[Flutter]dart 中 extends、implements、with 的用法与区别

dafasoft

概述

- 继承（关键字 extends)
- 混入 mixins (关键字 ~~width~~) with mixin 混合
- 接口实现（关键字 implements） implements 实现

这三种关系有可能同时存在，但是有先后顺序

优先级：

extends -> ~~width~~ -> implements with

解析

继承：

dart 中的继承规则：

- 子类使用 extends 关键词来继承父类
- 子类会继承父类里面可见的属性和方法 但是不会继承构造函数
- 子类能复写父类的方法 getter 和 setter
- 子类重写超类的方法，要用 @override

- 子类调用超类的方法，要用super
- 子类可以继承父类的非私有变量

示例：

```
class Person {
  //公有变量
  String name;
  num age;
  //私有变量
  String _gender = '男';
  //类名构造函数
  Person(this.name, this.age);

  work() {
    print("${this.name}在学习...");
  }
}

class Student extends Person {
  Student(String name, num age) : super(name, age);

  run() {
    print('run');
    work();
  }
}

main() {
  Student student = new Student("张三", 16);
  student.run();
}
```

运行结果：

```
run
张三在学习...
```

覆写方法work:

```
class Person {
  //公有变量
  String name;
  num age;
  //私有变量
  String _gender = '男';
  //类名构造函数
  Person(this.name, this.age);

  work() {
    print("${this.name}在学习...");
  }
}

class Student extends Person {
  String school = "清华大学";

  Student(String name, num age) : super(name, age);

  run() {
    print('run');
    work();
  }

  @override
  work() {
    print("${this.name} 在 ${school} 学习");
  }
```

```
}

main() {
  Student student = new Student("张三", 16);
  student.run();
}
```

整体和java的继承很类似

混合 mixins（with）

这里是要着重介绍的，因为我是一个Android程序员，第一次接触这个关键字

- Mixin 是复用类代码的一种途径， 复用的类可以在不同层级，之间可以不存在继承关系。
- 通过 with 后面跟一个或多个混入的名称，来 使用 Mixin
- 通过创建一个继承自 Object 且没有构造函数的类，来实现 一个 Mixin 。 如果 Mixin 不希望作为常规类被使用，使用关键字 mixin 替换 class 。 例如：

```
mixin Family {
  String address;
  String house;

  void setFamilyInfo(String address, String house) {
    this.address = address;
    this.house = house;
```

```
  }

  getAddress() {
    return address;
  }
}
```

- 作为 mixins 的类只能继承自 Object，不能继承其他类
- 作为 mixins 的类不能有构造函数
- 一个类可以 mixins 多个 mixins 类
- mixins 绝不是继承，也不是接口，而是一种全新的特性

使用：

```
class Person {
  //公有变量
  String name;
  num age;
  //私有变量
  String _gender = '男';
  //类名构造函数
  Person(this.name, this.age);

  work() {
    print("${this.name}在学习...");
  }
}

class Student extends Person with Family {
  String school = "清华大学";

  Student(String name, num age) : super(name, age);

  run() {
```

```
    setFamilyInfo("北京", "明天第一城");
    print('run');
    work();
  }

  @override
  work() {
    String add = getAddress();
    print("${this.name}的家在${add} 在 ${school} 学习");
  }
}

main() {
  Student student = new Student("张三", 16);
  student.run();
}
```

打印结果：

```
run
张三的家在北京 在 清华大学 学习
```

其他注意的点：

mixins有优先级顺序，我们通过一些例子看一下：

mixin

```
class A {
  run() {
    print("run A");
  }
}
```

mixin

```
class B {
  run() {
```

```
    print("run B");
  }
}

class C {
  run() {
    print("run C");
  }
}

class Test extends C with A, B {
}

main() {
  Test test = new Test();
  test.run();
}
```

打印结果：

```
run B
```

如果把 with B 去掉 只留下 with A,再运行一次：

```
class Test extends C with A {}

main() {
  Test test = new Test();
  test.run();
}
```

打印结果：

```
run A
```

如果 Test类中覆写了父类C的run方法：

```
class Test extends C with A {
  @override
  run() {
    print("run Test");
  }
}

main() {
  Test test = new Test();
  test.run();
}
```

结论：输出为：run Test

如果继承类、混合类中有相同的方法或变量，混合类会覆盖继承类的方法或变量，后混合的会覆盖先混合的方法或变量。

这个过程叫做mixins的线性化，具体过程：

- 如果当前使用类重写了该方法，就会调用当前类中的方法。
- 如果当前使用类没有重写了该方法，则会调用距离with关键字最远类中的方法。

with on 的用法

mixins不能继承于除Object之外的其他类，如果我们确实有继承的需要怎么办呢？ Dart引入了一个关键字on

具体使用方法：

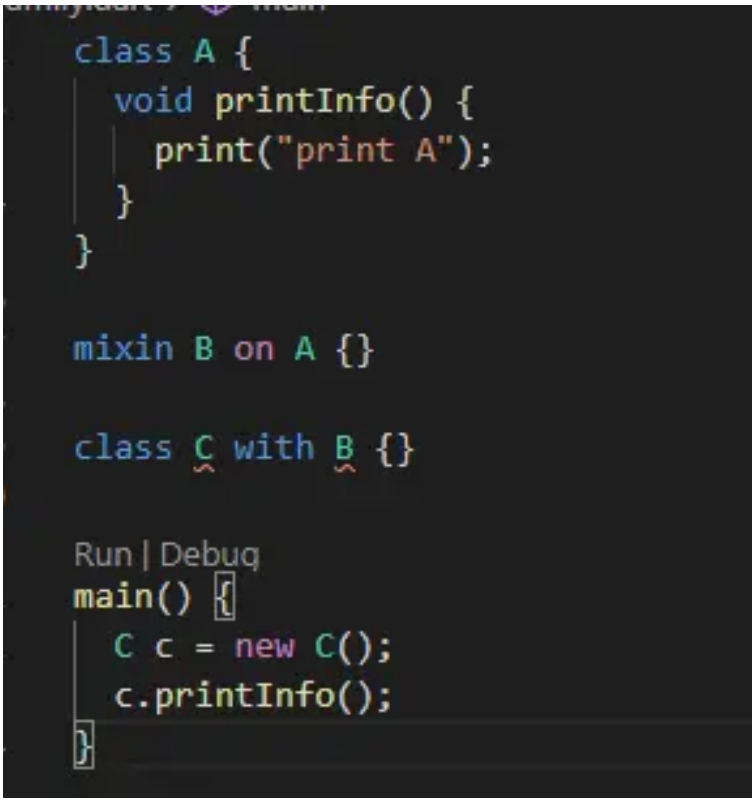
```
class A {
  void printInfo() {
    print("print A");
  }
}

mixin B on A {}
```

官网对它的描述：

> 指定只有某些类型可以使用的 Mixin - 比如， Mixin 可以调用 Mixin 自身没有定义的方法 - 使用 on 来指定可以使用 Mixin 的父类类型

如果我们用 on 关键字限定了 mixin ， 那么这个 mixin 只能适用于 on 后面限定的类的子类

举个例子：

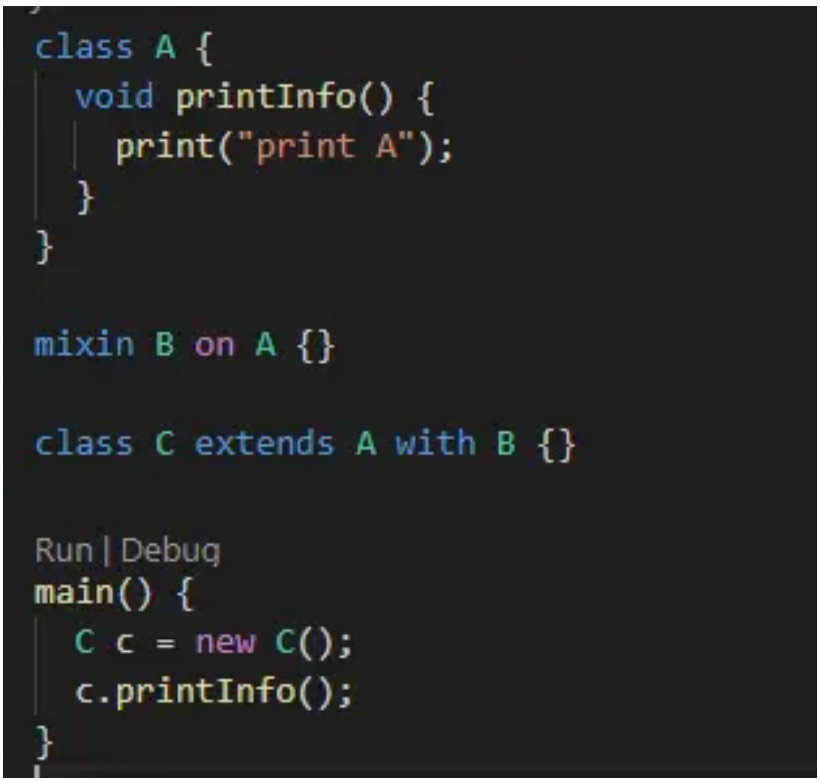


```dart
class A {
  void printInfo() {
    print("print A");
  }
}

mixin B on A {}

class C with B {}

Run | Debug
main() {
  C c = new C();
  c.printInfo();
}
```

这种情况C类是会报错的，报错信息：

```
'B' can't be mixed onto 'Object' because 'Object' doesn't implement 'A'.
```

需要将C类继承于A类, 如下：



```dart
class A {
  void printInfo() {
    print("print A");
  }
}

mixin B on A {}

class C extends A with B {}

Run | Debug
main() {
  C c = new C();
  c.printInfo();
}
```

报错消失

接口实现 (implements)

Dart是没有 interface的，但是Dart中的每个类都是一个隐式的接口，这个接口包含类里的所有成员变量，以及定义的方法。如果有一个类 A,你想让类B拥有A的API，但又不想拥有A里的实现，那么你就应该把A当做接口，类B implements 类A.

所以在Dart中:class 就是 interface

- 当class被当做interface用时，class中的方法就是接口

  当class被当做interface时，类前面要加abstract

  implements 实现

的方法，需要在子类里重新实现，在子类实现的时候要加 @override

- 当 class 被当做 interface 用时，class 中的成员变量也需要在子类里重新实现。在成员变量前加 @override

```
/*
Dart中一个类实现多个接口:
*/

abstract class A{
  String name;
  printA();
}

abstract class B{
  printB();
}

class C implements A,B{
  @override
  String name;
  @override
  printA() {
    print('printA');
  }
  @override
  printB() {
    // TODO: implement printB
    return null;
  }
}
```

```
void main(){
  C c=new C();
  c.printA();
}
```