상속과 추상형

Dart

상속 알아보기

상속과 오버라이딩

- 상속inheritance은 클래스를 재활용하는 객체지향 프로그래밍의 핵심 기능
- 다트에서 클래스를 선언할 때 어떤 클래스를 상속받으려면 extends 예약어를 사용

• 함수에서 널 불허 지역 변수 초기화

```
class SuperClass {
  int myData = 10;
  void myFun() {
    print('Super..myFun()...');
  }
}

class SubClass extends SuperClass {
}

main(List<String> args) {
  var obj = SubClass();
  obj.myFun();
  print('obj.data : ${obj.myData}');
}
```

▶ 실행 결과

```
Super..myFun()...
obj.data : 10
```

상속 알아보기

상속과 오버라이딩

- 오버라이딩

• 오버라이딩

```
class SuperClass {
  int myData = 10;
  void myFun() {
    print('Super..myFun()...');
  }
}

class SubClass extends SuperClass {
  int myData = 20;
  void myFun() {
    print('Sub... myFun()...');
  }
}

main(List<String> args) {
  var obj = SubClass();
  obj.myFun();
  print('obj.data : ${obj.myData}');
}
```

▶ 실행 결과

```
Sub... myFun()...
obj.data : 20
```

상속 알아보기

상속과 오버라이딩

- 자식 클래스에서 부모 클래스에 선언된 멤버를 재정의할 때 부모 클래스에 선언된 똑같은 이름의 멤버를 이용하고 싶다면 다음처럼 super라는 예약어로 접근

• 부모 클래스의 멤버에 접근하기

```
class SuperClass {
  int myData = 10;
  void myFun() {
    print('Super..myFun()...');
  }
}
class SubClass extends SuperClass {
  int myData = 20;
  void myFun() {
    super.myFun();
    print('Sub... myFun()..myData : $myData, super.myData : ${super.myData}');
  }
}

main(List<String> args) {
  var obj = SubClass();
  obj.myFun();
}
```

▶ 실행 결과

```
Super..myFun()...
Sub... myFun()..myData : 20, super.myData : 10
```

상속 알아보기

부모 생성자 호출하기

- 자식 클래스의 객체를 생성할 때 자신의 생성자가 호출되는데, 이때 부모 클래스의 생성자도 반드시 호출되게 해줘야 합니다.

• 자식 클래스의 생성자 호출(부모 생성자는 자동 호출됨)

```
class SuperClass {
  SuperClass() {}
}
class SubClass extends SuperClass {
  SubClass() {}
}
main() {
  var obj = SubClass();
}
```

상속 알아보기

부모 생성자 호출하기

- 자식 클래스의 생성자에서 부모 클래스의 생성자를 호출하려면 super() 문을 사용

• 부모 생성자 호출

```
class SuperClass {
  SuperClass() {}
}
class SubClass extends SuperClass {
  SubClass() : super() {}
}
```

• 부모 생성자 호출의 잘못된 예

```
class SuperClass {
  SuperClass(int arg) {}
  SuperClass.first() {}
}
class SubClass extends SuperClass {
  SubClass() : super() {}   // 오류
}
```

• 부모 생성자 호출의 올바른 예

```
class SuperClass {
  SuperClass(int arg) {}
  SuperClass.first() {}
}
class SubClass extends SuperClass {
  SubClass() : super(10) {}          // 성공
  SubClass.name() : super.first() {}  // 성공
}
```

상속 알아보기

부모 클래스 초기화

- 부모 클래스의 생성자를 호출하는 super() 구문에 매개변숫값을 전달

• 부모 클래스의 멤버 변수 초기화 1

```
class SuperClass {
  String name;
  int age;
  SuperClass(this.name, this.age) {}
}
class SubClass extends SuperClass {
  SubClass(String name, int age) : super(name, age) {}   // 부모 클래스 멤버 초기화
}
main() {
  var obj = SubClass('kkang', 10);
  print('${obj.name}, ${obj.age}');
}
```

▶ 실행 결과

```
kkang, 10
```

상속 알아보기

부모 클래스 초기화

- 생성자의 매개변수에 super로 부모 클래스의 멤버를 작성하면 해당 값으로 부모 클래스의 생성자가 호출돼 멤버 변수가 초기화

• 부모 클래스의 멤버 변수 초기화 2

```
class SuperClass {
  String name;
  int age;
  SuperClass(this.name, this.age) {}
}
class SubClass extends SuperClass {
  SubClass(super.name, super.age);
}
main() {
  var obj = SubClass('kkang', 10);
  print('${obj.name}, ${obj.age}');
}
```

▶ 실행 결과

```
kkang, 10
```

추상 클래스와 인터페이스

추상 클래스 알아보기

- 추상 클래스는 추상 함수를 제공하여 상속받는 클래스에서 반드시 재정의해서 사용하도록 강제하는 방법
- 추상 함수는 실행문이 작성된 본문이 없는 함수를 의미

• 추상 함수 선언

```
class User {
   void some();   // 오류
}
```

• 추상 클래스 선언

```
abstract class User {
   void some();   // 성공
}
```

추상 클래스와 인터페이스

추상 클래스 알아보기

- 추상 클래스를 상속받은 자식 클래스에서 추상 함수를 재정의해 줘야 합니다.

• 추상 함수 재정의

```
abstract class User {
  void some();
}
class Customer extends User {
  @override
  void some() {}
}
```

추상 클래스와 인터페이스

인터페이스 알아보기

- 다트에서는 implements만 지원하고 interface는 지원하지 않습니다.
- 인터페이스를 명시적으로 선언하지 않아도 다른 클래스를 도구 삼아 구현하는 방법을 제공
- 암시적 인터페이스란 클래스 자체가 인터페이스라는 의미
- 클래스를 implements로 선언하면 다른 클래스를 인터페이스로서 활용할 수 있다는 의미

• 일반 클래스

```
class User {
  int no;
  String name;

  User(this.no, this.name);
  String greet(String who) => 'Hello, $who. I am $name. no is $no';
}
```

• User의 자식 클래스 선언

```
class MySubClass extends User {
  MySubClass(super.name, super.no);
}
```

추상 클래스와 인터페이스

인터페이스 알아보기

- User 클래스는 암시적 인터페이스가 되고, MyClass는 User를 새로 구현한 클래스

• 인터페이스 구현 클래스 선언

```
class MyClass implements User {   // 오류
}
```

- 클래스에 implements를 추가해 어떤 클래스를 구현하는 클래스는 대상 클래스에 선언된 모든 멤버를 재정의

• 인터페이스의 모든 멤버 재정의

```
class MyClass implements User {
  int no = 10;

  String name = 'kim';
  @override
  String greet(String who) {
    return 'hello';
  }
}
```

추상 클래스와 인터페이스

인터페이스 알아보기

- 하나의 클래스에 여러 인터페이스를 지정해서 선언

• 한 클래스에 여러 인터페이스 지정

```
class MyClass implements User, MyInterface {
}
```

- 구현 클래스의 객체는 다음처럼 인터페이스 타입으로 선언

• 인터페이스 타입 객체 선언

```
main() {
  User user = MyClass();
  print('${user.greet('lee')}');
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare