Lập trình Mobile

Tuần 3

Giảng viên: Trần Đức Minh

Nội dung bài giảng

- Lớp
- Kế thừa
- Interface

Lớp

- Khai báo lớp

class <tên lớp> {

<các thuộc tính>

<các setters và các gettters>

<các cấu tử>

<các phương thức>

}

<các thuộc tính>: Dữ liệu liên quan đến đối tượng.

<các setters và các getters>: Đặt và lấy giá trị của các thuộc tính.

<các cấu tử>: Cấp phát bộ nhớ cho các đối tượng của lớp.

<các phương thức>: Các hành động mà đối tượng thực hiện.

Lớp

- Ví dụ:

```
class SinhVien {
    String ten = “Nguyen Lan Anh”;
    int tuoi = 19;

    void showInfo() {
        print(ten + “ co tuoi la ${tuoi}”);
    }
}
```

Lớp

- Khởi tạo đối tượng của một lớp
  - Cú pháp:

    var <tên đối tượng> = **new** <tên lớp>(<các tham số>);

    var <tên đối tượng> = <tên lớp>(<các tham số>);
  - Ví dụ:

    var sv = new SinhVien();
- Truy cập thuộc tính và phương thức thông qua đối tượng
  - Cú pháp:
    - obj.<tên trường>
    - obj.<tên phương thức>
  - Ví dụ:
    - sv.tuoi
    - sv.showInfo()

Lớp

- Cấu tử
  - Được dùng để khởi tạo giá trị cho các thuộc tính.
  - Nếu không khai báo cấu tử thì một cấu tử mặc định (không tham số) sẽ tự động được thêm vào lớp.
  - Cú pháp:

```
<tên lớp>(<các tham số>) {
    <Các lệnh phần thân cấu tử>
}
```

  - Ví dụ:

```
SinhVien(String name) {
    ten = name;
}
```

Lớp

- Cấu tử được đặt tên
  - Cho phép lớp có thể định nghĩa nhiều cấu tử.
  - Cú pháp:
    - <tên lớp>.<tên cấu tử>(<các tham số>);
  - Ví dụ:

```
SinhVien.khoiTao(String name, int age) {
    ten = name;
    tuoi = age;
}
```

  - Gọi cấu tử đặt tên:

```
var sv = new SinhVien.khoiTao(“Hoang”, 18);
```

Lớp

- Từ khóa **this**
  - Ám chỉ đối tượng hiện tại.
  - Ví dụ:

```
SinhVien.khoiTao(String ten, int tuoi) {
    this.ten = ten;
    this.tuoi = tuoi;
}
```

Lớp

- Các setters và các getters dùng để đặt và lấy giá trị thuộc tính
  - Cú pháp:

    [void] **set** <định danh>(<kiểu dữ liệu> <tên đối số>) {
        <phần thân setter>
    }

    <kiểu dữ liệu trả về> **get** <định danh> {
        return <giá trị trả về>
    }
  - Ví dụ:

    set name (String ten) {
        this.ten = ten;
    }

    String get name {
        return ten;
    }

Lớp

- Các setters và các getters dùng để đặt và lấy giá trị thuộc tính
  - Ví dụ: Gọi đến các setters và gettters
    - sv.name = “Lam”;
    - print(sv.name);

Lớp

- Truyền giá trị cho phương thức thông qua tên đối số.
- Cú pháp:
  - Khai báo phương thức

    <giá trị trả về> <tên phương thức>({các đối số}) {

    ....

    }
  - Gọi phương thức:

    <tên phương thức>(<đối số 1> : <giá trị 1>, <đối số 2> : <giá trị 2>, …)
- Ví dụ:

```
SinhVien({String? ten, int? tuoi}) {
    this.ten = ten;
    this.tuoi = tuoi;
}

SinhVien sv = SinhVien(tuoi : 8, ten : “Lan”);
```

Lớp

- Truyền giá trị mặc định cho đối số của phương thức.
  - Nếu ta không truyền giá trị cho đối số, đối số sẽ nhận ngay giá trị mặc định.
  - Ví dụ:

```
SinhVien({String? ten, int tuoi = 15}) {
    this.ten = ten;
    this.tuoi = tuoi;
}
SinhVien sv = SinhVien(ten : “Minh”);
```

- Từ khóa **required**
  - Dành cho đối số bắt buộc phải truyền giá trị khi gọi phương thức
  - Ví dụ:

```
SinhVien({required String ten, required int tuoi}) {
    this.ten = ten;
    this.tuoi = tuoi;
}
SinhVien sv = SinhVien(ten : “Minh”, tuoi : 15);
```

Lớp

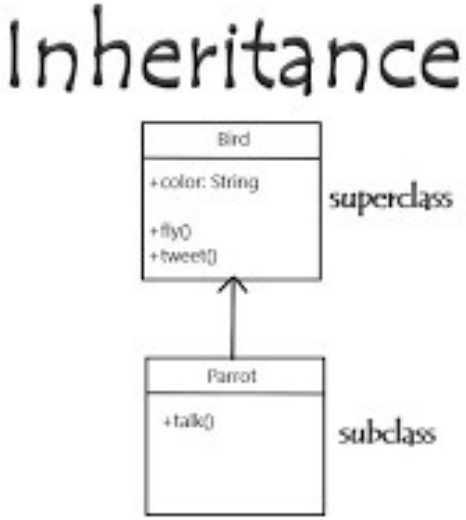
- Cấu tử rút gọn
  - Ví dụ:

```
SinhVien({this.ten, this.tuoi});

SinhVien sv = SinhVien(ten : “Lan”, tuoi : 18);
```

Kế thừa

- Trong Dart, kế thừa là việc tạo ra các lớp mới từ lớp hiện có.
  - Lớp tạo mới được gọi là lớp dẫn xuất hay lớp con.
  - Lớp hiện có được gọi là lớp cơ sở hay lớp cha.



Kế thừa

- Cú pháp

  class <tên lớp con> **extends** <tên lớp cha> {

  ...

  }

- Ví dụ:

```
class Shape {
    void cal_area() {
        print("calling calc area defined in the Shape class");
     }
}

class Circle extends Shape {
...
}
```

Kế thừa

- Trong Dart, chỉ có đơn kế thừa, không có đa kế thừa.
- Kế thừa đa mức (multi-level). Ví dụ:

```
class Circle extends Shape {
...
}

class Sphere extends Circle {
...
}
```

Kế thừa

- Nạp chồng phương thức
  - Định nghĩa lại phương thức của lớp cha ở lớp con.
- Ví dụ:

```
void main() {
   Child c = new Child();
   c.m1(12);
}

class Parent {
   void m1(int a){ print("value of a ${a}"); }
}

class Child extends Parent {
   @override
   void m1(int b) { print("value of b ${b}"); }
}
```

Từ khóa static

- Các thuộc tính và phương thức có dạng **static** được truy cập thông qua tên của lớp.
- Ví dụ:

```
class StaticMem {
   static var num;
   static disp() {
      print("The value of num is ${StaticMem.num}")  ;
   }
}

void main() {
   StaticMem.num = 12;
   StaticMem.disp();
}
```

Từ khóa super

- Từ khóa **super** được dùng để ám chỉ cha của lớp hiện tại.
- Ví dụ:

```
void main() {
  Child c = new Child();
  c.m1(12);
}

class Parent {
  String msg = "message variable from the parent class";
  void m1(int a){ print("value of a ${a}");}
}

class Child extends Parent {
  @override
  void m1(int b) {
    print("value of b ${b}");
    super.m1(13);
    print("${super.msg}")   ;
  }
}
```

Toán tử cascade

- Toán tử **cascade** (..) được dùng để đưa ra một dãy các lời gọi hàm thông qua chỉ một đối tượng.
- Ví dụ:

```
class Student {
  void test_method() {
    print("This is a  test method");
  }

  void test_method1() {
    print("This is a  test method1");
  }
}

void main() {
  var sv = new Student();
  sv
  ..test_method()
  ..test_method1();
}
```

Lớp trừu tượng

- Lớp trừu tượng là lớp có ít nhất một phương thức trừu tượng (phương thức không thực hiện - không code).
- Lớp trừu tượng có thể chứa cả các phương thức thông thường.
- Cú pháp:

```
abstract class ClassName {
…
}
```

- Ví dụ:

```
abstract class NhanSu {
    double tinhLuong();
}
```

Sử dụng lớp trừu tượng

- Lớp trừu tượng không được sử dụng để tạo đối tượng mà chỉ được sử dụng để làm lớp cơ sở cho các lớp khác.
- Khi lớp dẫn xuất kế thừa từ lớp cơ sở trừu tượng, nó cần phải nạp chồng lên các phương thức trừu tượng.
- Ví dụ:

```
class GiamDoc extends NhanSu {
    double tinhLuong() {
        return 1000;
    }
}
```

Interface

- Trong Dart, các lớp (class) cũng có thể được sử dụng như interface.
- Một lớp khi thực hiện (implement) interface đồng nghĩa với việc nó bắt buộc phải viết code cho các phương thức thuộc về interface.
- Một lớp có thể thực hiện một lúc nhiều interface.
- Cú pháp:
  - class <tên lớp> implements <danh sách các interface>
- Ví dụ:

```
class KySu implement NhaNghienCuu, NhanVien {
...
}
```

Interface

- Ví dụ:

```
class NhaNghienCuu {
  void thongTinNhaNghienCuu() {}
}

class NhanVien {
  void thongTinNhanVien() {}
}
```

Interface

- Ví dụ:

```
class KySu implements NhaNghienCuu, NhanVien {
  var name;
  var so_cong_trinh;
  var so_gio_lam;

  KySu({this.name, this.so_cong_trinh, this.so_gio_lam});

  void thongTinNhaNghienCuu() {
    print("$name co so cong trinh nghien cuu la $so_cong_trinh");
  }

  void thongTinNhanVien() {
    print("$name co so gio lam la $so_gio_lam");
  }
}
```

Interface

- Ví dụ:

```
void main() {
  NhaNghienCuu nnc = KySu(name : "Quang",
so_cong_trinh : 9 , so_gio_lam: 165);

  nnc.thongTinNhaNghienCuu();
  nnc.thongTinNhaVien();   // Lỗi
}
```

Generics với hàm

- Generics là cách thức lập trình tổng quát cho phép một lớp hoạt động với nhiều kiểu dữ liệu khác nhau.
- Ví dụ:

```
int countItem<T>(List<T> list, T itemToCount) {
  int count = 0;

  for(T item in list)
    if (item == itemToCount)
      count++;

  return count;
}

void main() {
  List<int> myList = [3, 6, 8, 9, 6, 3, 5, 7];

  print(countItem<int>(myList, 7));
}
```

Lớp Generic

- Ví dụ:

```
class GenericClass<T> {
  var obj;

  void set object(T obj) {
    this.obj = obj;
  }

  T get object {
    return obj;
  }
}

void main() {
  var gc = new GenericClass<String>();

  gc.object = "This is a generic class.";

  print(gc.object);
}
```

Thư viện

- Thư viện chứa các hàm, thủ tục, lớp được sử dụng thường xuyên.
- Cú pháp gọi thư viện:
  - import ‘<URI>’ [show | hide] <các lớp>
- Ví dụ:
  - import 'dart:io'
  - import 'package:lib1/libfile.dart'
  - import 'package:lib1/lib1.dart' show foo, bar;

    // Import chỉ foo và bar.
  - import 'package: mylib/mylib.dart' hide foo;

    // Import tất cả, ngoại trừ foo

Thư viện

- Một số thư viện thường dùng trong Dart

dart:io	Làm việc với file, HTTP, socket và các ứng dụng Vào/Ra khác.
dart:core	Các kiểu dữ liệu, collections và các chức năng lõi. Thư viện này được import tự động.
dart:math	Chứa các hàm toán học.
dart:convert	Mã hóa, giải mã, chuyển đổi giữa các loại dữ liệu khác nhau.

Hết Tuần 3

Cảm ơn các bạn đã chú ý lắng nghe !!!