



Modul III. Lesson 1

Abstraction

Nasibali Abdiyev. Flutter Development

Repeat the previous lesson

- Callable class
- Cloning objects
- Regular Expressions

Plan

- What is Abstraction
- Abstraction in real life
- Abstract class
- Abstract methods
- Rules of Abstraction
- When to use Abstraction

What is Abstraction

- Abstraction(Abstraktsiya) - ilovaning ichki tafsilotlarini tashqi dunyodan yashirish jarayoni. Abstraktsiya narsalarni oddiy so'zlar bilan tasvirlash uchun ishlatiladi.
- Abstraktsiya murakkablikni ifodalash uchun oddiy sinflardan foydalanishni anglatadi. Asosan, biz foydalanuvchiga faqat tegishli va foydali ma'lumotlarni ko'rish imkonini berib, murakkablikni hal qilish uchun abstraksiyadan foydalanamiz.

- Buni tushuntirishning yaxshi misoli avtopilotli mashinani haydashdir. Agar sizda avtopilotli mashina bo'lsa va A nuqtadan B nuqtaga borishni istasangiz, unga borar joyni belgilashingiz va mashinani ishga tushirishingiz kifoya. Keyin u sizni manzilingizga olib boradi.
- Mashinaning qanday ishlab chiqarilgani, ko'rsatmalarni qanday to'g'ri qabul qilishi va unga rioya qilishi, avtomobil eng yaxshi marshrutni topish uchun turli xil variantlarni qanday filtrlashini bilishingiz shart emas va hokazo.
- Xuddi shu kontseptsiya OOP ilovalarini qurishda qo'llaniladi. Siz buni foydalanuvchi ko'rishi uchun zarur bo'lmagan tafsilotlarni yashirish orqali qilasiz. Abstraktsiya uni osonlashtiradi va loyihalaringizni kichik, boshqariladigan qismlarda boshqarishga imkon beradi.

Abstraction in real life

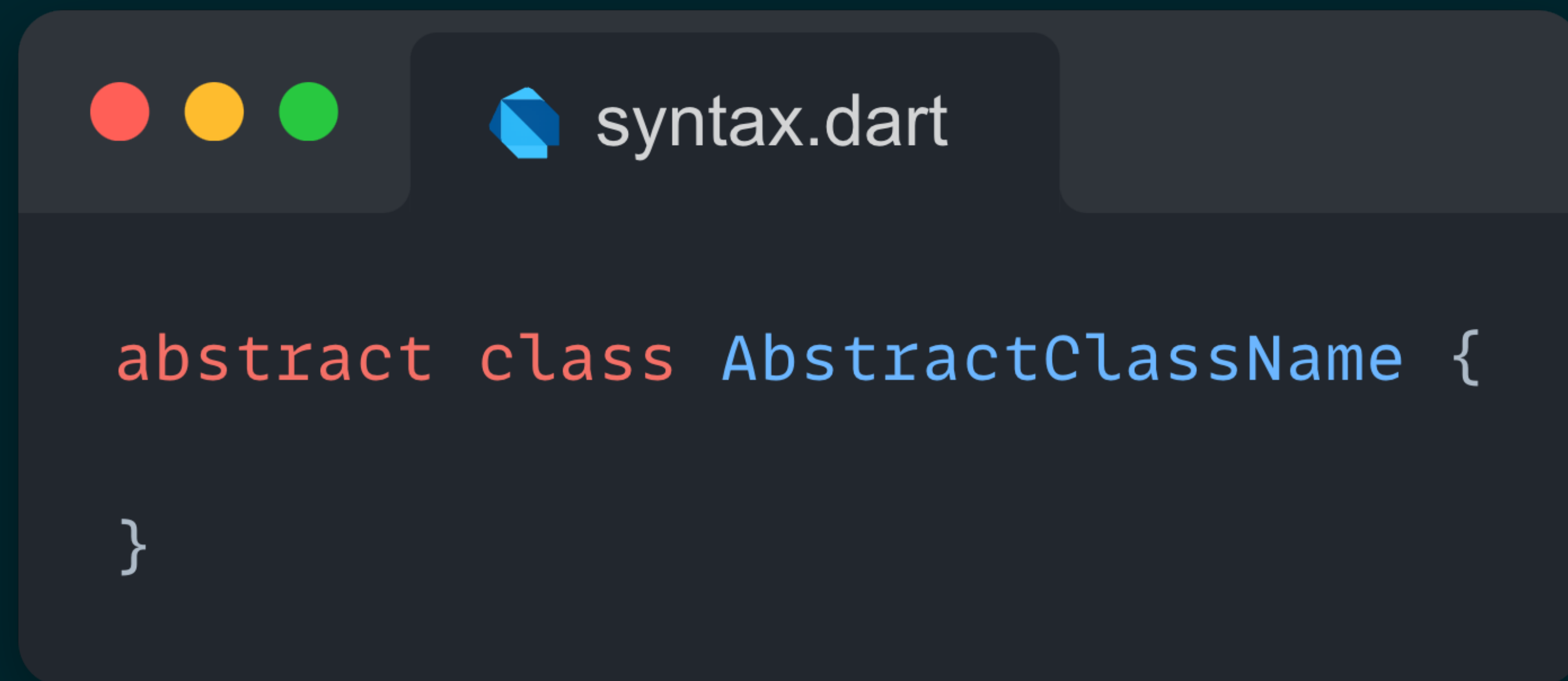
- Abstraktsiya deyarli barcha real hayot mashinalarida mavjud.
- Sizning mashinangiz abstrakt(mavhum)likning ajoyib namunasidir. Siz kalitni burish yoki start tugmasini bosish orqali mashinani ishga tushirishingiz mumkin. Dvigatel qanday ishga tushayotganini, mashinangizning barcha komponentlari qanday ekanligini bilishingiz shart emas. Avtomobilning ichki bajarilishi va murakkab mantiq foydalanuvchidan butunlay yashiringan.
- Mikroto'lqinli pechda ovqatni isitishimiz mumkin. Taymer va taom turini o'rnatish uchun ba'zi tugmalarni bosamiz. Nihoyat, biz issiq va mazali taom olamiz. Mikroto'lqinli pechning ichki tafsilotlari bizdan yashiringan. Bizga funktsionallikka juda oddiy tarzda kirish huquqi berildi.

- Ilgari siz yaratgan klasslar va sub klasslar aniq klasslar(concrete class) edi. Bu shunchaki siz ulardan haqiqiy narsalarni yasashingiz mumkinligini anglatadi. Yani ob'ektlar hosil qila olmaydigan mavhum klasslar(abstract)dan farqli o'laroq.
- Siz doimo abstrakt tushunchalar bilan shug'ullanasiz va ular haqida umuman o'ylamaysiz.
- Siz hayvonni ko'rganmisiz? Yani "siz tovuq yoki sichqonchani ko'rganmisiz" emas - "hayvon" ning o'zi? Bu nimaga o'xshaydi? Uning oyog'i doim 4 ta bo'ladimi, o'rdaklar hayvonlardir va ularning ikkita oyog'i bor. Uning sochlari bo'lishi mumkinmi, chunki ilonlar hayvonlardir va ularning sochlari yo'q. Bazi birlarida na ko'zlar va na suyaklari bor.

- Hech kim to'g'ri ma'noda "hayvon" ni ko'rmagan, lekin hamma abstrakt hayvonlar toifasiga mos keladigan narsalarning aniq misollarini ko'rgan. Insonlar o'zlari olib borgan kuzatishlarini umumlashtirish va turkumlashtirishni yaxshi biladilar, bu abstraktlik juda foydali. Ular sizga "Men sher, fil, lemur, akulani ko'rdim ..." o'rniga "Men hayvonot bog'ida juda ko'p hayvonlarni ko'rdim" kabi qisqa bayonotlar berishga imkon beradi.
- Xuddi shu narsa ob'ektga yo'naltirilgan dasturlashda ham qo'llaniladi. Siz yozayotgan sinflaringizning shakllari va umumiy xususiyatlarini sezishni boshlaysiz. Shunday qilib, sinfni amalga oshirishning aniq usulini ko'rsatmasdan, sinfning umumiy xususiyatlari va xatti-harakatlarini tasvirlashni xohlash nuqtasiga kelganingizda, siz abstrakt sinflarni yozishga tayyor bo'lasiz. Ba'zi tillarda bu umumlashtirilgan xatti-harakatlar protokol deb ataladi, ammo Dartda u interfeys deb ataladi. Ammo bu haqida keyingi dars suhbatlashamiz. Hozir esa abstrakt sinflarni qanday hosil qilinishini o'rganamiz.

Abstract class

- Dartda abstrakt sinfini yaratish uchun biz ***abstract*** kalit so'zdan foydalanamiz.



A code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) and a tab labeled 'syntax.dart' with a blue icon. The code inside is:

```
abstract class AbstractClassName {  
  
}
```

Abstract methods

- abstract kalit so'z yordamida yaratilgan sinf mavhum va mavhum bo'lmagan methodlarga (body ga ega bo'lgan method) ega bo'lishi mumkin. Endi mavhum methodlar nima?
- Mavhum methodlar (abstract method) - bu amalga oshirilmaydigan methodlar yani body ga ega emas. Ular faqat boshqa sinflar ularni amalga oshirishni, yani override qilishini kutishadi.

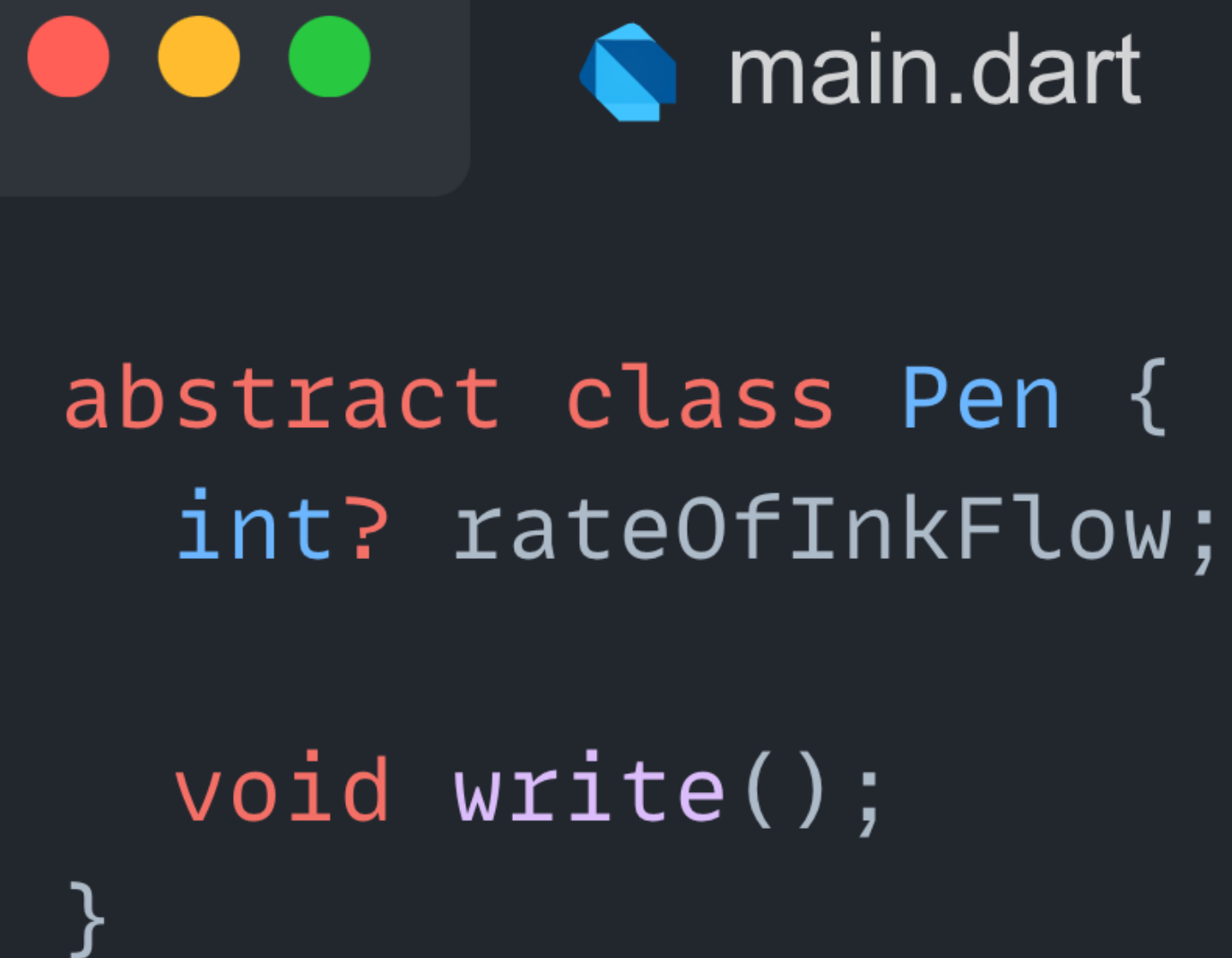


syntax.dart

```
abstract class AbstractClassName {  
    type abstractMethodName(parameters);  
}
```

Abstract Class and Abstract Methods

- Bu yerda write methodi abstrakt method hisoblanadi. Buning sababi, uning body qismi mavjud emas.



```
abstract class Pen {  
  int? rateOfInkFlow;  
  
  void write();  
}
```

Abstract classes can't be instantiated

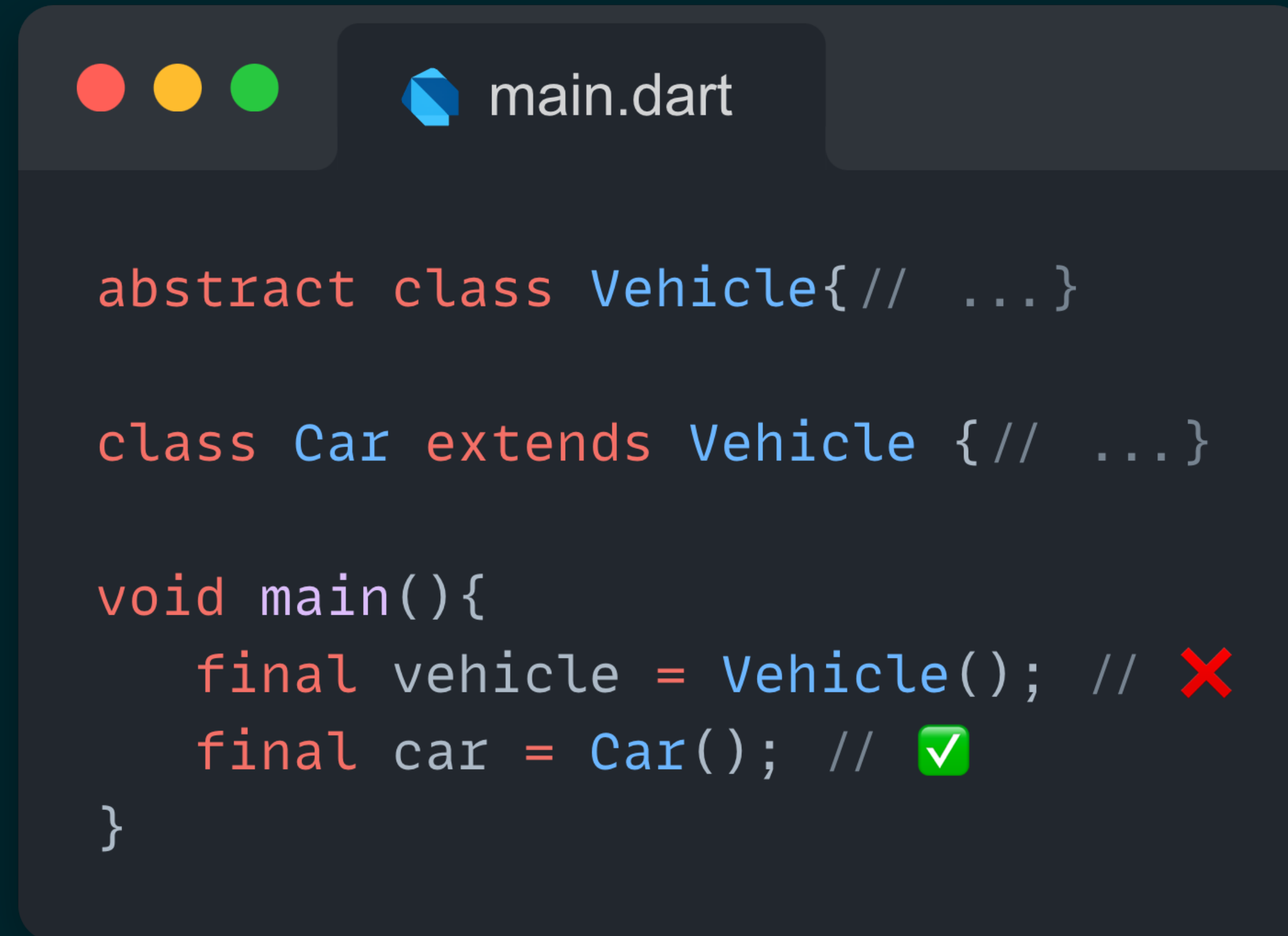
- Siz abstrakt sinfdan obyekt yarata olmaysiz.



main.dart

```
void main() {  
    // Abstract classes can't be instantiated.  
    final pen = Pen();  
}
```

- Ammo uning pastki sinfi yani child class aniq sinf bo'lsa obyekt yaratishingiz mumkin:



```
abstract class Vehicle{// ...}  
  
class Car extends Vehicle { // ...}  
  
void main(){  
    final vehicle = Vehicle(); // ✗  
    final car = Car(); // ✓  
}
```

- Bizning Pen sinfirmizda write() deb nomlangan abstract method mavjud.
- BallPoint sinfini (sub class) Pen klassi (abstract class) bilan kengaytirganingizda yani meros olganingizda, Pen sinfida (abstract class) belgilangan barcha abstract methodlarni amalga oshirishingiz kerak yani qayta yozish kerak bo'ladi.

```
main.dart

abstract class Pen{
    int? rateOfInkFlow;

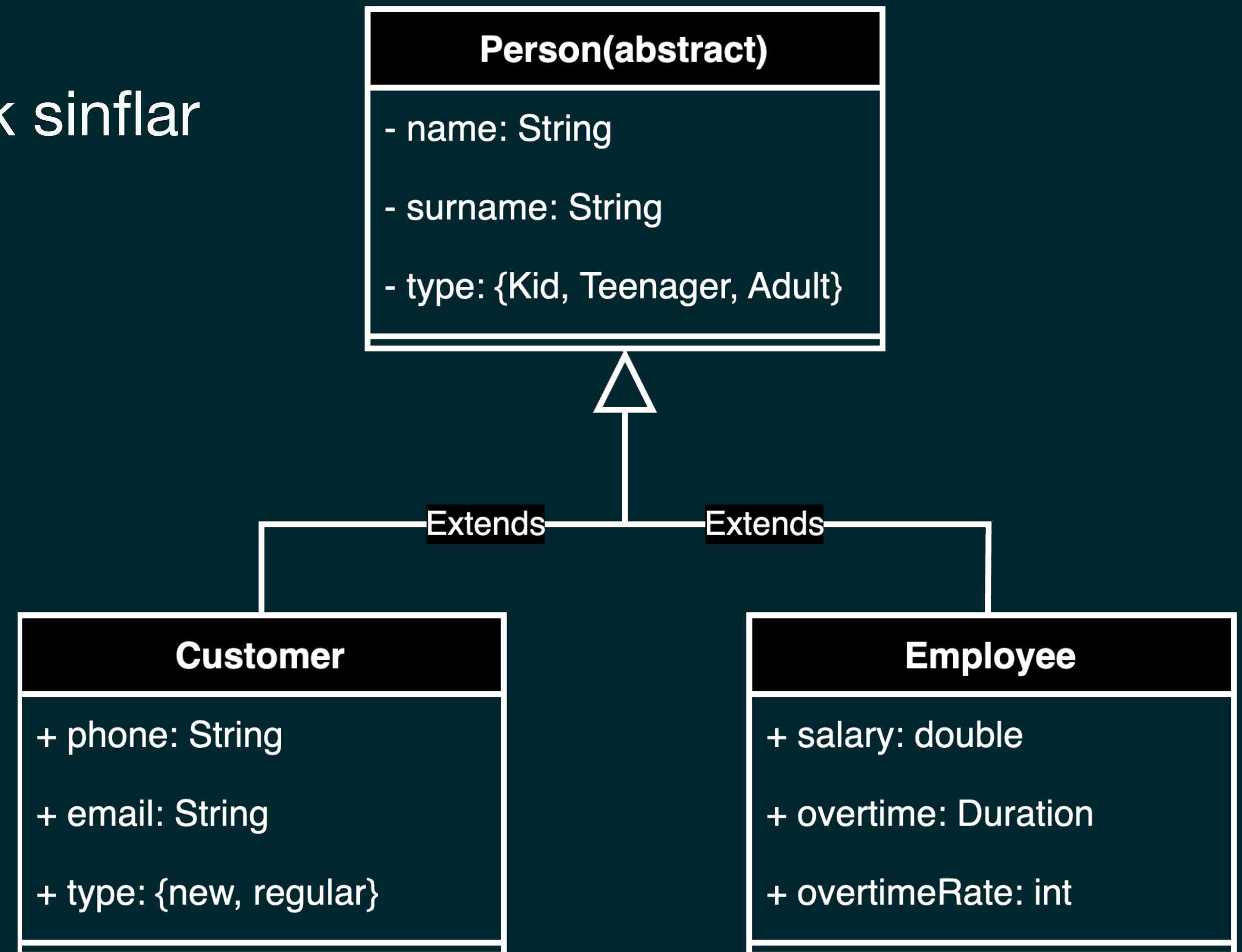
    void write();
}

class BallPoint extends Pen{
    @override
    void write(){
        print("writing...");
    }
}

void main(){
    final ballPoint = BallPoint();
    ballPoint.write();
}
```


Exercise

- Quyidagi UMLda ko'rsatilgandek sinflar ketma ketlikda tuzilsin:



- Agar sinfda kamida bitta abstract method bo'lsa u sinf abstract bo'lishga majbur.
- Keltirilgan misolda abstract klassda ham oddiy methodlar bo'lishi mumkinligi va kerak bo'lganda child klassda override qilib yozish mumkin.
- Ammo shuni yodda tutingki abstract klassda oddiy method bo'lishi abstraktsiyaning umumiy qoidasiga mos emas.



 main.dart

```
abstract class Pen{
    int? rateOfInkFlow;

    void write(){
        print("Writing...");
    }
}

class BallPoint extends Pen{}

void main(){
    final ballPoint = BallPoint();
    ballPoint.write();
}
```

Abstract classes can have Constructor



main.dart

```
abstract class Pen{
  int? rateOfInkFlow;

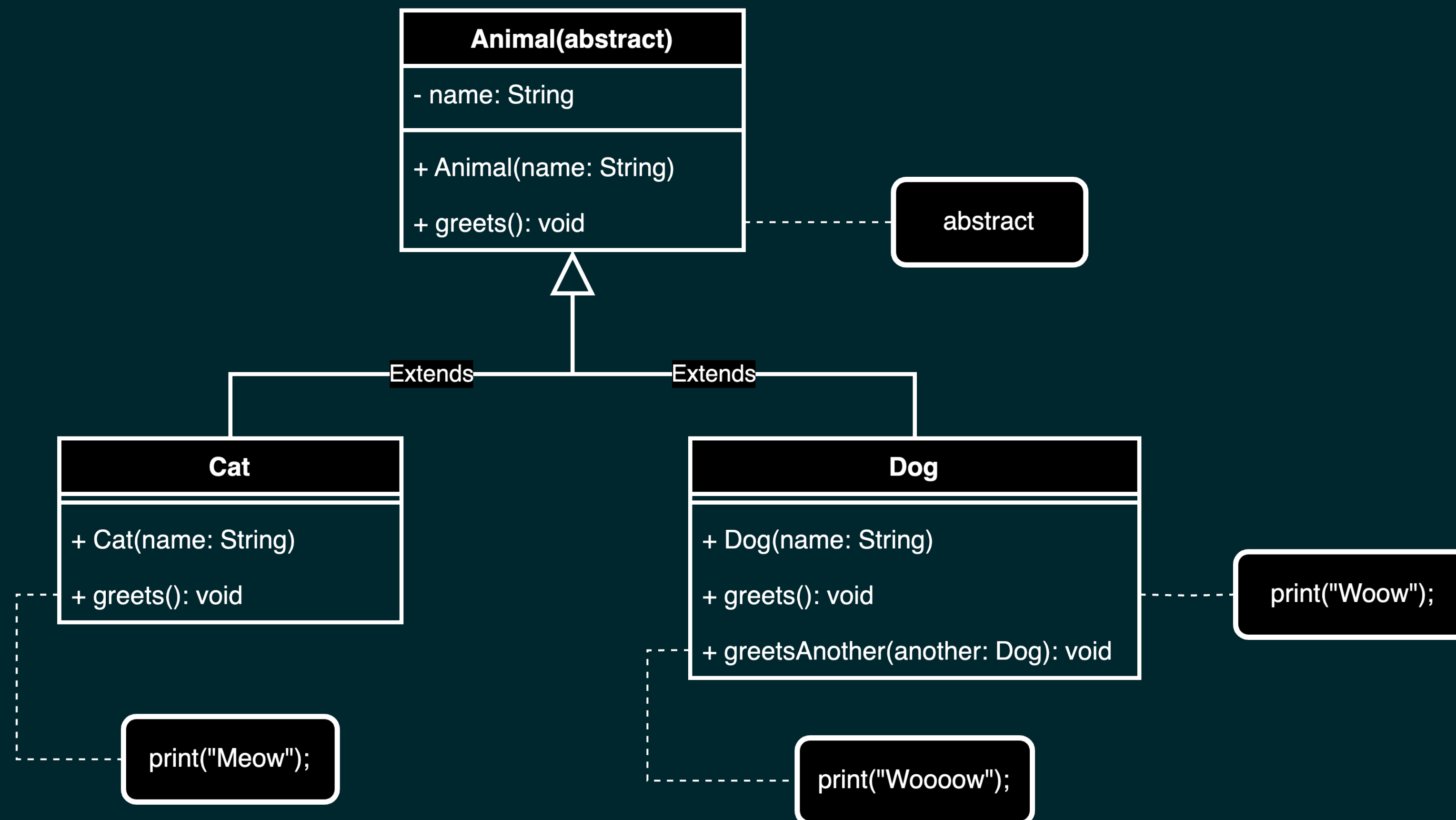
  Pen({required this.rateOfInkFlow});

  void write();
}

class BallPoint extends Pen{
  BallPoint() : super(rateOfInkFlow: 1);
  @override
  void write(){
    print("Writing with BallPoint");
  }
}
```

Exercise

- Quyidagi UMLda ko'rsatilgandek sinflar ketma ketlikda tuzilsin:



Example



main.dart

```
abstract class Animal{
  bool isAlive = true;
  void eat();
  void move();

  @override
  String toString(){
    return "I'm a $runtimeType";
  }
}
```



main.dart

```
class Lion extends Animal{
  @override
  void eat() {
    print("The $runtimeType eats");
  }

  @override
  void move() {
    print("The $runtimeType moves");
  }
}
```



 main.dart

```
class Circle extends Shape {  
  final double radius;  
  Circle({this.radius = 0});  
  
  @override  
  double area() => 3.14 * radius * radius;  
}  
  
class Square extends Shape {  
  final double length;  
  Square({this.length = 0});  
  
  @override  
  double area() => length * length;  
}
```



 main.dart

```
abstract class Shape{  
  double area();  
}  
  
void main(){  
  var circle = Circle(radius: 10);  
  print('The area of the circle is ${circle.area()}');  
  
  var square = Square(length: 10);  
  print('The area of the square is ${square.area()}');  
}
```


- Bu holatda, Rectangle abstract bo'lishi shart, chunki u ota sinfining barcha abstract metodlarini override qilmaydi.
- Agar biz bu ierarxiyani davom ettirsak, aniq(concrete) sinf hosil qilish uchun qolgan abstract a'zolari override qilishimiz lozim bo'ladi.
- Square turi aniq sinfdir, chunki u ota sinfidan area() metodini meros oladi va perimeter() methodini override qiladi.

```
main.dart

abstract class Shape {
  const Shape();

  double area();
  double perimeter();
}

abstract class Rectangle extends Shape {
  final double width;
  final double length;
  const Rectangle({this.width = 0, this.length = 0});

  @override
  double area() => width * length;
}

class Square extends Rectangle {
  const Square(double length)
    : super(length: length, width: length);

  @override
  double perimeter() => 4 * length;
}
```

When to use Abstract Class

- Keng tarqalgan yoki qayta ishlatiladigan sinfni yaratishda.
- Umumiy tayanch(asos) sinfni aniqlash uchun.
- Standart xatti-harakatni ta'minlash uchun.

Summary

Interview Questions

- Abstraktsiya nima?
- Abstract class qanday yaratiladi? Sintaksisi qanday? Misol keltiring!
- Abstract method nima?
- Abstract class dan object yaratsa bo'ladimi?
- Abstract class dan qay holda object yaratish mumkin?
- Abstract class dan meros olganda nega override qilishga majburmiz?
- Agar class da kamida bitta abstract method bo'lsa u class abstract bo'lishga majburmi?

- Abstract class da oddiy metoddan foydalansak bo'ladimi?
- Abstract class dan qachon foydalanamiz?
- Abstract class dan obyekt yaratib bo'lmas ekan, abstract class da konstruktor bo'lishi mumkinmi?

Homework

1. Account nomli abstrakt klass tuzing. Uni 3 ta fieldi bo'lsin password email va name. Account klassdan meros olib BankAccount nomli klass yarating. BankAccount klassida ham qo'shimcha accountNumber va balance nomli ikkita private fieldi bo'lsin. Qachonki passwordni to'g'ri kiritgandagina ularni o'qish va o'zgartirish mumkin bo'lsin, bu display methodi orqali qilinsin. topUp nomli method tuzing va u orqali balansni to'ldirish mumkin bo'lsin. transferMoney nomli method tuzing va u orqali boshqa account hisob raqamiga pul o'tkazish imkoni bo'lsin. Yuqoridagi shartlar asosida realga yaqin ishlaydigan klass va methodlarni tuzishga harakat qiling va har bir methodni void mainda ishlatib ko'rsating.
2. User nomli nomli abstrakt klass tuzing. Uni 3 ta fieldi bo'lsin password email va name. User klassdan meros olib ClientUser(xaridor) va VendorUser(sotuvchi) nomli klass yarating. ClientUserda balance(double) va purchasedItems (sotib olingan narsalar: list) nomli alohida private fieldlari bo'lsin.

VendorUserda esa balance(double) va products(list) nomli listi bo'lsin va private bo'lsin. Product nomli alohida yana klass hosil qiling, bu klassni o'zingiz tuzishga harakat qiling ammo id, name, price va quantity nomli fieldlari bo'lishi kerak. ClientUserda shopping nomli method bo'lsin parametriga prduct va necha dona olishi ni ifodalovchi amount nomli parametr qabul qilsin.

VendarUser da esa toSell nomli method bo'lsin va ham parametriga prduct va necha dona olishi ni ifodalovchi amount nomli parametr qabul qilsin. shopping mehtodi ishlagandagina toSell metodi ishlasin. Bu ikki methodni tuzayotganda ClientUserni balance da olmoqchi bo'lgan productga yetarlicha miqdorda puli bor yo'qligi va bu product ni sotayotganda esa VendorUserda so'ralgan miqdorda product bormi tekshirilsin. Agar hamma shart to'g'ri keladigan bo'lsa olinayotgan product narxicha pul clientdan yechib olinib, vendor balance ga tushurilsin. Shu kabi purchasedItems va product nomli listlardagi productlar ham, soni ham sinxron o'zgartirilsin. Aks holda yani balance va product bor yo'qligi bilan muammo bo'lsa har bir holatga mos xatolik habarini bering.

Yuqoridagi shartlar asosida realga yaqin ishlaydigan klass va methodlarni tuzishga harakat qiling va har bir methodni void mainda ishlatib ko'rsating.

Q&A

Thank you for your time!