProductData("apple", 100);
12         ProductData p1 = new ProductData("grape", 200);
13
14         OrderData o0 = new OrderData(p0, 2);
15         OrderData o1 = new OrderData(p1, 3);
16
17         OrderList order = new OrderList(10, "買い物かごの中身");
18         order.contents[0] = o0;
19         order.contents[1] = o1;
20         order.index = 2;
21
22         Spreadsheet.show();
23         order.contents(0);
24     }
25 }
```

■例題23

問題：買い物かごに新たにOrderDataのインスタンスを追加するメソッドadd
　　を作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
ProductData	add(OrderData o)	注文データoをProductListに追加する。 既に商品リストの要素がいっぱいで追加できない場合nullを返す。 追加に成功した場合は追加したインスタンスを返す
既存インスタンスメソッド(OrderList内)		
void contents(int row)		
void header(int row)		

※既存のOrderListクラスを編集する。
動作確認クラス：Ex23TestOrderList

例題23(OrderList)

```
 1 package j2.lesson09;
 2
 3 import gpjava.Spreadsheet;
 4
 5 import javax.swing.JOptionPane;
 6
 7 public class OrderList {
 8
 9     String title;
10     int index = 0;
11     int maxRecords;
12     OrderData[] contents;
13
14     OrderList(int maxRecords, String title) {..
19
20     // 例題21->問題21
21     void contents(int row) {..
31
32     void header(int row) {..
36
37     // 例題23
38     OrderData add(OrderData o) {
39         if (index == maxRecords) {
40             return null;
41         }
42         contents[index] = o;
43         index++;
44         return o;
45     }
46 }
```

例題23(Ex23TestOrderList)

```java
package j2.lesson09;

import gpjava.Spreadsheet;

public class Ex23TestOrderList {
    public static void main(String[] args) {
        new Ex23TestOrderList().start();
    }

    void start() {
        ProductData p0 = new ProductData("apple", 100);
        ProductData p1 = new ProductData("grape", 200);

        ProductList list = new ProductList(2, "青果メニュー");
        list.add(p0);
        list.add(p1);

        int row = 0;
        Spreadsheet.show();

        list.contents(row);
        row += list.index + 3;

        OrderList order = new OrderList(10, "買い物かごの中身");

        OrderData o0 = new OrderData(p0, 2);
        OrderData o1 = new OrderData(p1, 3);

        order.add(o0);
        order.add(o1);

        order.contents(row);
    }
}
```

A	B	C	D
青果メニュー			
商品名	単価		
apple	100		
grape	200		
買い物かごの中身			
商品名	単価	個数	
apple	100	2	
grape	200	3	

■例題24

問題：OrderListクラスに次のメソッドaddを作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
OrderData	add(ProductData p, int amount)	引数p,amountを基にOrderDataインスタンスを作成しOrderListに追加する。 ※例題23のaddメソッドとは、引数部が違うことに留意せよ。（メソッドのオーバーロード）。内部で例題23のaddメソッドを呼ぶ。
既存インスタンスメソッド(OrderList内)		
void contents(int row)		
void header(int row)		
OrderData add(OrderData o) （例題23)		

※既存のOrderListクラスを編集する。
動作確認クラス：Ex24TestOrderList

例題24(OrderList)

```
 1  package j2.lesson09;
 2
 3  import gpjava.Spreadsheet;
 4
 5  import javax.swing.JOptionPane;
 6
 7  public class OrderList {
 8
 9      String title;
10      int index = 0;
11      int maxRecords;
12      OrderData[] contents;
13
14      OrderList(int maxRecords, String title) {..
19
20      // 例題21->問題21
21      void contents(int row) {..
31
32      void header(int row) {..
36
37      // 例題23
38      OrderData add(OrderData o) {..
46
47      // 例題24
48      OrderData add(ProductData p, int amount) {
49          OrderData o = new OrderData(p, amount);
50          return add(o);
51      }
52  }
```

例題24(Ex24TestOrderList)

```java
package j2.lesson09;

import gpjava.Spreadsheet;

public class Ex24TestOrderList {
    public static void main(String[] args) {
        new Ex24TestOrderList().start();
    }

    void start() {
        ProductData p0, p1;
        ProductList list = new ProductList(2, "青果メニュー");

        p0 = list.add("apple", 100);
        p1 = list.add("grape", 200);

        int row = 0;
        Spreadsheet.show();

        list.contents(row);
        row += list.index + 3;

        OrderList order = new OrderList(10, "買い物かごの中身");
        order.add(p0, 2);
        order.add(p1, 3);

        order.contents(row);
    }
}
```

スプレッドシート

ファイル(F)　ウィンドウ(W)　ヘルプ(H)

A	B	C	D
青果メニュー			
商品名	単価		
apple	100		
grape	200		
買い物かごの中身			
商品名	単価	個数	
apple	100	2	
grape	200	3	

■例題3

例題3，問題1, 問題2では、次のような商品オーダーシステムを実際に作成していく。例題1，2で作成したクラスを利用する。

このシステムは、ユーザー（商品購入者）が、商品メニューから購入する商品を選択して注文リストに追加し、購入する商品が全てきまったら合計金額を確かめる（会計を行う）というシステムである。

「商品メニュー」：取り扱う商品のデータ（商品名と価格）の一覧表である。
「注文リスト（買い物かご）」：注文内容の一覧である。1件の注文は商品のデータと購入する数量からなり、複数件の注文をリストとしてもつ。最大10件（例）まで登録できるものとする。

システムの機能は、次のようなものである。詳しくは問題13を参照せよ。

(1) メニューの提示： 商品メニューを作成し、表示。
(2) 商品の注文：ユーザー（買い物をする人、顧客）は、注文リスト（買い物かご）に、注文を追加していく。その際、商品とその個数を指定する。
(3) 会計： 最後に、合計金額を算出し、表示する。

■例題31(1)

問題：次の商品メニューを作成するメソッドsetFruitsMenu()を作成せよ。

返り値の型	メソッド名(引数)	機能
ProductList	setFruitsMenu( )	商品メニュー(タイトルは「果物メニュー」、配列のサイズは10)を作成し、以下の商品データを持たせ、その商品リストを返す。 商品名：apple　単価:100 grape　200 orange　300

（クラス名： Ex31OrderSystem ）

例題31(1) Ex31OrderSystem

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex31OrderSystem_1 {
6      public static void main(String[] args) {
7          new Ex31OrderSystem_1().start();
8      }
9
10     void start() {
11         ProductList menu = setFruitsMenu();
12         Spreadsheet.show();
13         menu.contents(0);
14     }
15
16     ProductList setFruitsMenu(){
17         ProductList list = new ProductList(10, "果物メニュー");
18         list.add("apple", 100);
19         list.add("grape", 200);
20         list.add("orange", 300);
21         return list;
22     }
23 }
```

■例題31(2)

問題：指定する名前をもつ商品データを商品メニューから取得するメソッドgetを作成し、動作を確認せよ。

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
ProductData	get(String name)	引数nameと同じ名前を持つ商品データをProductList（商品メニュー）から取得し、その商品データを返す。 見つからない場合はnullを返す。
既存インスタンスメソッド(ProductList内)		
void contents(int row)		
void header(int row)		
ProductData add(ProductData p)		
ProductData add(String name, int price)		

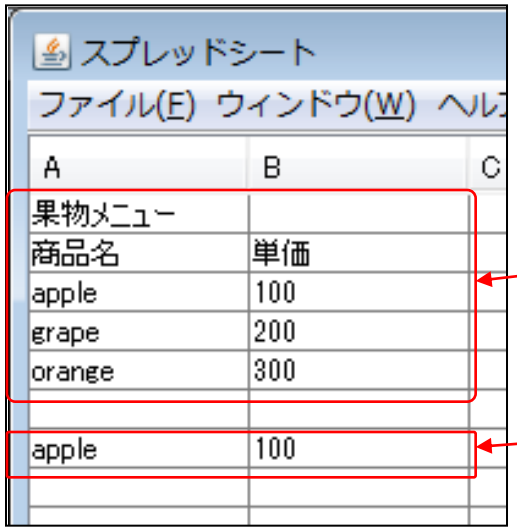
※既存のProductListクラスを編集する。

動作確認クラス：既存のEx31OrderSystemを編集

例題31(2)
ProductList

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  public class ProductList {
6      String title;
7      int index = 0;
8      int maxRecords;
9      ProductData [] contents;
10
11     ProductList(int maxRecords, String title) {
12         this.title = title;
13         this.maxRecords = maxRecords;
14         this.contents = new ProductData [maxRecords];
15     }
16     //例題12
17     void contents(int row) {..
26
27
28     //例題13
29     ProductData add(ProductData p) {..
37
38     //例題14
39     ProductData add(String name, int price) {..
43
44     //例題3(2)
45     ProductData get(String name) {
46         for (int i = 0; i < index; i++ ) {
47             ProductData p = contents[i];
48             if (p.name.equals(name)){
49                 return p;
50             }
51         }
52
53         return null;
54     }
55 }
```

例題31(2) Ex31OrderSystem




```java
package j2.lesson09;

import gpjava.Spreadsheet;

public class Ex31OrderSystem_2 {
    public static void main(String[] args) {
        new Ex31OrderSystem_2().start();
    }

    void start() {
        ProductList menu = setFruitsMenu();

        Spreadsheet.show();
        int row = 0;

        menu.contents(row);
        row += menu.index + 3;

        ProductData p = menu.get("apple");

        p.showProductData(row);

        p = menu.get("pear");
        System.out.println(p); // nullが表示される。
    }

    // 例題3(1)
    ProductList setFruitsMenu(){..
}
```

■例題31(3)

問題：買い物かごの合計金額を算出するメソッドsumを作成し、動作を確認せよ。

インスタンスメソッド

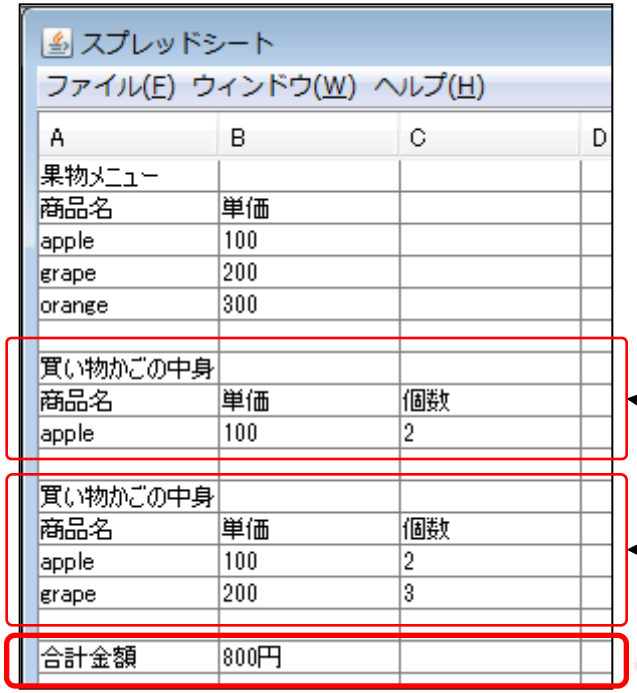
返り値の型	メソッド名(引数)	機能
int	int sum()	買い物かごの合計金額を算出し、その値を返す。
既存インスタンスメソッド(OrderList内)		
void contents(int row)		
void header(int row)		
OrderData add(OrderData o)		
OrderData add(ProductData p, int amount)		

※既存のOrderListクラスを編集する。
動作確認クラス：既存のEx31OrderSystemを編集

例題31(3)
OrderList

```
1  package j2.lesson09;
2
3  import gpjava.Spreadsheet;
4
5  import javax.swing.JOptionPane;
6
7  public class OrderList {
8
9      String title;
10     int index = 0;
11     int maxRecords;
12     OrderData[] contents;
13
14     OrderList(int maxRecords, String title) {
15         this.title = title;
16         this.maxRecords = maxRecords;
17         this.contents = new OrderData[maxRecords];
18     }
19
20     // 例題21->問題21
21     void contents(int row) {..
31
32     void header(int row) {..
36
37     // 例題23
38     OrderData add(OrderData o) {..
46
47     // 例題24
48     OrderData add(ProductData p, int amount) {..
52
53     // 例題3(3)
54     int sum() {
55         int sum = 0;
56         for (int i = 0; i < index; i++) {
57             OrderData o = contents[i];
58             sum += o.product.price * o.num;
59         }
60         return sum;
61     }
62 }
```

例題31(3)
Ex31OrderSystem



1品目目

2品目目

合計

```java
package j2.lesson09;

import gpjava.Spreadsheet;

public class Ex31OrderSystem_4 {
    public static void main(String[] args) {
        new Ex31OrderSystem_4().start();
    }

    void start() {
        ProductList menu = setFruitsMenu();

        Spreadsheet.show();
        int row = 0;

        menu.contents(row);
        row += menu.index + 3;

        ProductData p0 = menu.get("apple");
        ProductData p1 = menu.get("grape");

        OrderList order = new OrderList(10, "買い物かごの中身");

        order.add(p0, 2);
        order.contents(row);
        row += order.index + 3;

        order.add(p1, 3);
        order.contents(row);
        row += order.index + 3;

        Spreadsheet.setString(row, 0, "合計金額");
        Spreadsheet.setString(row, 1, order.sum() + "円");
    }

    ProductList setFruitsMenu(){
        ProductList list = new ProductList(10, "果物メニュー");
        list.add("apple", 100);
        list.add("grape", 200);
        list.add("orange", 300);
        return list;
    }
}
```