LẬP TRÌNH CHO CÁC THIẾT BỊ DI ĐỘNG

Bài 7: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG-ARRAYLIST-LISTVIEW

Lập trình hướng đối tượng
Lớp trong Java

- Lớp được xem như một khuôn mẫu (template) của đối tượng (Object).
- Trong lớp bao gồm các thuộc tính của đối tượng (properties) và các phương thức (methods) tác động lên các thuộc tính.
- Đối tượng được xây dựng từ lớp nên được gọi là thể hiện của lớp (class instance).

Khai báo lớp

class <ClassName>

{

<kiểu dữ liệu> <field_1>;

<kiểu dữ liệu> <field_2>;

constructor

method_1

method_2

}

Thuộc tính của lớp

- Vùng dữ liệu (fields) hay thuộc tính (properties) của lớp được khai báo bên trong lớp như sau:

```
class <ClassName>

        // khai báo những thuộc tính của lớp
        <tiền tố> <kiểu dữ liệu> field1;
        // …
}
```

- Để xác định quyền truy xuất của các đối tượng khác đối với vùng dữ liệu của một lớp người ta thường dùng 3 tiền tố sau:
  - public: có thể truy xuất từ tất cả các đối tượng khác
  - private: một lớp không thể truy xuất vùng private của một lớp khác.
  - protected: vùng protected của một lớp chỉ cho phép bản thân lớp đó và những lớp dẫn xuất từ lớp đó truy cập đến.

Phương thức (method) của lớp

- Hàm hay phương thức (method) trong Java là khối lệnh thực hiện các chức năng, các hành vi xử lý của lớp lên vùng dữ liệu.
- Khai báo phương thức:

<Tiền tố> <kiểu trả về> <Tên phương thức> (<danh sách đối số>)

{

<khối lệnh>;

}

Phương thức (method) của lớp

- Để xác định quyền truy xuất của các đối tượng khác đối với các phương thức của lớp người ta thường dùng các tiền tố sau:

  public, protected, private, static, final, abstract, synchronized
  - public: phương thức có thể truy cập được từ bên ngoài lớp khai báo.
  - protected: có thể truy cập được từ lớp khai báo và những lớp dẫn xuất từ nó.
  - private: chỉ được truy cập bên trong bản thân lớp khai báo.
  - static: phương thức lớp dùng chung cho tất cả các thể hiện của lớp, có nghĩa là phương thức đó có thể được thực hiện kể cả khi không có đối tượng của lớp chứa phương thức đó.
- <kiểu trả về>: có thể là kiểu void, kiểu cơ sở hay một lớp.
- <Tên phương thức>: đặt theo qui ước giống tên biến.
- <danh sách tham số>: có thể rỗng

Ví dụ

```
public class xemay
{
    public String nhasx;
    public String model;
    private float chiphisx;
    protected int thoigiansx;
    // so luong so cua xe may: 3, 4 so
    protected int so;
    // là biến tĩnh có giá trị là 2 trong tất cả các thể hiện tạo ra từ lớp
    xemay
    public static int sobanhxe = 2;
    public float tinhgiaban() { return 1.5 * chiphisx; }
}
```

Khởi tạo một đối tượng

- Constructor thật ra là một loại phương thức đặc biệt của lớp.
- Constructor dùng gọi tự động khi khởi tạo một thể hiện của lớp, có thể dùng để khởi gán những giá trị măc định.
- Các constructor không có giá trị trả về, và có thể có tham số hoặc không có tham số.
- Constructor phải có cùng tên với lớp và được gọi đến khi dùng từ khóa new.
- Nếu một lớp không có constructor thì Java sẽ cung cấp cho lớp một constructor mặc định (default constructor). Những thuộc tính, biến của lớp sẽ được khởi tạo bởi các giá trị mặc định (số: thường là giá trị 0, kiểu luận lý là giá trị false, kiểu đối tượng giá trị null, …)

```
public class xemay
  {          // …
             public xemay() { }
             public xemay(String s_nhasx, String s_model,
                                  f_chiphisx, int i_thoigiansx, int i_so);
             {           nhasx = s_nhasx;
                         model = s_model;
                         chiphisx = f_chiphisx;
                         thoigiansx = i_thoigiansx;
                         so = i_so;
                         // hoặc
                         // this.nhasx = s_nhasx;
                         // this.model = s_model;
                         // this.chiphisx = f_chiphisx;
                         // this.thoigiansx = i_thoigiansx;
                         // this.so = i_so;
             }
}
```

ArrayList

- Theo mặc định, các mảng trong Java bị cố định phần tử và không thể thay đổi. Nhưng thực tế nhu cầu sử dụng lại cần mảng linh hoạt hơn.

- ArrayList là một cấu trúc dữ liệu có thể được kéo dài để chứa thêm thành phần bên trong và thu hẹp kích thước khi các thành phần bị loại bỏ.

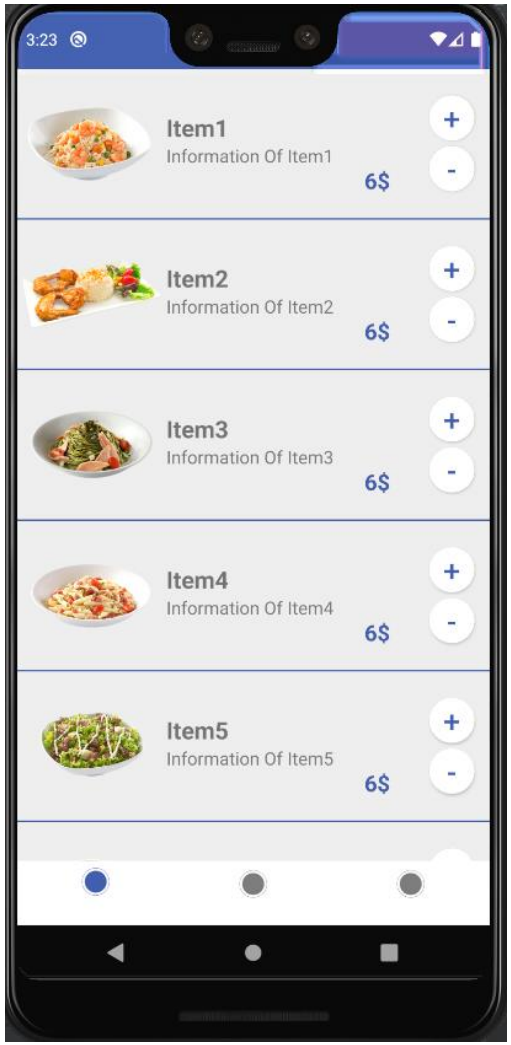
Phương thức của lớp ArrayList

Phương thức	Mô tả
boolean add(Object o)	Nó được sử dụng để nối thêm phần tử được chỉ định vào cuối một danh sách.
void add(int index, Object element)	Nó được sử dụng để chèn phần tử element tại vị trí index vào danh sách.
void remove(int pos)	Nó được sử dụng để xóa phần tử tại vị trí pos từ danh sách này
void clear()	Nó được sử dụng để xóa tất cả các phần tử từ danh sách này.
int size()	Đêm số phần tử trong mảng

Listview

- ListView là một view group, hiển thị các thành phần (elements) theo một danh sách, có thể cuộn được theo chiều thẳng đứng. ListView là một view quan trọng, nó được sử dụng rộng rãi trong các ứng dụng Android. Một ví dụ đơn giản của ListView là danh bạ liên lạc của bạn, nơi bạn có một danh sách các địa chỉ liên lạc của bạn hiển thị trong một ListView.
- Để tạo ListView hiển thị thông tin cần có 3 thành phần chính bao gồm :
  - ListView
  - ListItem
  - Adapter

Listview


3:23
Item1
Information Of Item1
6$
Item2
Information Of Item2
6$
Item3
Information Of Item3
6$
Item4
Information Of Item4
6$
Item5
Information Of Item5
6$

Listview

- android:id: Là thuộc tính định danh của ListView
- android:divider: Thuộc tính này có thể là một image hay màu dùng để phân chia giữa các dòng trong ListView.
- android:dividerHeight: Thuộc tính này xác định chiều cao của thuộc tính android:divider.
- android:listSelector: Thuộc tính này thường được sử dụng để thiết lập màu hoặc hình dòng được chọn trong listView. Thường nó sử dụng màu cam hoặc màu xanh dương, nhưng chúng ta cũng có thể thiết lập lại màu hoặc image cho ListView

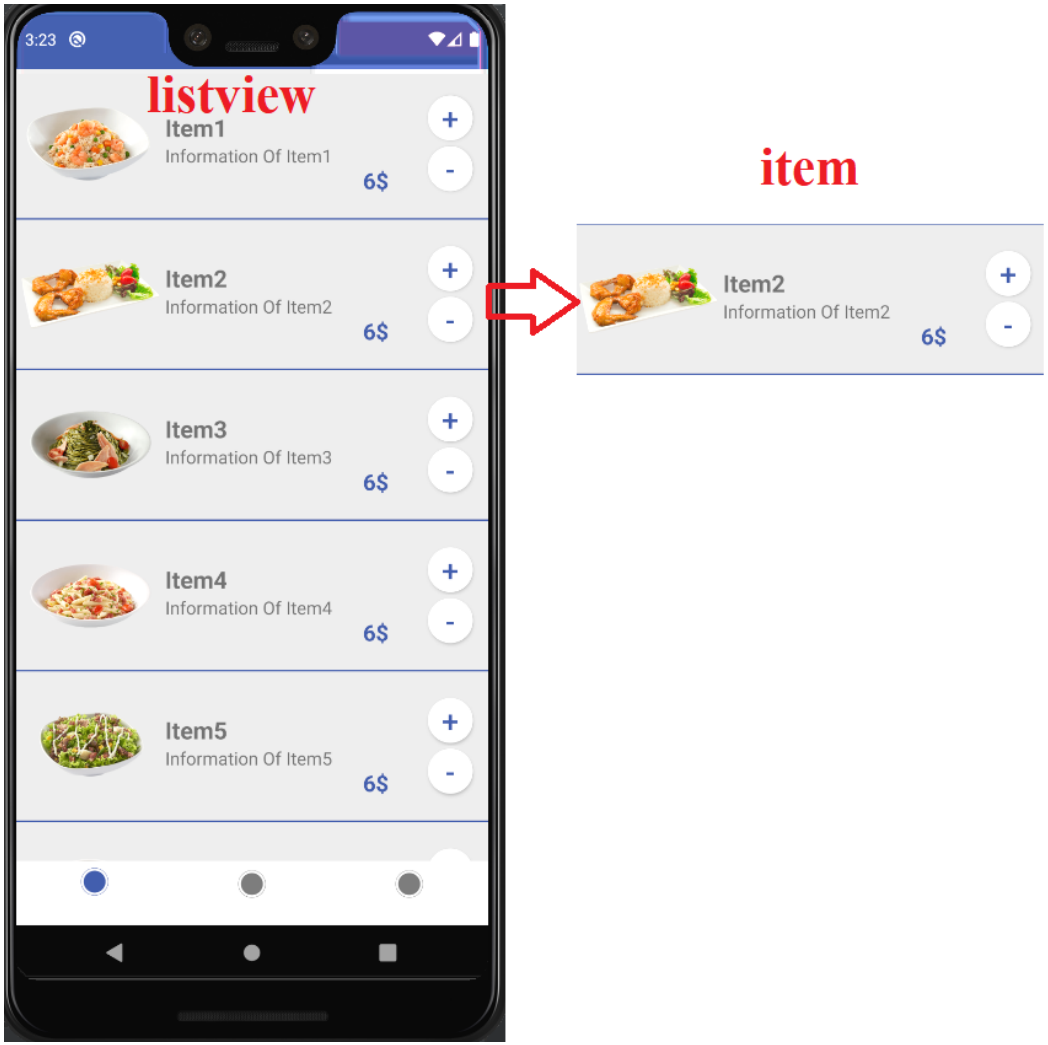
ListView

- setOnItemClickListener: Sự kiện này xảy ra khi người dùng click lên ListView
- setOnItemLongClickListener: Sự kiện được gắn cho ListView Item, khi nhấn lâu từ 2.5 tới 3 giây thì sự kiện này sẽ sảy ra

ListItem

- Một ListView được tạo từ một danh sách các ListItem. ListItem là một dòng (row) riêng lẻ trong listview nơi mà dữ liệu sẽ được hiển thị. Bất kỳ dữ liệu nào trong listview chỉ được hiển thị thông qua listItem. Có thể coi listview như là một nhóm cuộn của các ListItem.

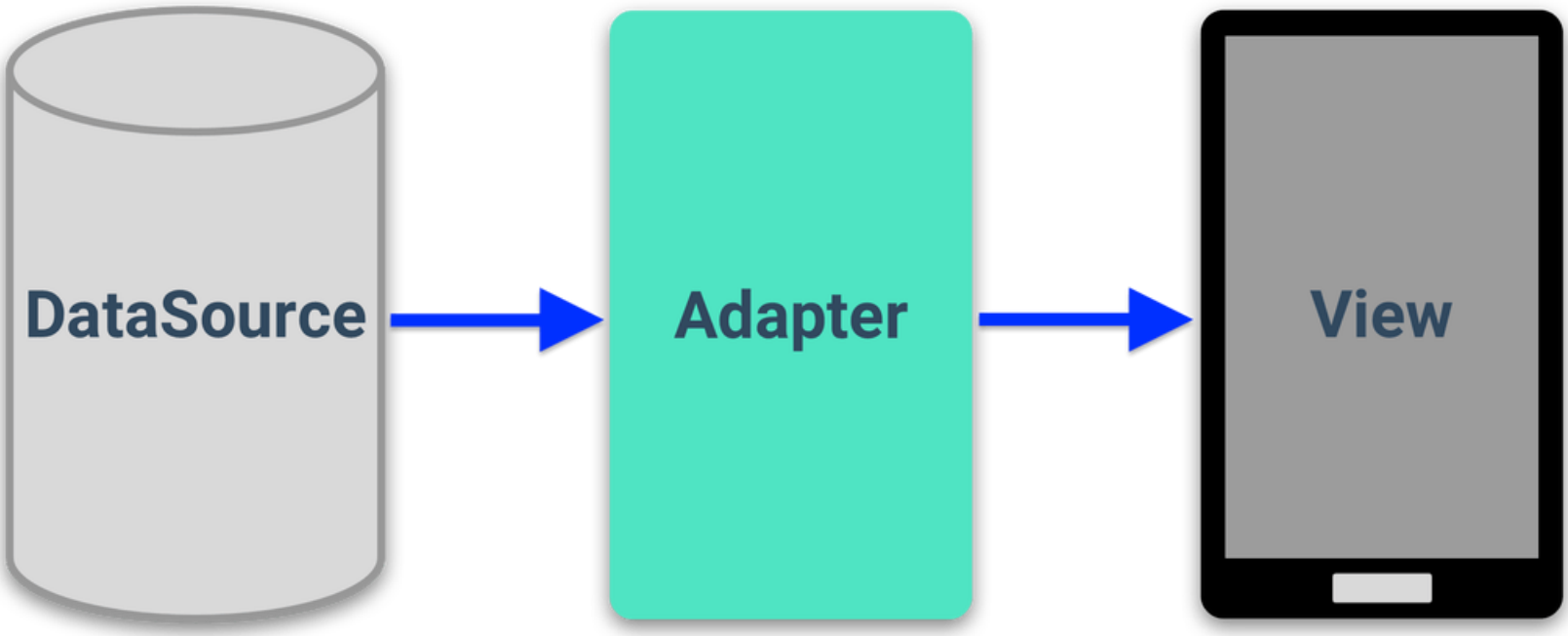
ListItem


3:23
listview
Item1
Information Of Item1
6$
Item2
Information Of Item2
6$
Item3
Information Of Item3
6$
Item4
Information Of Item4
6$
Item5
Information Of Item5
6$
item
Item2
Information Of Item2
6$

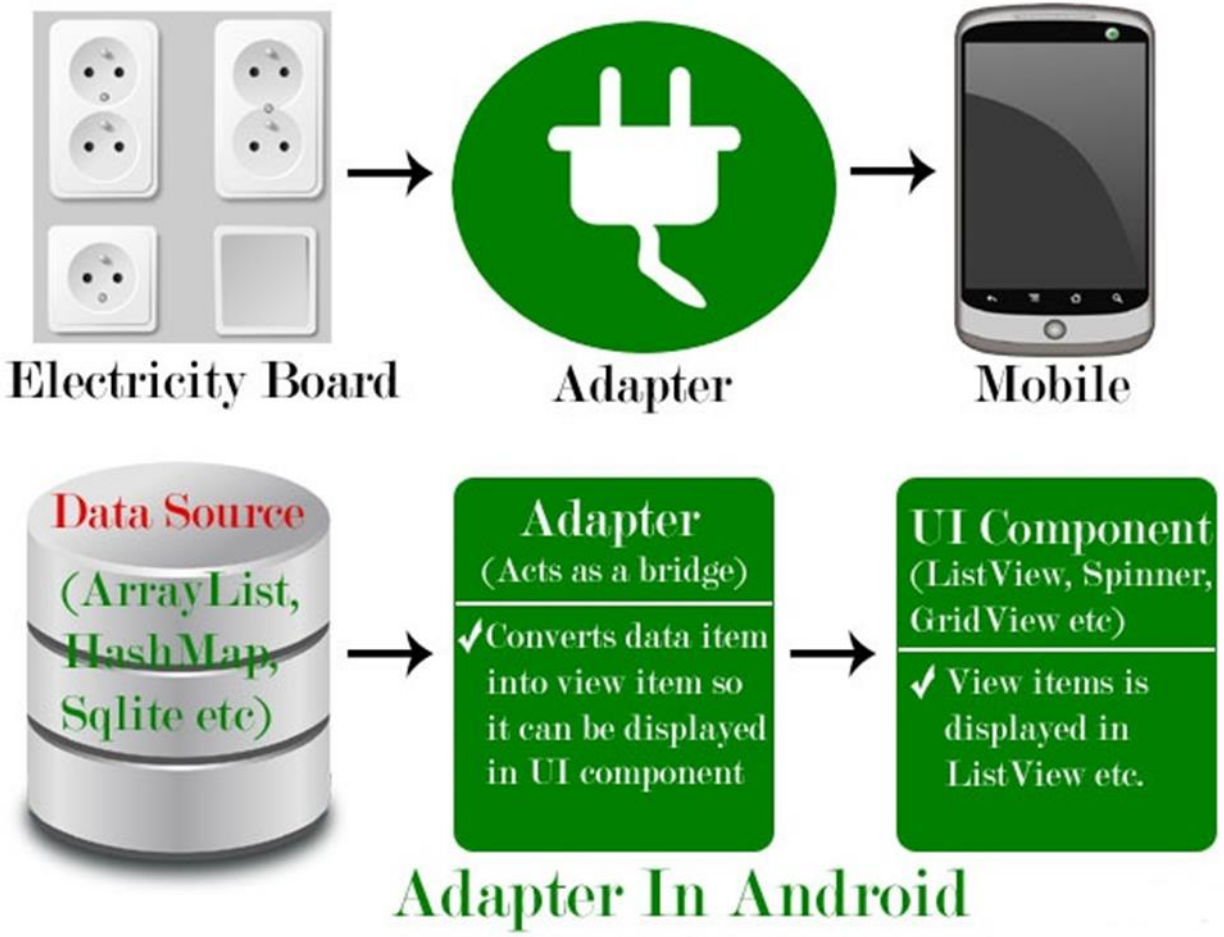
Adapter

- Một Adapter là một đối tượng của một lớp cài đặt giao diện Adapter. Nó đóng vai trò như là một liên kết giữa một tập hợp dữ liệu và một Adapter View, một đối tượng của một lớp thừa kế lớp trừu tượng AdapterView. Tập hợp dữ liệu có thể là bất cứ điều gì mà trình bày dữ liệu một cách có cấu trúc. Mảng, các đối tượng List và các đối tượng Cursor thường sử dụng bộ dữ liệu.
- Một Adapter có trách nhiệm lấy dữ liệu từ bộ dữ liệu và tạo ra các đối tượng View dựa trên dữ liệu đó. Các đối tượng View được tạo ra sau đó được sử dụng để gắn lên bất kỳ Adapter View mà ràng buộc với Adapter.

Adapter



Adapter


Electricity Board
Adapter
Mobile
Data Source
(ArrayList, HashMap, Sqlite etc)
Adapter
(Acts as a bridge)
✓Converts data item into view item so it can be displayed in UI component
UI Component
(ListView, Spinner, GridView etc)
✓ View items is displayed in ListView etc.
Adapter In Android

Adapter View làm việc như thế nào

- Adapter View có thể hiển thị các bộ dữ liệu lớn rất hiệu quả. Ví dụ, ListView có thể hiển thị hàng triệu phần tử mà không có bất cứ độ trễ đáng kể nào trong khi vẫn sử dụng bộ nhớ và CPU rất thấp. Chúng có thể làm điều đó như thế nào? Các Adapter View khác nhau tuân theo những chiến lược khác nhau.
- Chúng chỉ kết xuất những đối tượng View mà đã trên màn hình hoặc nó đang di chuyển vào màn hình. Bằng cách này, bộ nhớ tiêu thụ bởi một Adapter View có thể được cố định và độc lập với kích thước của tập dữ liệu.
- Chúng cũng cho phép các nhà phát triển giảm thiểu công sức cho các hoạt động inflate layout và tái sử dụng các đối tượng View sẵn có đã di chuyển khỏi màn. Điều này sẽ giúp tiêu thụ CPU thấp.

ArrayAdapter

- Một ListAdapter có thể quản lí một ListView chứa danh sách các phần tử có kiểu bất kì.
- Cú pháp sử dụng ArrayAdapter:

  ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)

  1. context:

  Tham số đầu tiên là context dùng để tham chiếu đến lớp hiện tại.Thông thường sử dụng từ khóa this. Ngoài ra, chúng ta cũng có thể sử dụng getApplicationContext(), getActivity() để thay thế từ khóa this. getApplicationContext(), được sử dụng trong Activity, còn getActivity() được sử dụng trong Frament.

ArrayAdapter

2. resource:

Tham số thứ 2 là tài nguyên (resource) id được sử dụng thiết lập các danh sách trong Layout. Trong Layout có thể TextView, ImageView, Button .v.v

3. textViewResourceId:

Tham số thứ 3 dùng để hiển thị dữ liệu dạng TextView (Mặc định không khai báo tham số này).

4. objects:

- Tham số thứ 4 là nguồn dữ liệu kiểu mảng đối tượng, Chúng ta có thể lấy, thiết lập dữ liệu thông qua tham số này.

```
String animalList[] =
{"Lion","Tiger","Monkey","Elephant","Dog","Cat","Camel"};
ArrayAdapter arrayAdapter = new ArrayAdapter(this,
R.layout.list_view_items, R.id.textView, animalList);
```

Phương thức ArrayAdapter

- getCount(): Phương thức này trả về số dòng của List
- getView(int i, View view, ViewGroup viewGroup): Phương thức này được gọi tự động các mục danh sách để sẵn sàng hiển thị. Trong phương thức chúng ta thiết lập layout cho danh sách các mục bằng cách sử dụng lớp LayoutInflater và sau đó thêm dữ liệu cho các view: ImageView, TextView .v.v.