PDP ACADEMY

Modul III. Lesson 3

Mixin

Nasibali Abdiyev. Flutter Development

Repeat the previous lesson

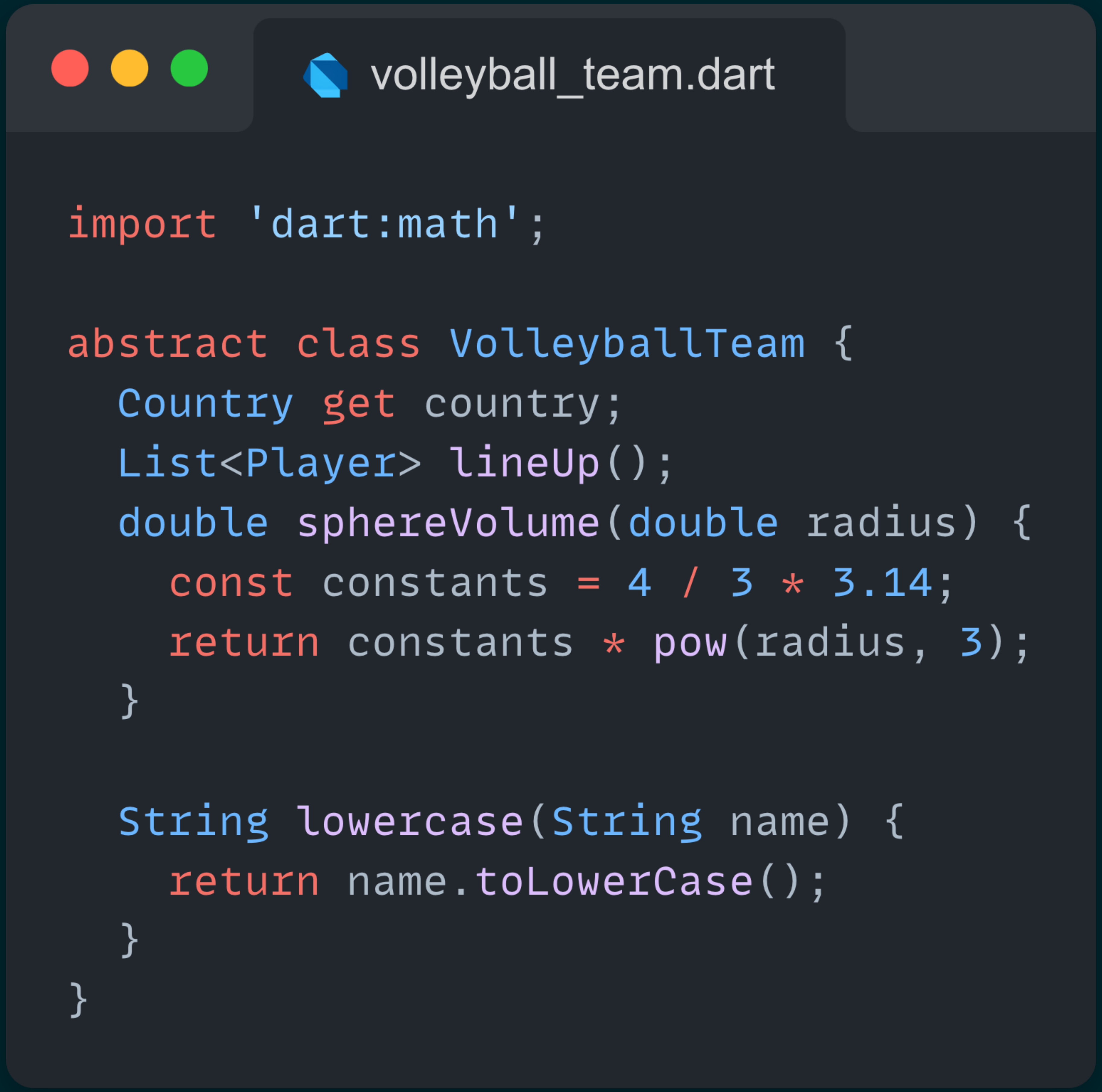
- Interface
- Business logic
- Abstract interface class
- Implements keywords

Plan

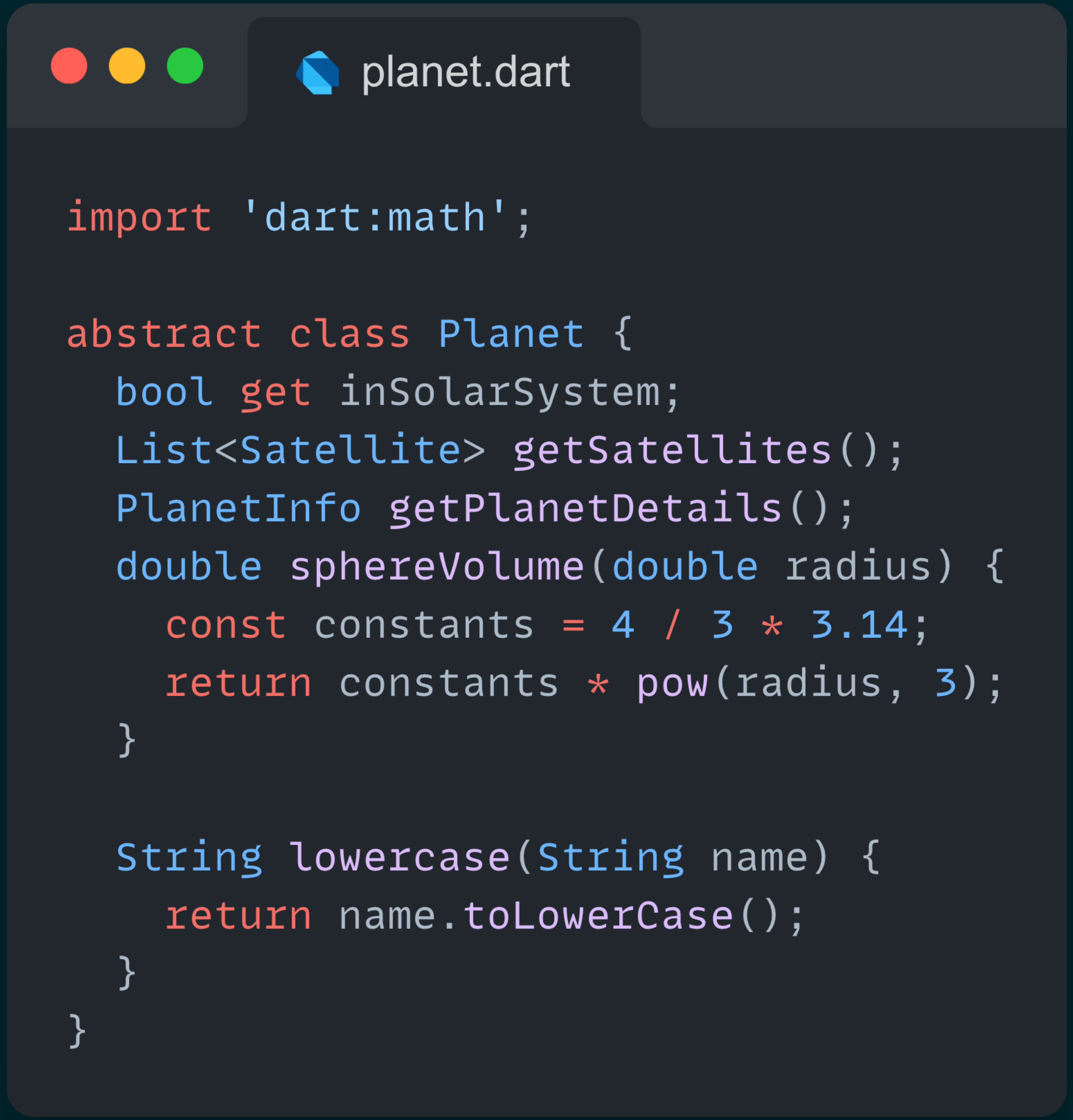
- Mixin
- Problem in Extends and Implements
- Mixin Sytax
- On keyword
- Rules
- When to use mixin?

Mixin

- Mixin - bu meros qo'llamay turib metodlarni qayta ishlatish uchun boshqa sinflar bilan "birlashtirilishi" mumkin bo'lgan sinfdir. Masalan, tasavvur qiling, sizda ikki alohida ierarxiyada bu ikki sinf bor:


volleyball_team.dart

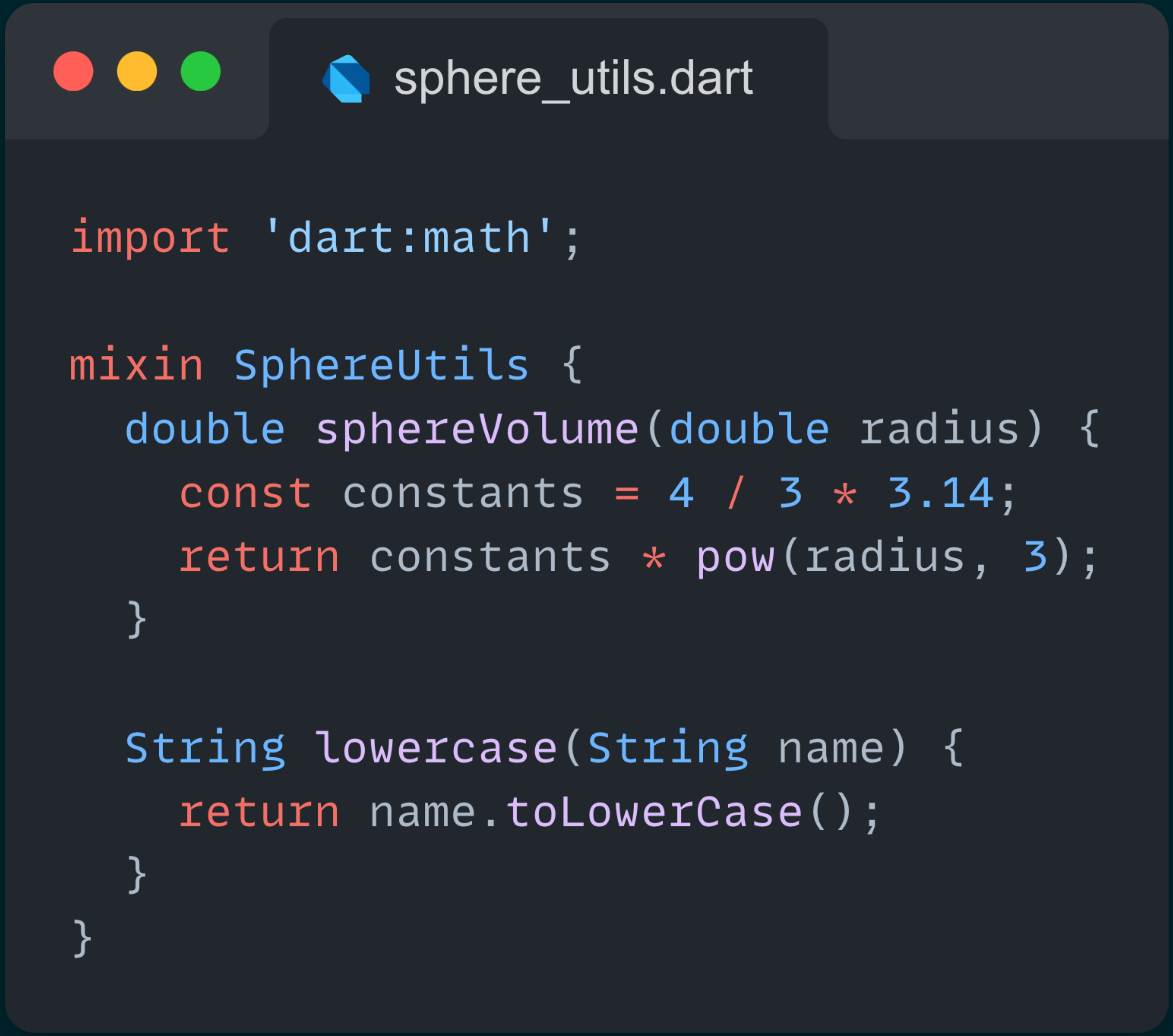
```dart
import 'dart:math';

abstract class VolleyballTeam {
  Country get country;
  List<Player> lineUp();
  double sphereVolume(double radius) {
    const constants = 4 / 3 * 3.14;
    return constants * pow(radius, 3);
  }

  String lowercase(String name) {
    return name.toLowerCase();
  }
}
```


planet.dart

```dart
import 'dart:math';

abstract class Planet {
  bool get inSolarSystem;
  List<Satellite> getSatellites();
  PlanetInfo getPlanetDetails();
  double sphereVolume(double radius) {
    const constants = 4 / 3 * 3.14;
    return constants * pow(radius, 3);
  }

  String lowercase(String name) {
    return name.toLowerCase();
  }
}
```

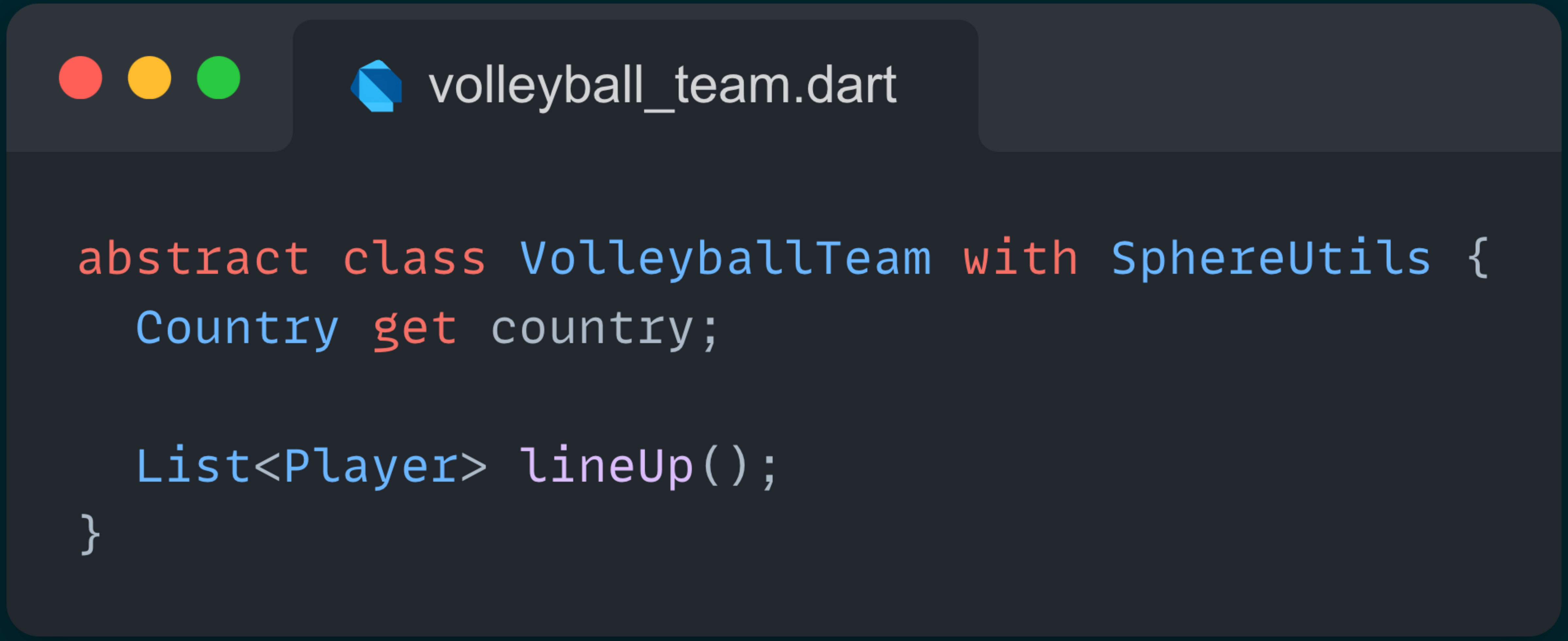
- Yuqorida ko’rishingiz mumkin, ikki sinf ham sphereVolume va lowercase metodlariga ega. Bu holat siz kodni takrorlayotganingizni anglatadi, bu DRY tamoyilini buzadi.
- Biroq, VolleyballTeam va Planet sinfi o'rtasida hech qanday umumiylik yo'q, shuning uchun superclass yaratish ma'nosiz bo'ladi. Shu bilan birga, biz kod takrorlanishidan ham qochishimiz kerak.
- Shunday holatlarda, inheritance dan foydalanmasdan metodlarni qayta ishlatmoqchi bo'lganingizda, mixin eng yaxshi yechimdir. Asosan, bu konstruktori bo'lmagan sinf bo'lib, u o'z a'zolarini baham ko'rish uchun istalgan boshqa sinf bilan "birlashtirilishi" mumkin.

Example

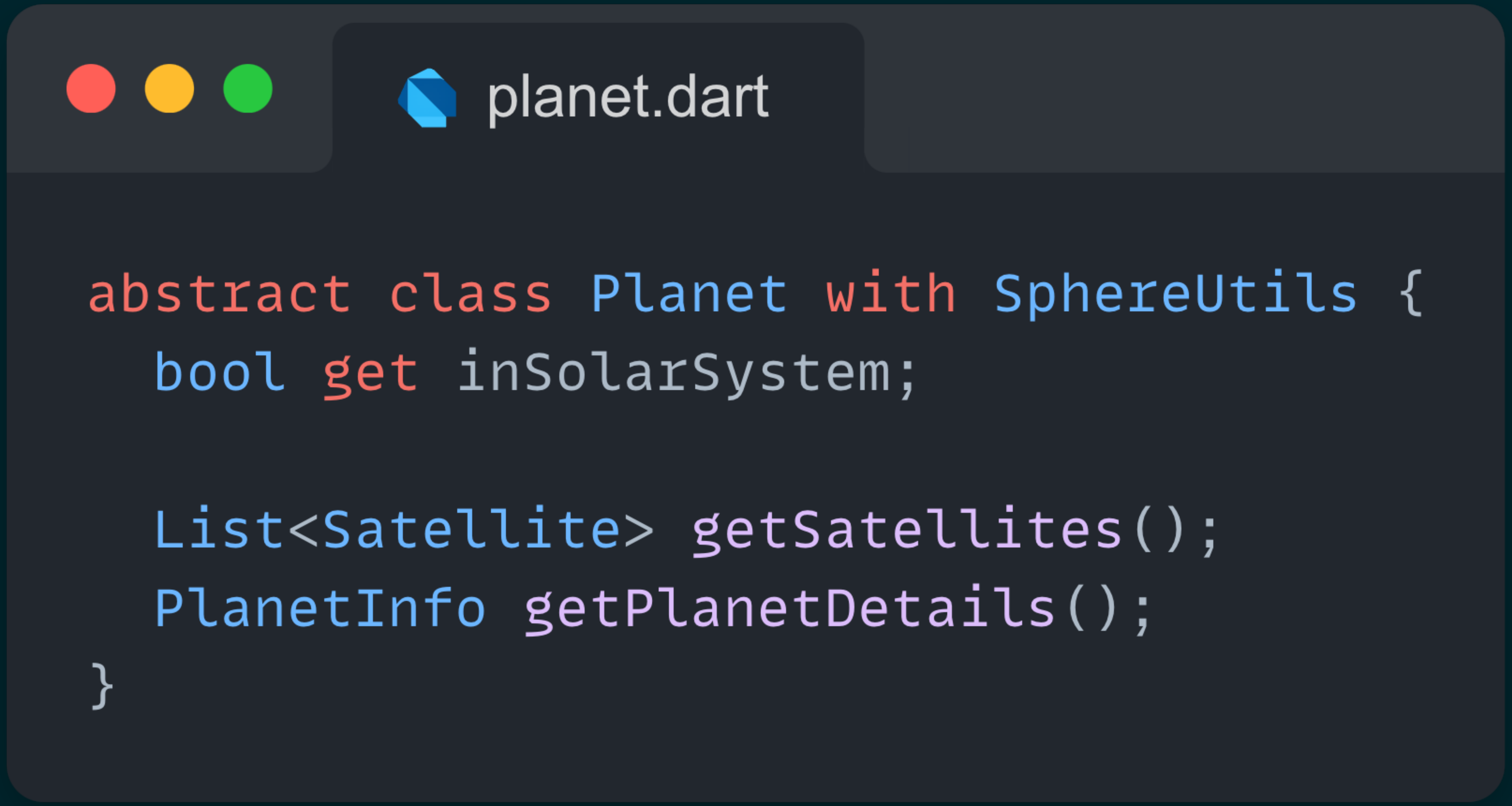

sphere_utils.dart

```dart
import 'dart:math';

mixin SphereUtils {
  double sphereVolume(double radius) {
    const constants = 4 / 3 * 3.14;
    return constants * pow(radius, 3);
  }

  String lowercase(String name) {
    return name.toLowerCase();
  }
}
```

- Biz "umumiy" metodlarni SphereUtils ichida to'pladik va endi uni boshqa sinflar bilan birgalikda, barcha a'zolarini avtomatik ravishda "import" qilishimiz mumkin. Mixin ni with kalit so'zi yordamida boshqa sinfga bog’laymiz:

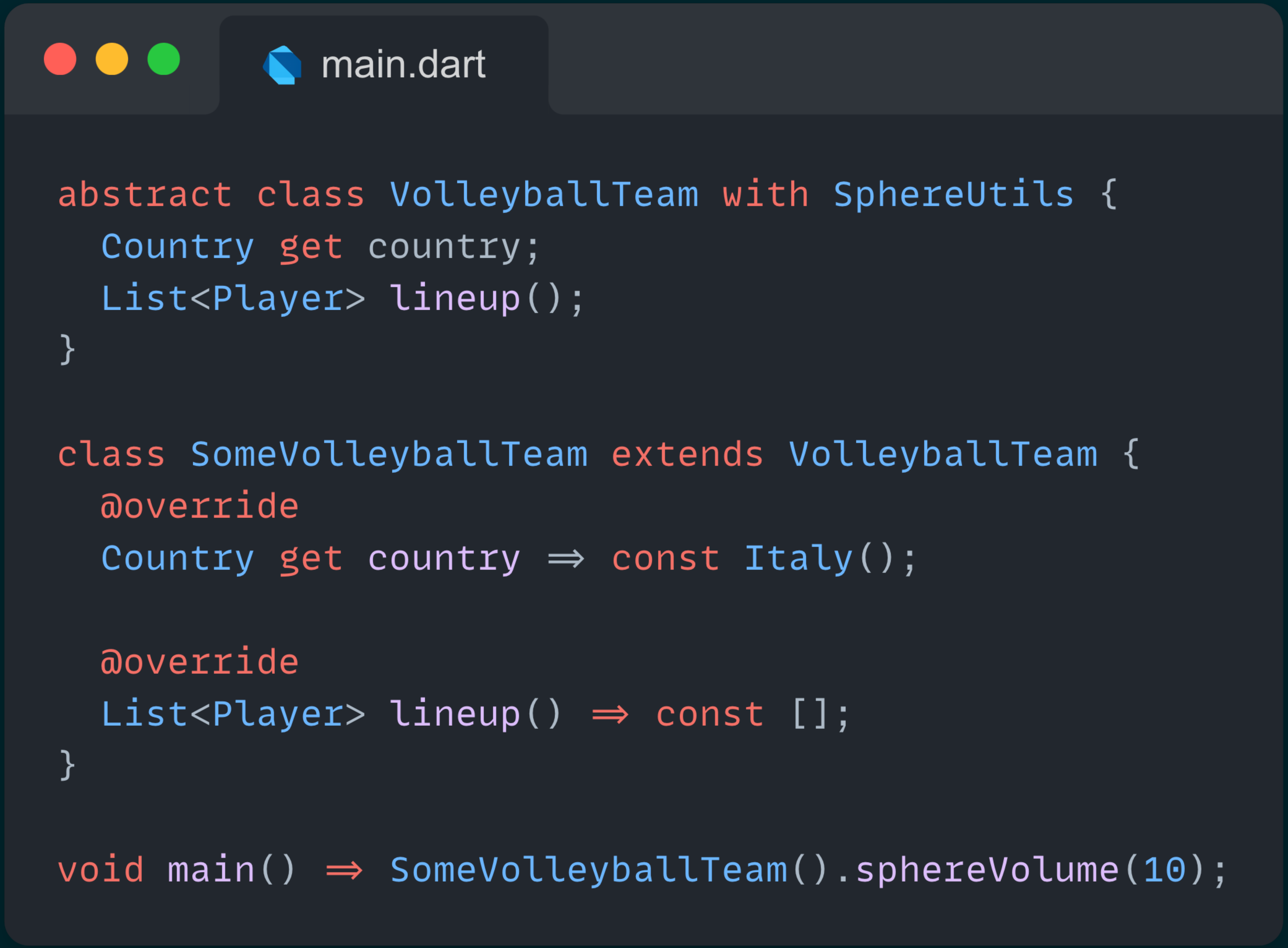

volleyball_team.dart

```dart
abstract class VolleyballTeam with SphereUtils {
  Country get country;

  List<Player> lineUp();
}
```


planet.dart

```dart
abstract class Planet with SphereUtils {
  bool get inSolarSystem;

  List<Satellite> getSatellites();
  PlanetInfo getPlanetDetails();
}
```

- Ko'rinmasa ham, VolleyballTeam va Planet endi sphereVolume va lowercase dan xuddi ular oddiy class a'zolari bo'lgandek foydalanishlari mumkin.
- Bu bilan inheritance dan qochdik va mavjud kodni qayta ishlatdik.
- VolleyballTeam sinfi sphereVolume - metodini aniq belgilamasa ham, ushbu metod avtomatik tarzda mixin dan "import".

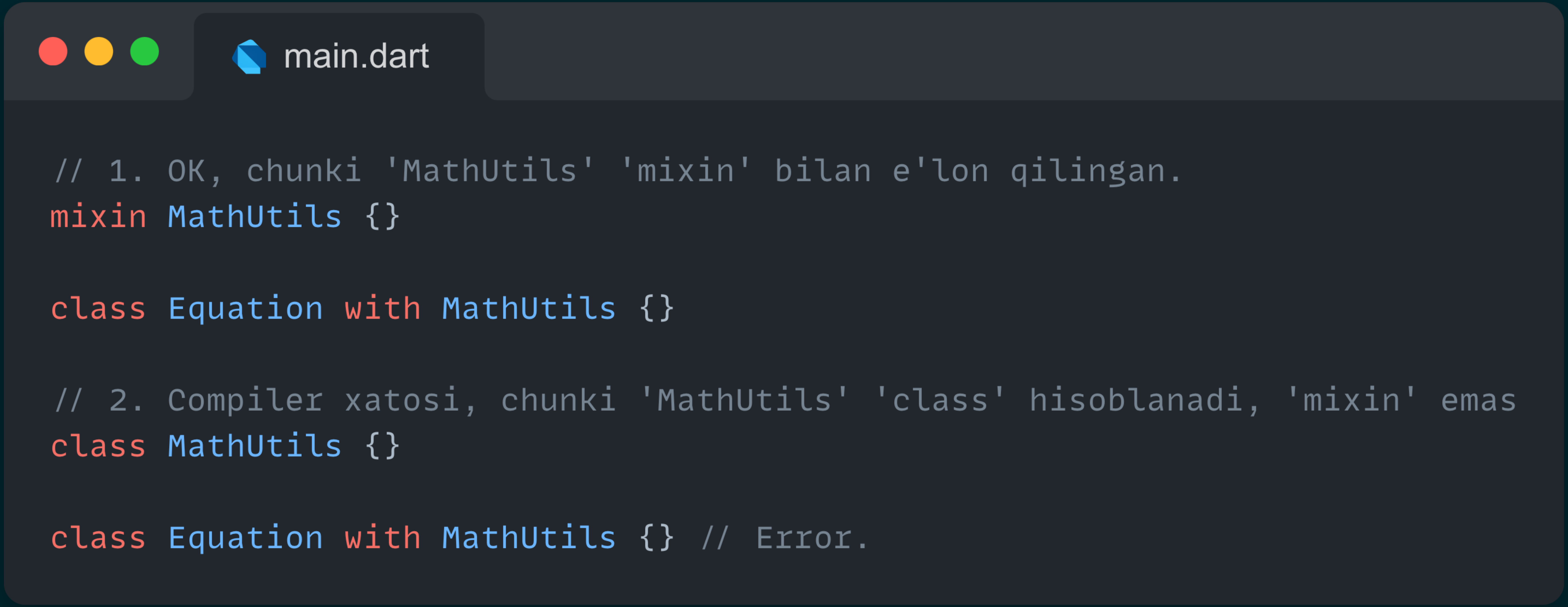


main.dart

```dart
abstract class VolleyballTeam with SphereUtils {
  Country get country;
  List<Player> lineup();
}

class SomeVolleyballTeam extends VolleyballTeam {
  @override
  Country get country => const Italy();

  @override
  List<Player> lineup() => const [];
}

void main() => SomeVolleyballTeam().sphereVolume(10);
```

Rules

- Oddiy sinflar mixin sifatida ishlatilolmaydi, deb eslatib o'tish kerak. with kalit so'zini faqat mixin modifikatori bilan e'lon qilingan sinflarda ishlatish mumkin.
- Agar siz class bilan with dan foydalanishga harakat qilsangiz, bu compile-time xatoga sabab bo'ladi. Faqat mixin modifikatori bilan e'lon qilingan sinfda mumkin:

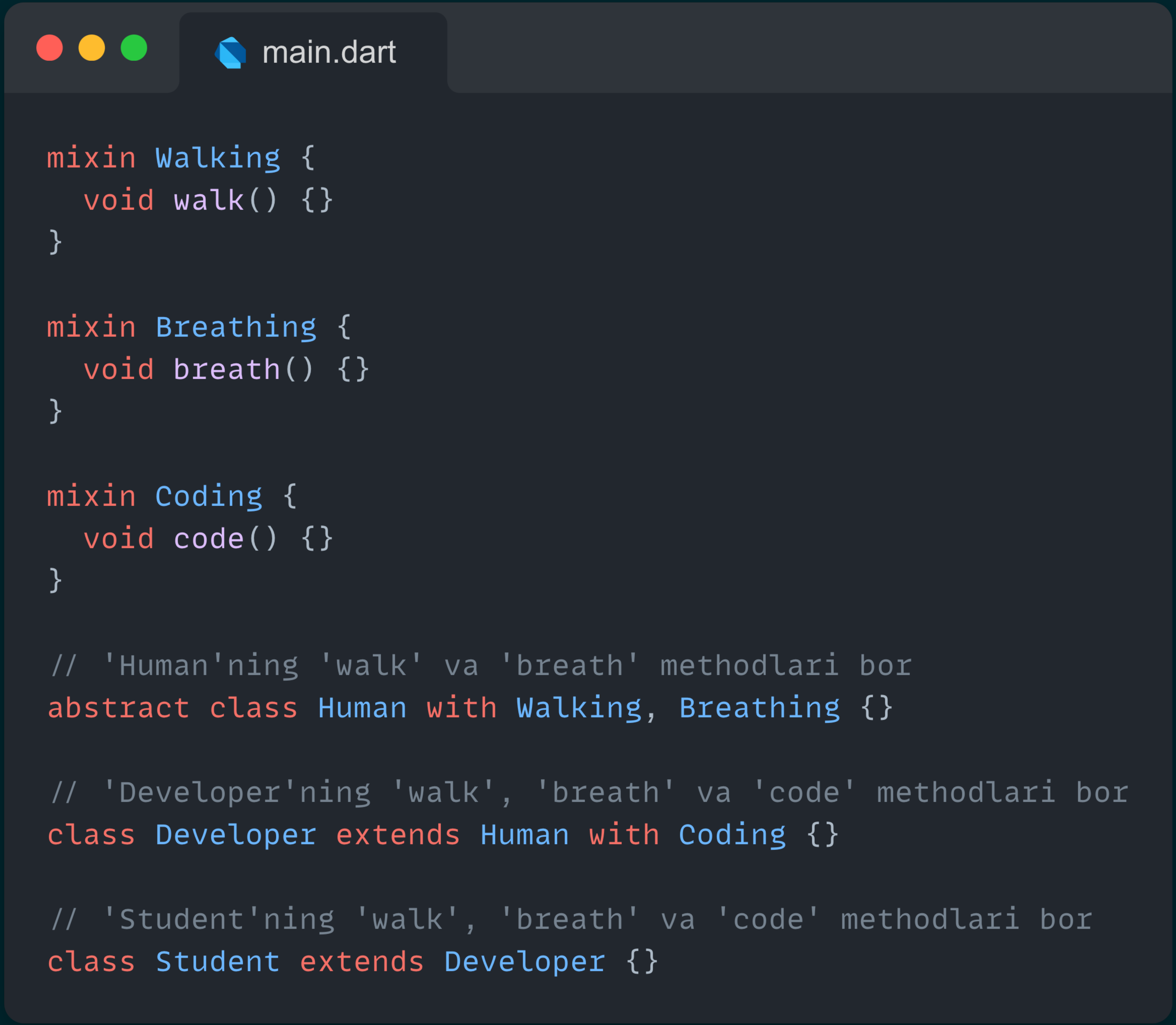

main.dart

```dart
// 1. OK, chunki 'MathUtils' 'mixin' bilan e'lon qilingan.
mixin MathUtils {}

class Equation with MathUtils {}

// 2. Compiler xatosi, chunki 'MathUtils' 'class' hisoblanadi, 'mixin' emas
class MathUtils {}

class Equation with MathUtils {} // Error.
```

Example

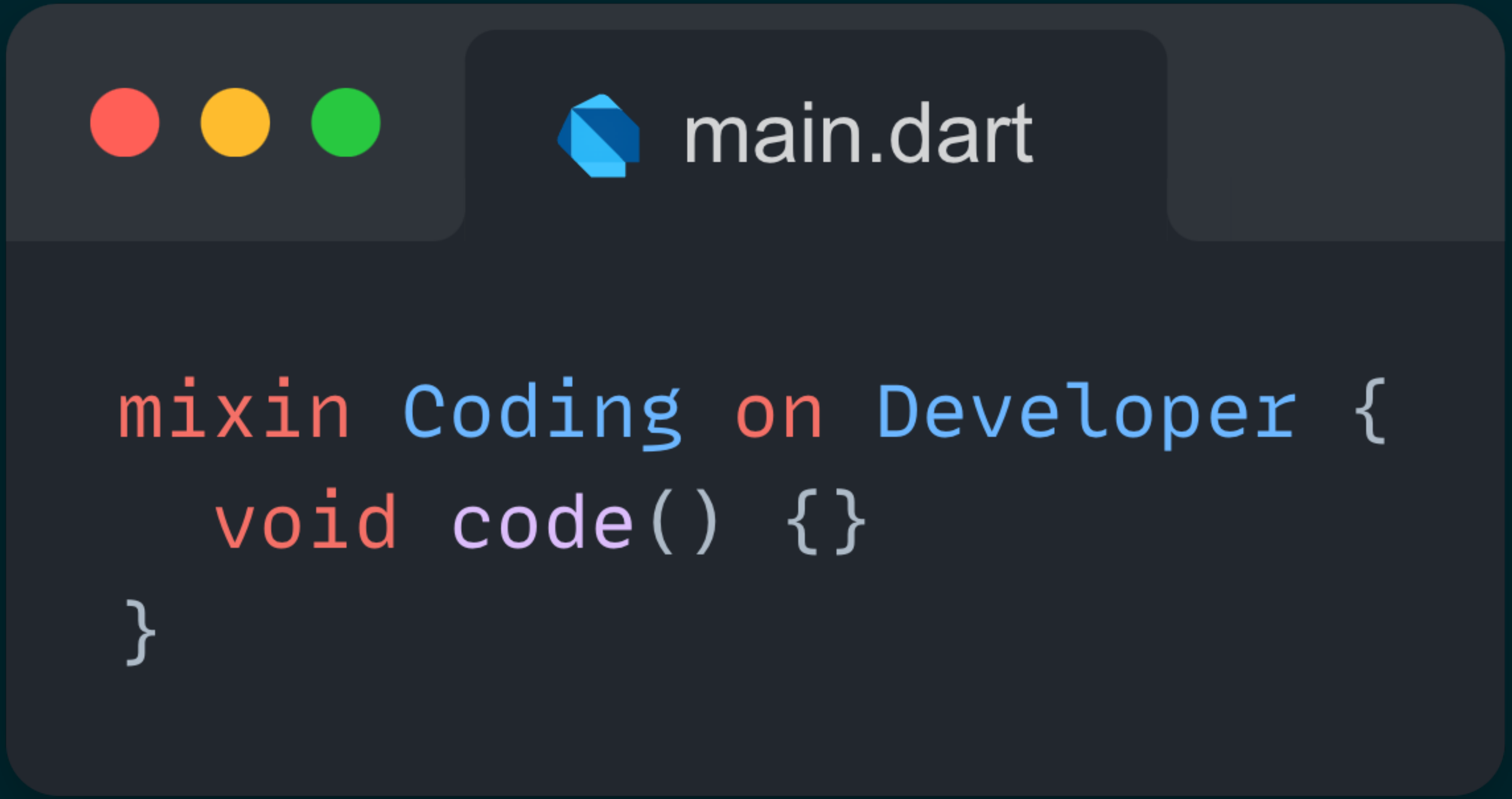
- Mixinlar instansiyalashtirilmaydi, shuning uchun mixin ichida konstruktor yaratish compile-time xatosi hisoblanadi.

- Bir class bir nechta mixin bilan mix qilish mumkin. Masalan:


main.dart

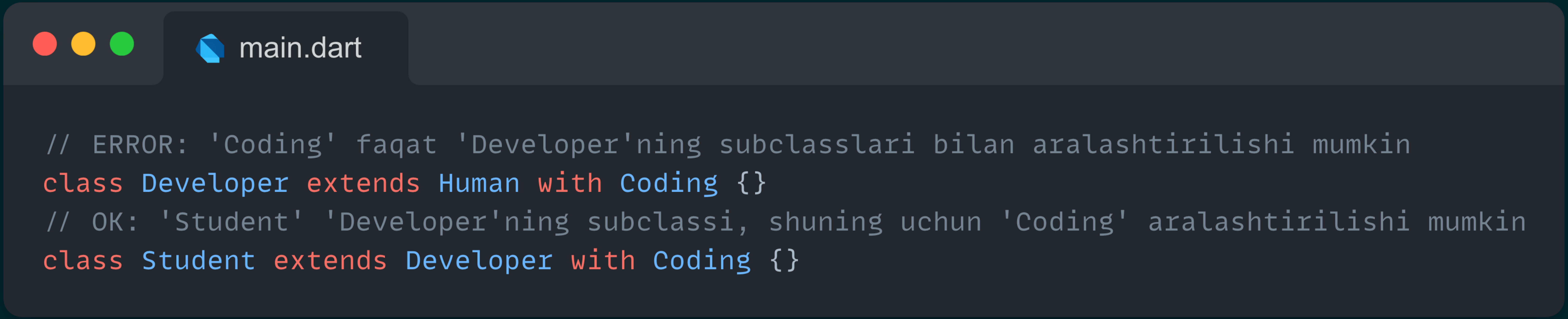
```dart
mixin Walking {
  void walk() {}
}

mixin Breathing {
  void breath() {}
}

mixin Coding {
  void code() {}
}

// 'Human'ning 'walk' va 'breath' methodlari bor
abstract class Human with Walking, Breathing {}

// 'Developer'ning 'walk', 'breath' va 'code' methodlari bor
class Developer extends Human with Coding {}

// 'Student'ning 'walk', 'breath' va 'code' methodlari bor
class Student extends Developer {}
```

- Mixindan foydalanishni on kalit so'zi yordamida aniq bir subtypeda cheklash mumkin. Masalan, Coding ni faqat Developer subclasslari bilan mix qilishni majburlashimiz mumkin:



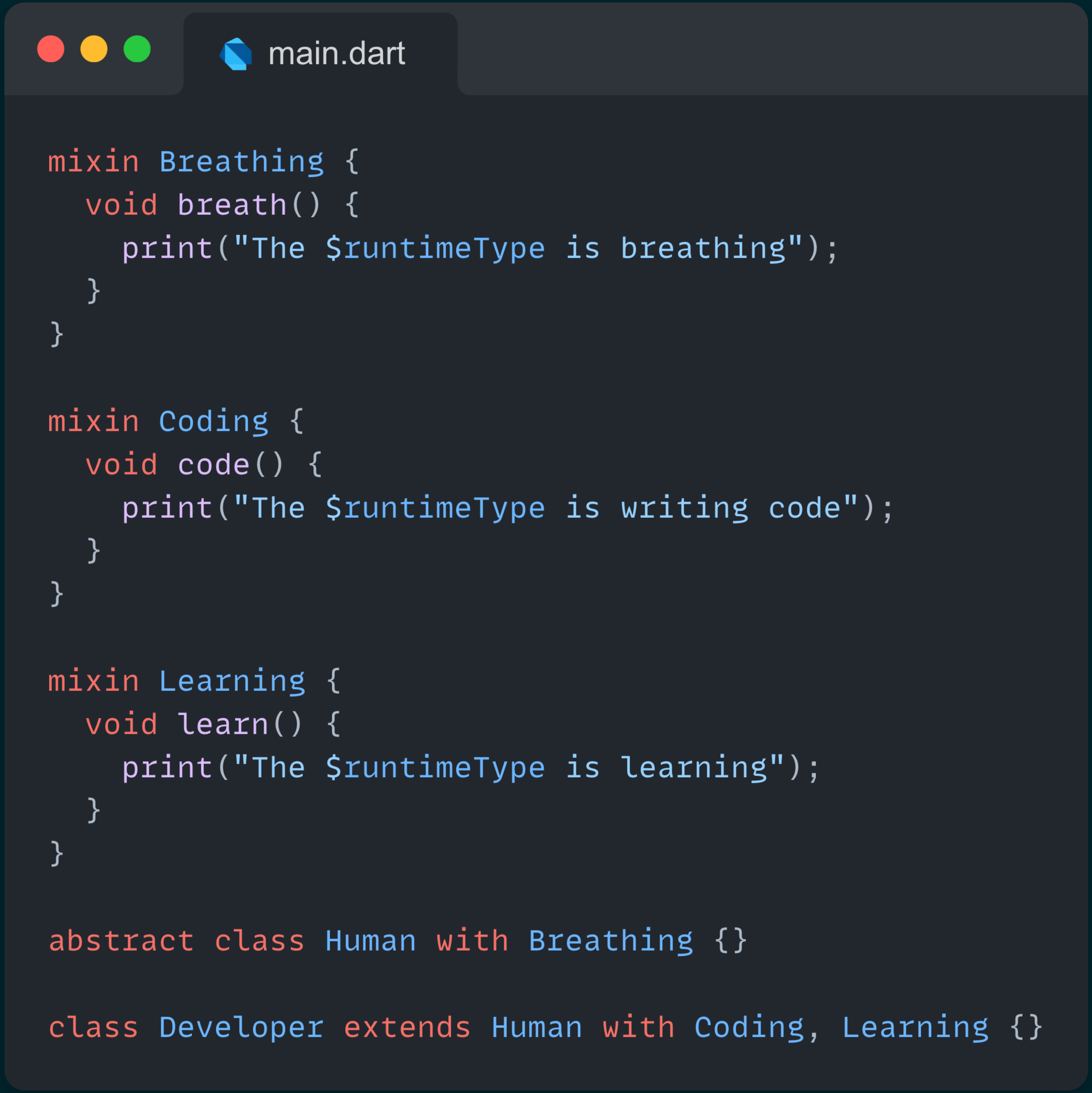
```dart
mixin Coding on Developer {
  void code() {}
}
```

- Bu o'zgarish bilan, faqat Developer dan subclass qilgan sinflar Coding bilan mix qilish mumkin.
- Agar siz Coding ni Developer o'zi yoki undan tarmoqlanmagan boshqa sinf bilan mix qilishga harakat qilsangiz, compilyator xatosini olasiz.
- on modifikatori bir yoki bir nechta turlarni ko'rsatishi mumkin.


main.dart

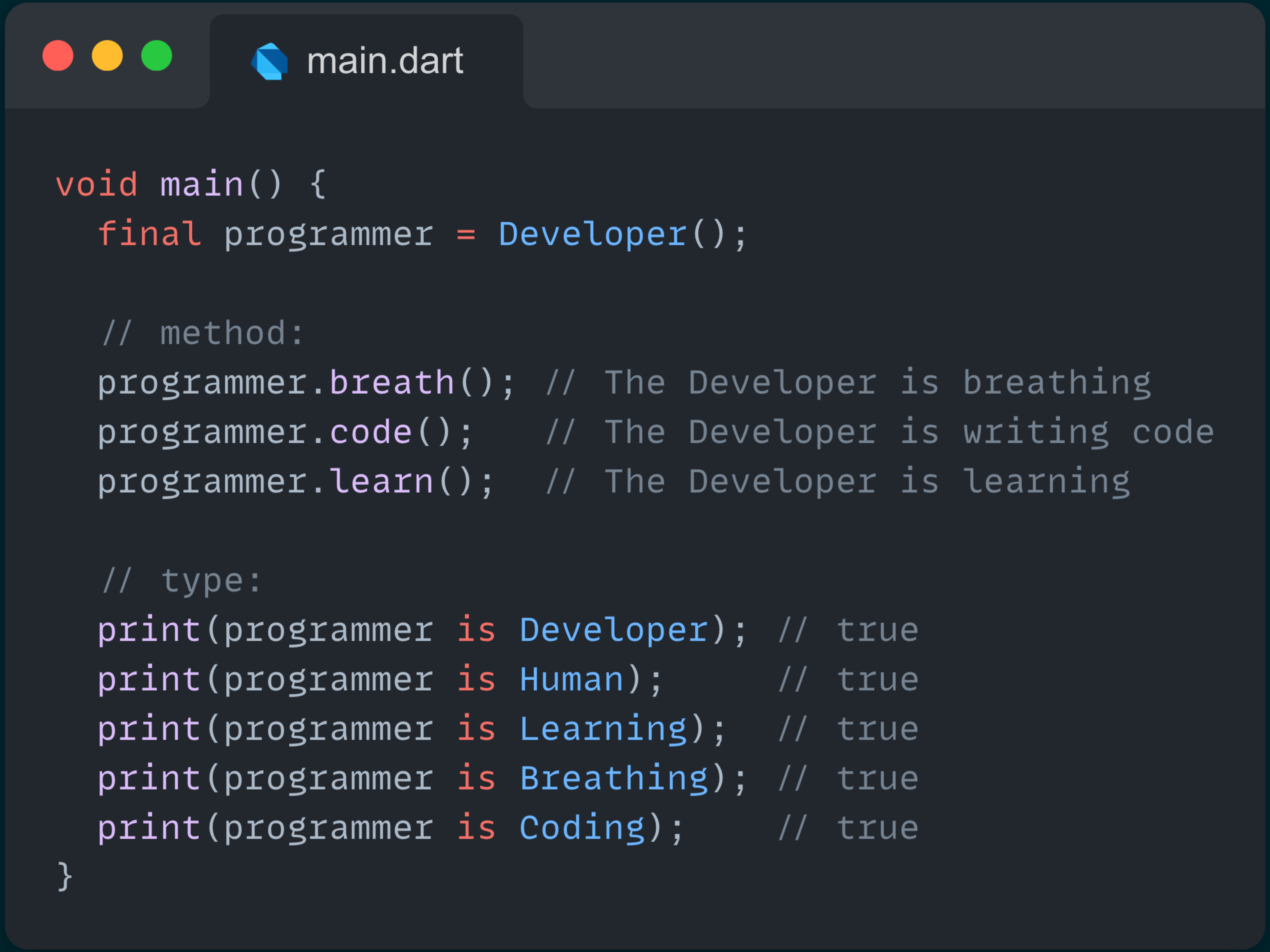
```dart
// ERROR: 'Coding' faqat 'Developer'ning subclasslari bilan aralashtirilishi mumkin
class Developer extends Human with Coding {}
// OK: 'Student' 'Developer'ning subclassi, shuning uchun 'Coding' aralashtirilishi mumkin
class Student extends Developer with Coding {}
```

- Developer sinfimiz juda ko'p mixinlar bilan mix qilinganda, uning asl turi qanday?
- Mixinlar meros qabul qilish ierarxiyasida qanday joylashgan?
- Keling, kodda tekshirib ko'ramiz!



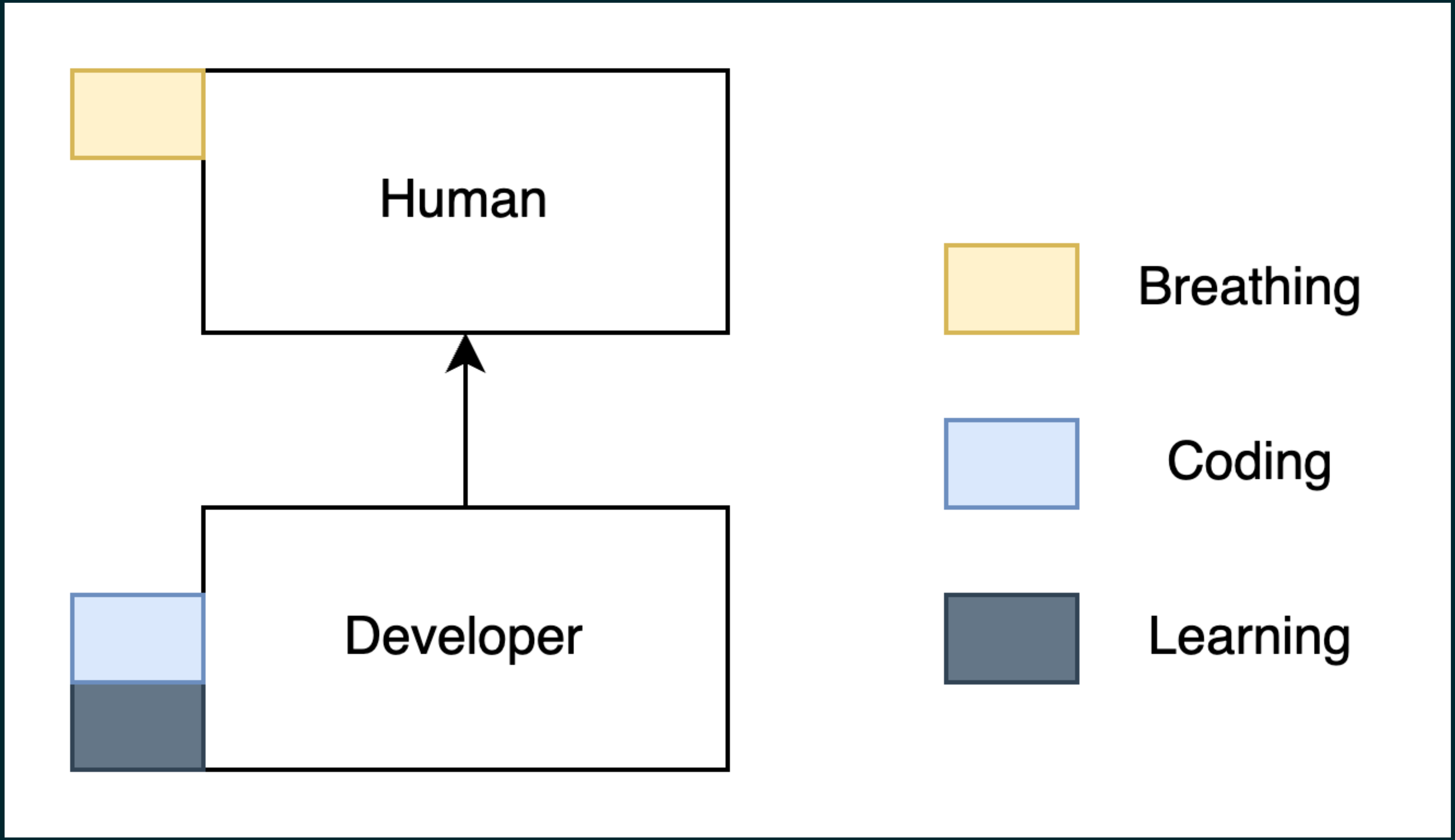
```dart
mixin Breathing {
  void breath() {
    print("The $runtimeType is breathing");
  }
}

mixin Coding {
  void code() {
    print("The $runtimeType is writing code");
  }
}

mixin Learning {
  void learn() {
    print("The $runtimeType is learning");
  }
}

abstract class Human with Breathing {}

class Developer extends Human with Coding, Learning {}
```

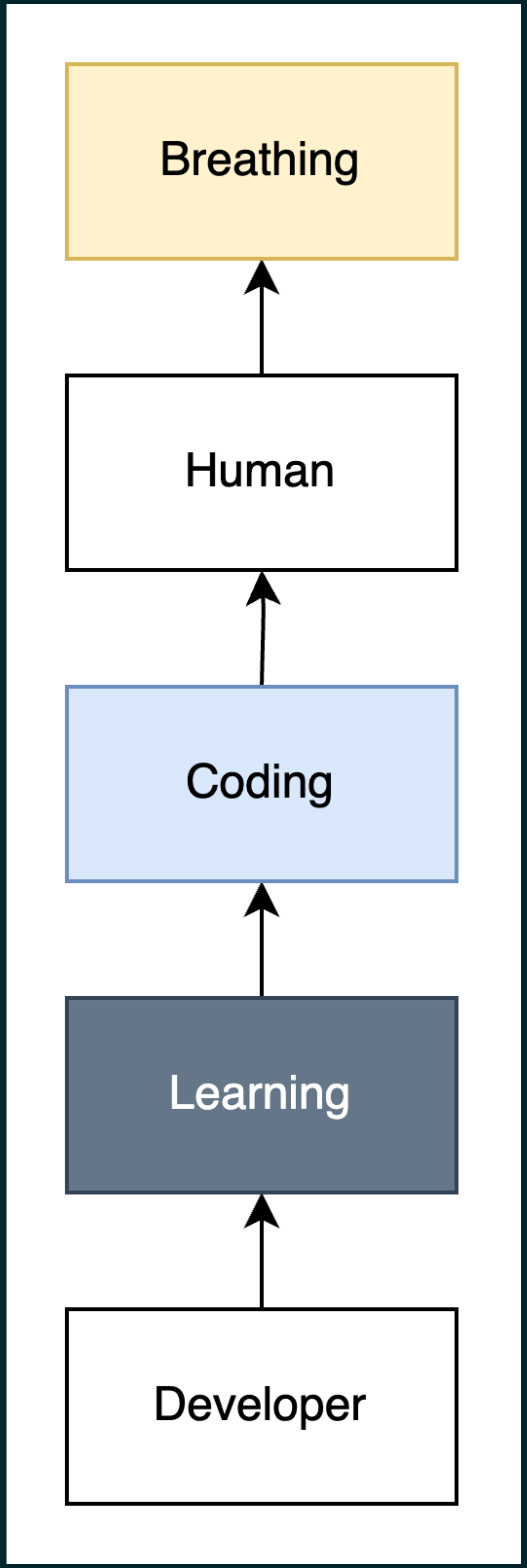
- Natijadan ko'rib turganimizdek, mixinlar polimorfikdir, xuddi sinflar kabi.
- Bir sinf o'zining ajdod sinflari va qo’shilgan mixinlari turiga aylanadi.


main.dart

```dart
void main() {
  final programmer = Developer();

  // method:
  programmer.breath(); // The Developer is breathing
  programmer.code();   // The Developer is writing code
  programmer.learn();  // The Developer is learning

  // type:
  print(programmer is Developer); // true
  print(programmer is Human);     // true
  print(programmer is Learning);  // true
  print(programmer is Breathing); // true
  print(programmer is Coding);    // true
}
```

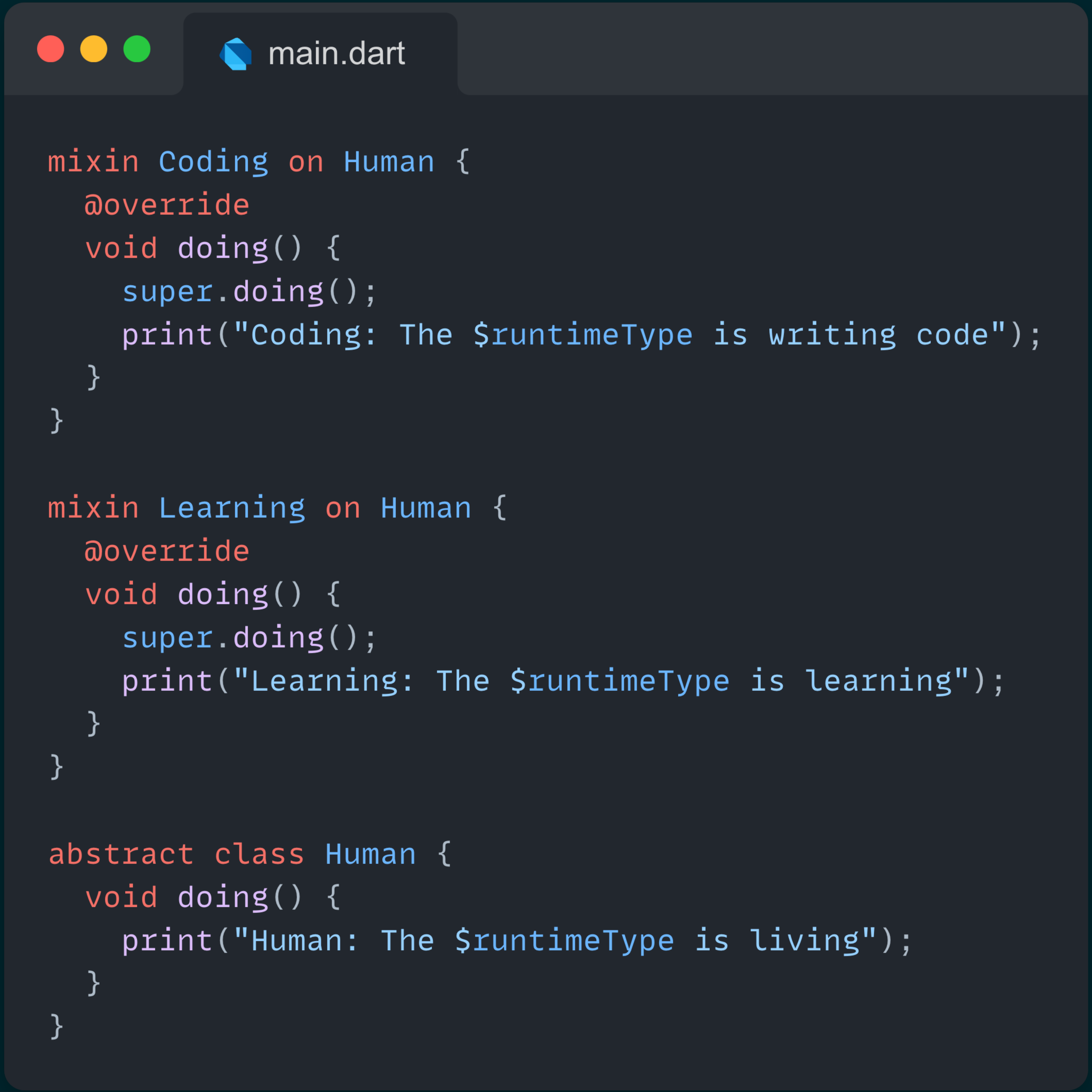
- Agar biz run timedagi meros qabul qilish tuzilmasiga qarasak, nima uchun bu yo'l bilan ishlashini ko'rishimiz mumkin:



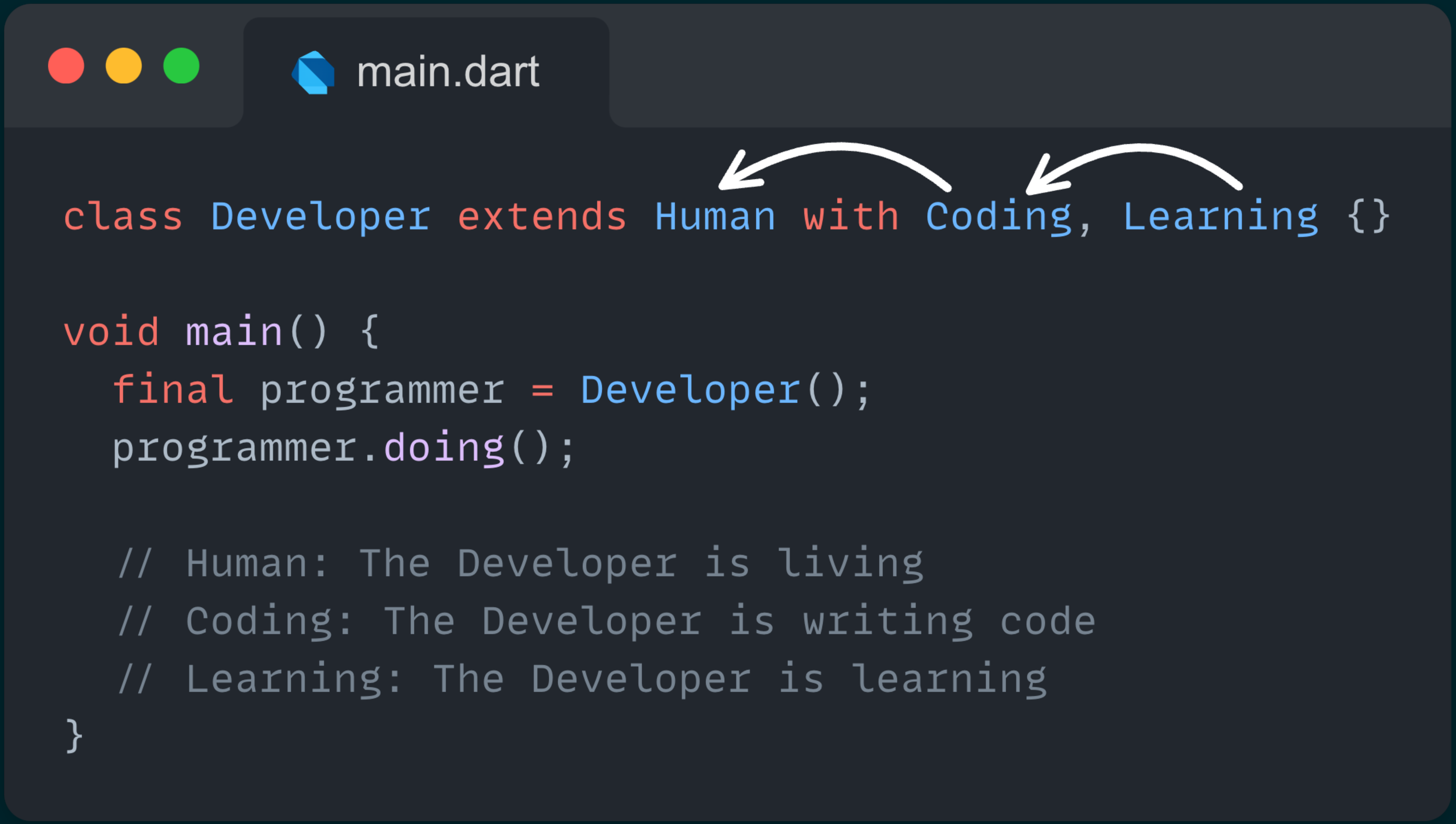


- Har bir mixin run timeda yangi sinf/interfeys yaratadi, u ierarxiyadagi keyingi sinf tomonidan meros qilib olinadi.
- Mixinlarning tartibi muhim ahamiyatga ega, chunki u mixinlar va sinflarning bir-biridan meros qilib olish tartibini belgilaydi.
- Asosiy qoida shundan iboratki, sinfning dekloratsiyasida har doim o'ngdan chapga o’qiladi.
- Mixinlar run timeda sinf ierarxiyasiga birlashtirilganligi sababli, ular methodlarni ham override qilishi mumkin.

Example

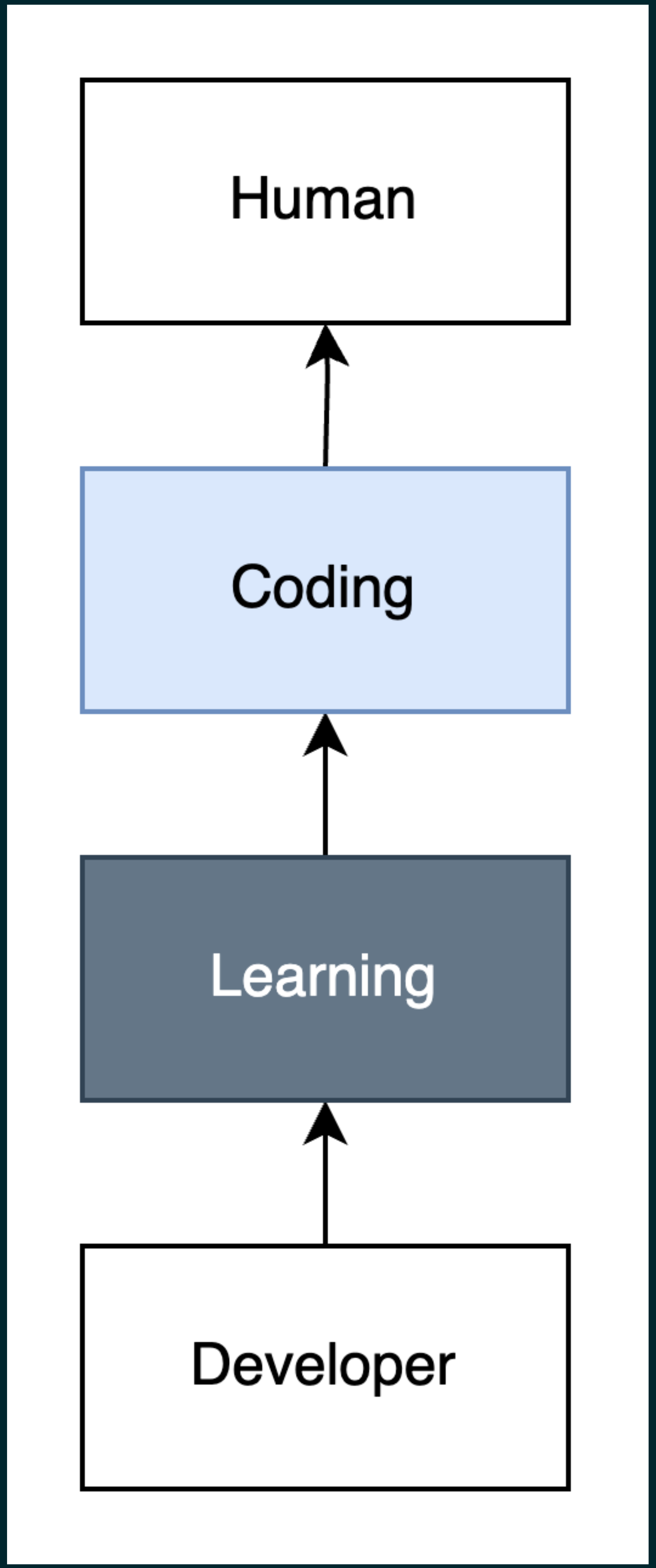


```dart
mixin Coding on Human {
  @override
  void doing() {
    super.doing();
    print("Coding: The $runtimeType is writing code");
  }
}

mixin Learning on Human {
  @override
  void doing() {
    super.doing();
    print("Learning: The $runtimeType is learning");
  }
}

abstract class Human {
  void doing() {
    print("Human: The $runtimeType is living");
  }
}
```

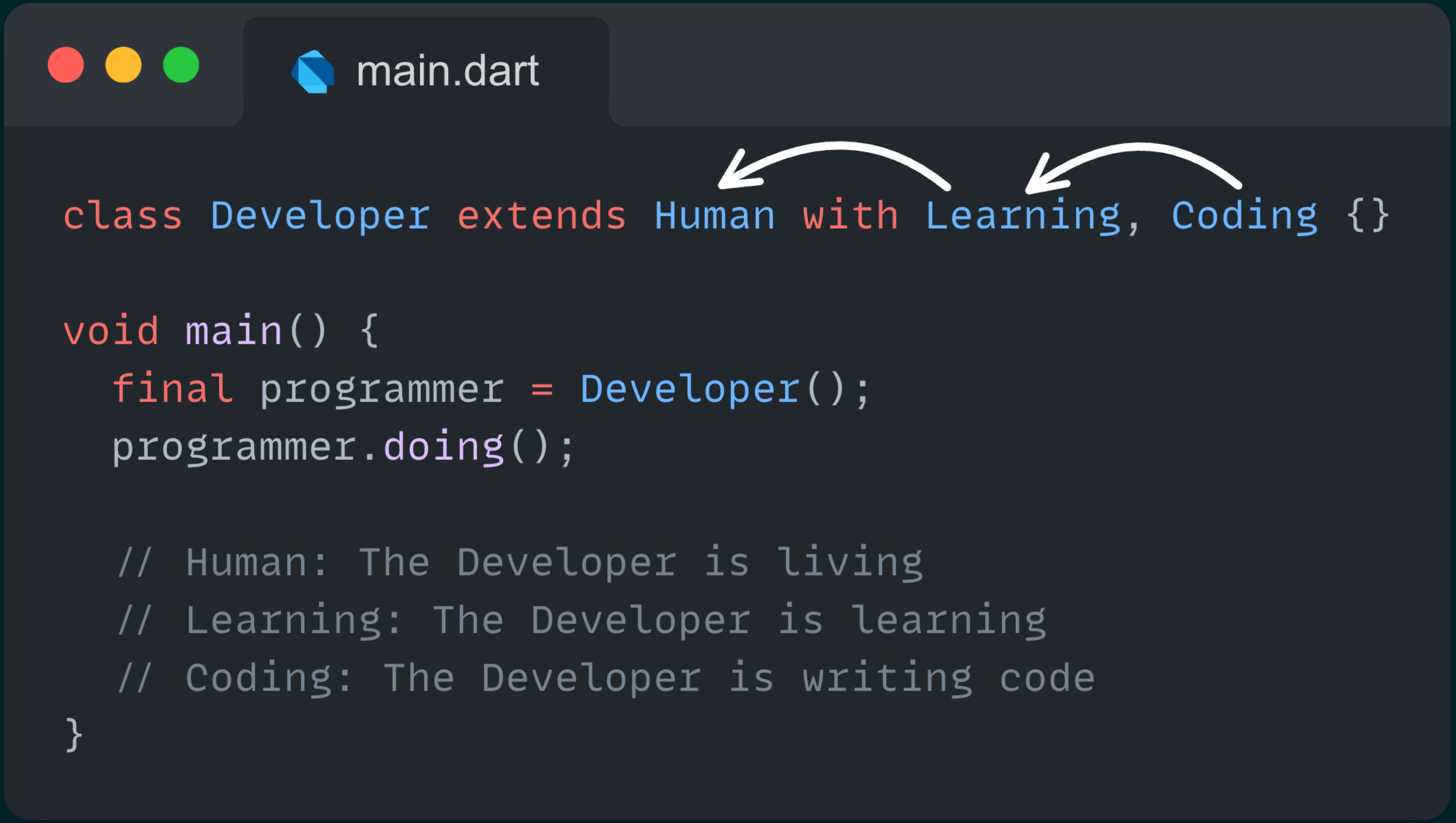


main.dart

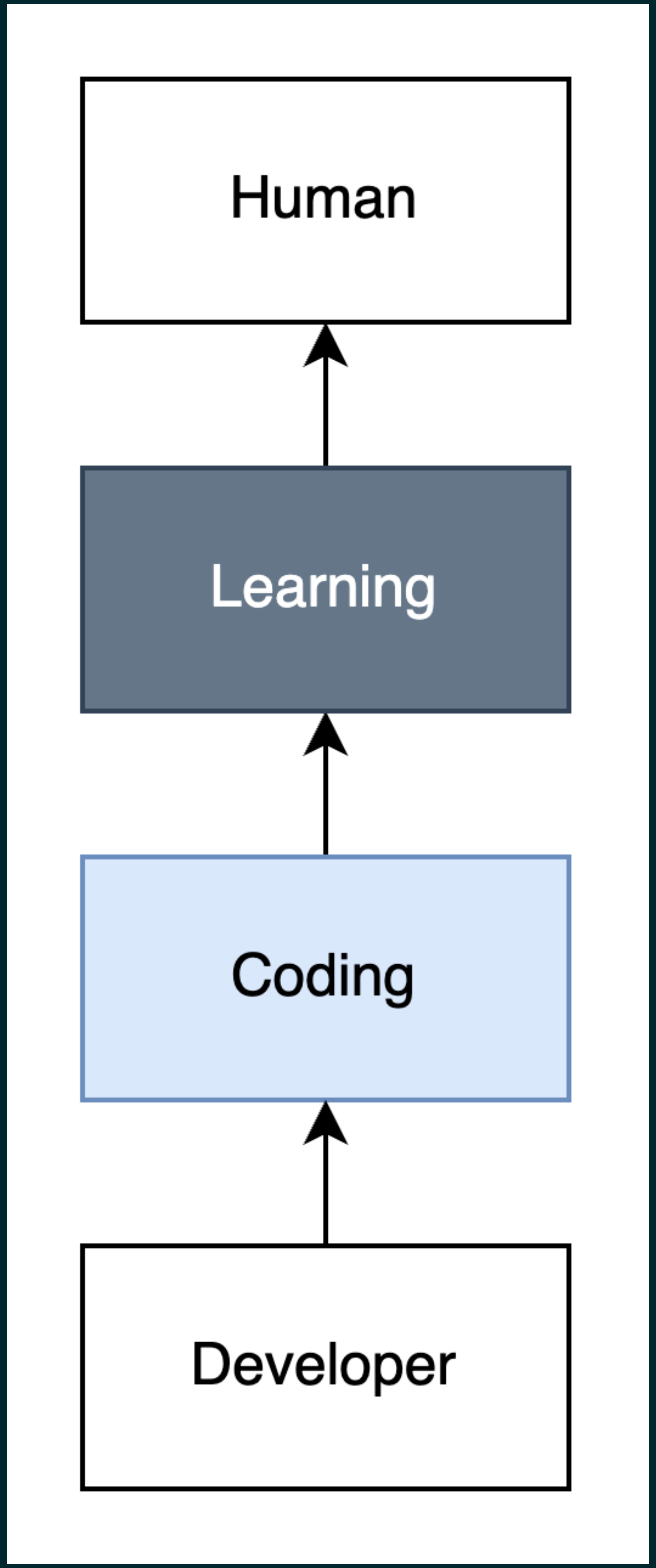
```dart
class Developer extends Human with Coding, Learning {}

void main() {
  final programmer = Developer();
  programmer.doing();

  // Human: The Developer is living
  // Coding: The Developer is writing code
  // Learning: The Developer is learning
}
```



- doing methodini chaqirganimizda, birinchi bo'lib Learning amalga oshiriladi, chunki u o'ng tomondan birinchi mixin.
- U avval super() ni chaqiradi, shuning uchun keyingi mixin o'ngdan (Coding) doing methodini bajarish uchun chaqiriladi. Yana u birinchi super() methodi tarkibidagi superni chaqiradi, bu esa Human sinfining doing methodini bajaradi.
- Shunday qilib, Human sinfining xabari birinchi consolega chiqariladi, keyin Coding ni xabari va nihoyat Learning xabari consolega chiqariladi.
- Agar biz Learning va Coding mixinlarini joylashuv tartibini o’zgartirsak, quyidagicha natija olamiz:


main.dart

```dart
class Developer extends Human with Learning, Coding {}

void main() {
  final programmer = Developer();
  programmer.doing();

  // Human: The Developer is living
  // Learning: The Developer is learning
  // Coding: The Developer is writing code
}
```



When to use mixin?

- Mixinlar biz bir nechta sinflar orasida xatti-harakatni bo'lishishni xohlaganimizda foydali, yoki bu xatti-harakatni superclassda amalga oshirish ma'nosiz bo'lganida foydali.

- Ammo haddan tashqari ko'p foydalanish muammo keltirib chiqarishi mumkin (har qanday dasturlash texnikasi bilan bo'lgani kabi). Mixinlar bir sinfga turli xil funktsiyalarni kiritishni juda oson qiladi. Lekin bir sinf juda ko'p narsalarni qila oladigan bo'lsa, bu uning maqsadi haqida umumiy tushunchani saqlash va intuitiv tushunishni qiyinlashtiradi. Bundan tashqari, qaysi method qaysi mixin'dan meros qilinganligi aniq bo'lmay qolishi mumkin va boshqa dasturchilar uchun kodning qaysi tartibda bajarilishini aniqlash chalkash bo'lishi mumkin.

Summary

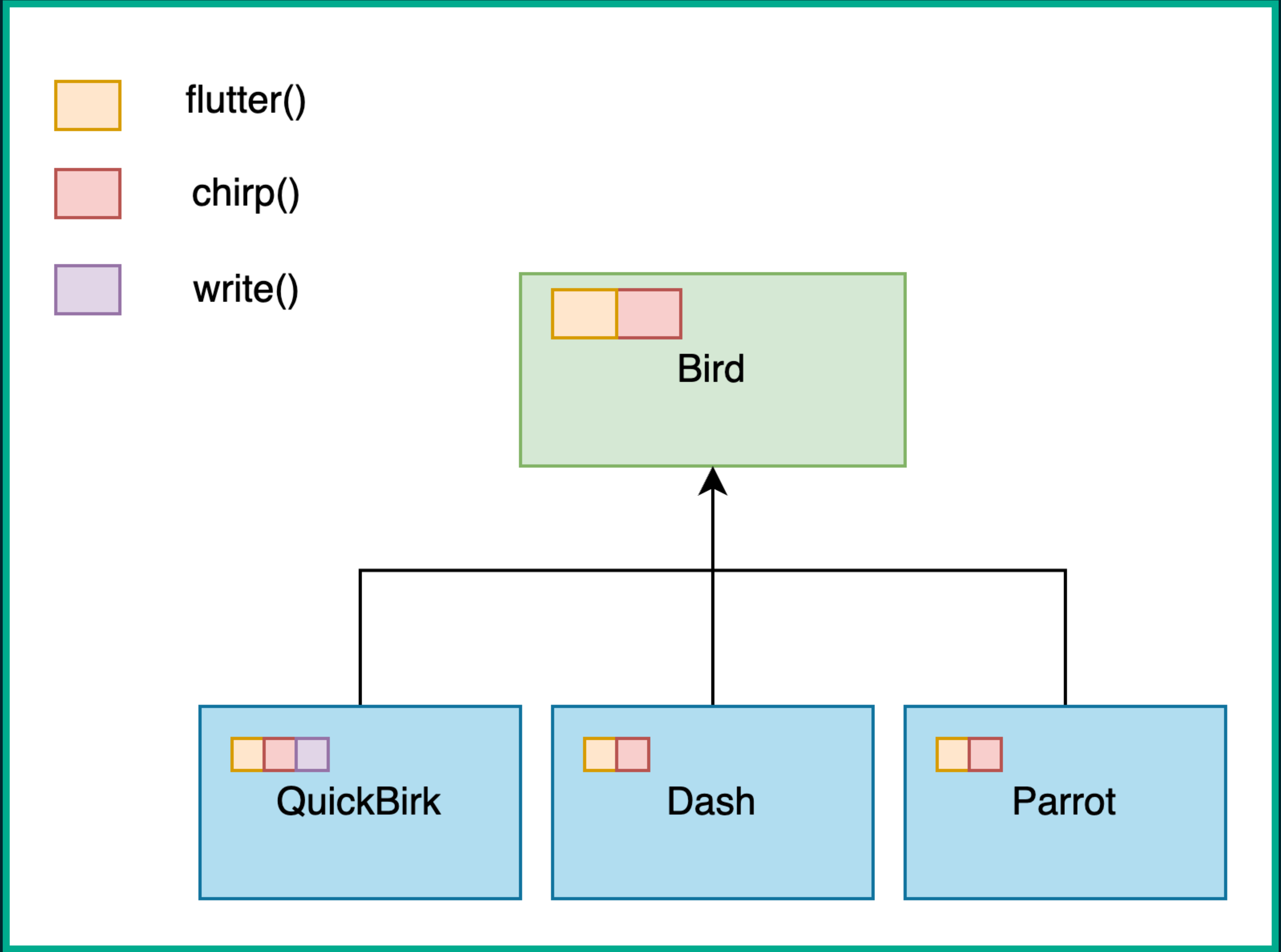
Interview Questions

- Mixin nima?
- Mixin qanday ko’rinishda hosil qilinadi va va qanday foydalaniladi?
- with kalit so’zi nima vazifani bajaradi?
- Mixin dan qanday holatlarda va nima uchun foydalanish kerak?
- Inheritance yoki interface da nima muammo borki biz mixin ishlatishimiz kerak.
- Bir klassda bir nechta mixindan foydalanish mumkinmi?

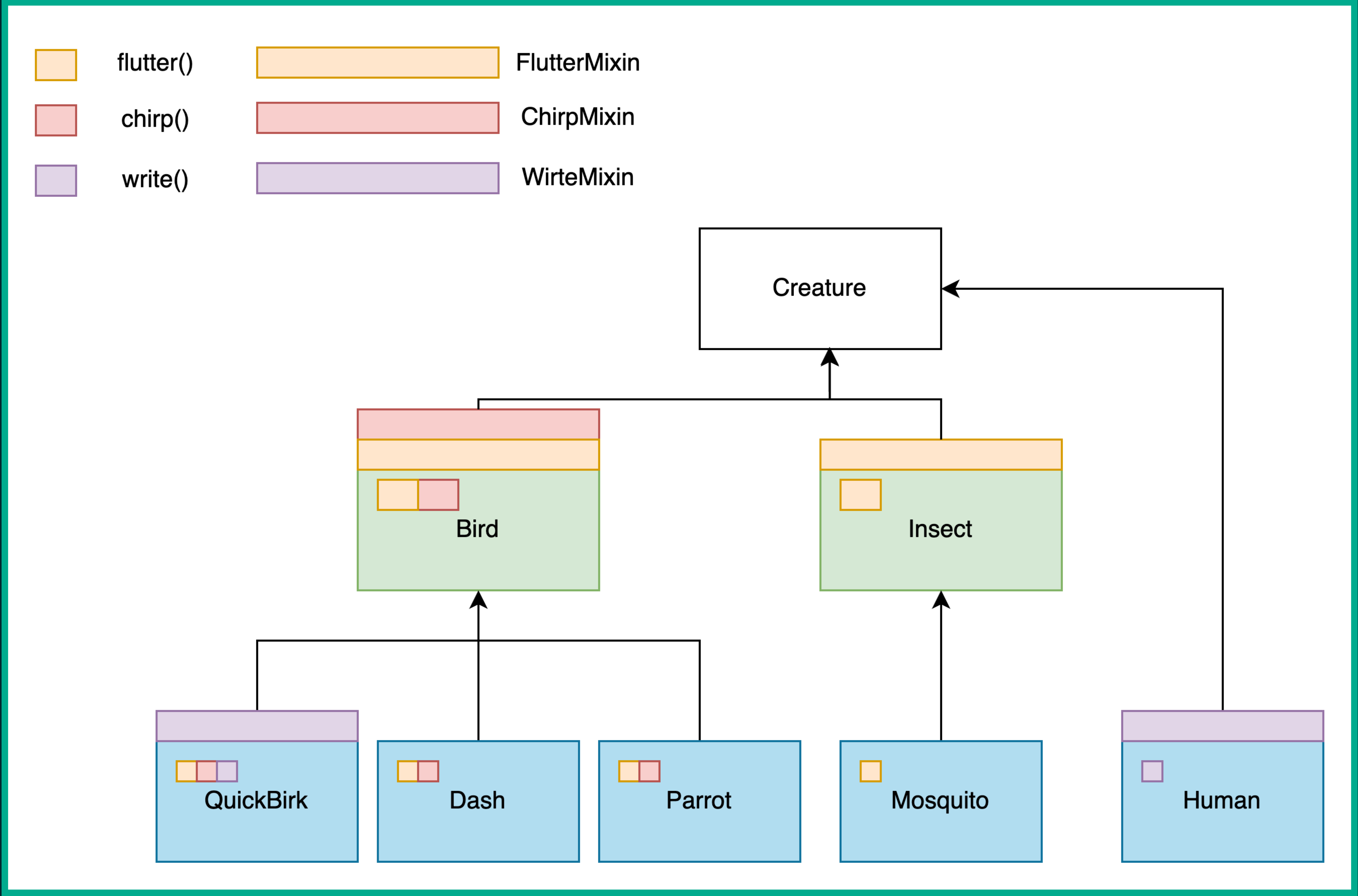
- Bir klassda bir nechta mixindan foydalanilsa ularni turgan o’rni ham ahamiyatga egami? Yani with kalit so’zidan so’ng birinchisi yoki ikkinchisi kabi taribi nazarda tutilgan.
- Agar bir klass ham ota klassdan meros olayotgan bo’lsa, ham interfeys va miksindan foydalanishi mumkinmi? Agar foydalanish mumkin bo’lsa ularning joylashuvi qanday bo’lishi kerak, birinchi extends orqali meros olamizmi yoki birinchi interfeys orqali impelements qilamizmi yoki avvalo mixin ni with kalit so’zi bilan klassimizga ulaymizmi?
- Oddiy klassdan ham mixin sifatida foydalanish mumkinmi? Agar mumkin bo’lsa u klass uchun qanday shartlar yani talablar bor?
- mixin e’lon qilinganda on kalit so’zidan foydalanishimiz mumkinmi? Agar mumkin bo’lsa on kalit so’zi yordamida qanday amal bajariladi?

Hometask

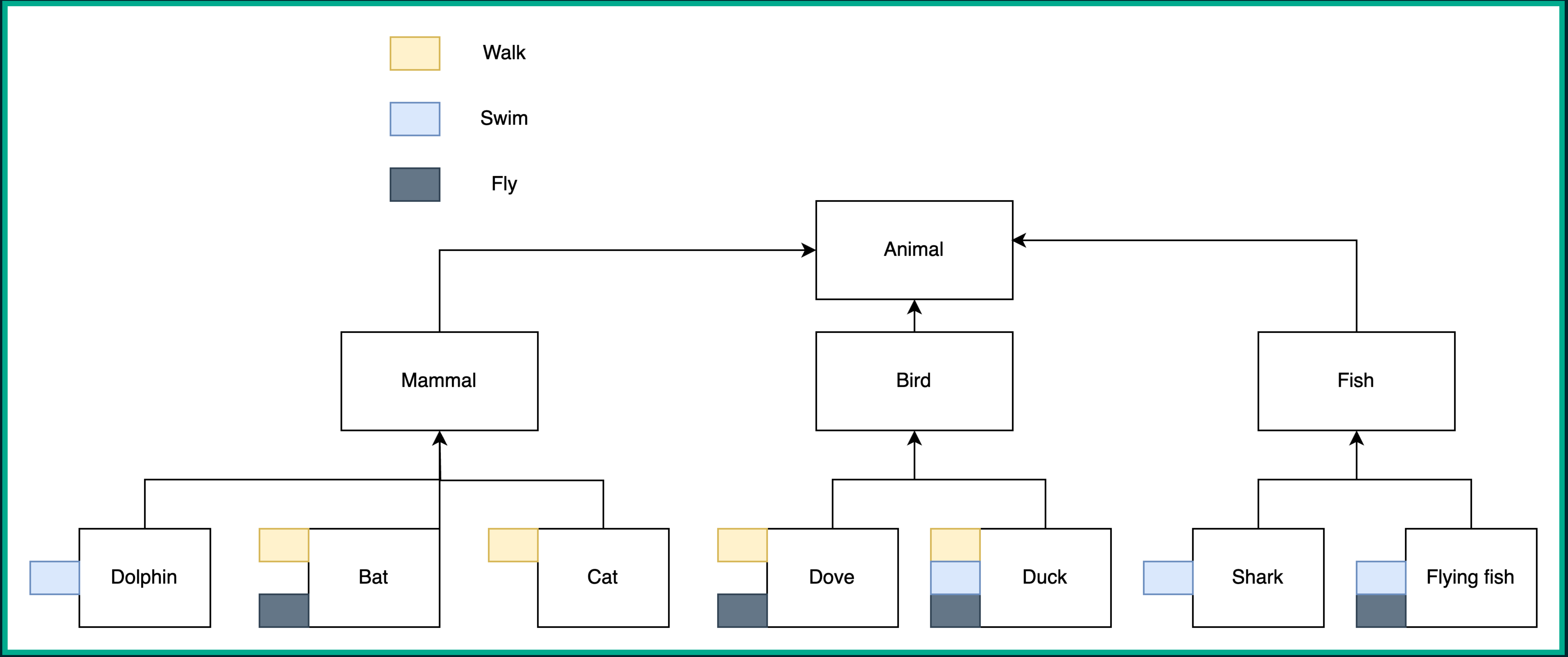
1. Quyidagi berilgan chizmaga asoslanib mixin va boshqa class larni hosil qiling.



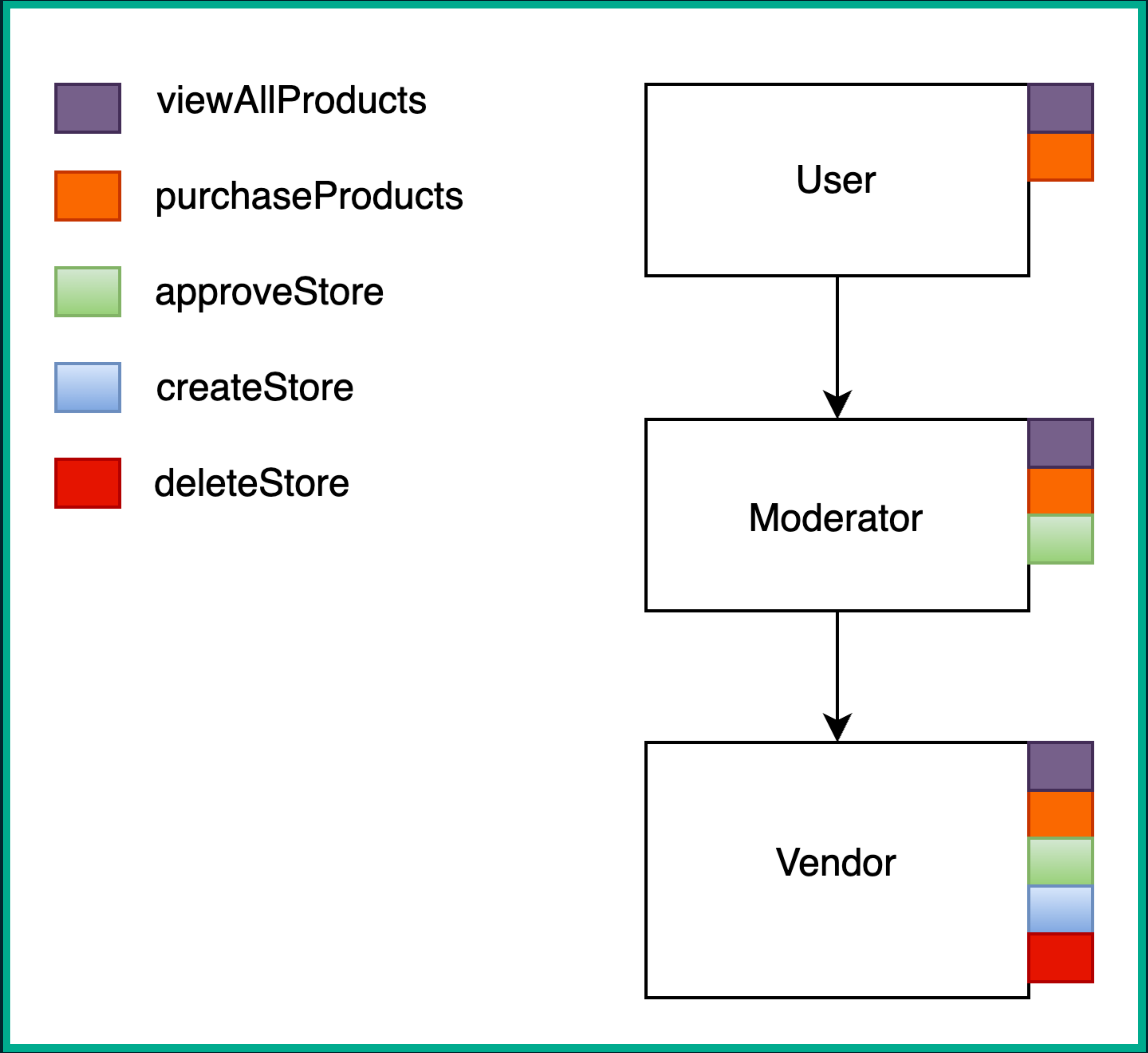
2. Quyidagi berilgan chizmaga asoslanib mixin va boshqa class larni hosil qiling.



3. Quyidagi berilgan chizmaga asoslanib mixin va boshqa class larni hosil qiling.



4. Quyidagi berilgan chizmaga asoslanib mixin va boshqa class larni hosil qiling.



Resource

- https://medium.com/flutter-community/dart-what-are-mixins-3a72344011f3
- https://quickbirdstudios.com/blog/flutter-dart-mixins/
- https://dart.dev/language/mixins

Q&A

Thank you for your time!