Aula 2: Orientação a Objetos com Dart

Professor(a): Virgínia Fernandes Mota
virginia@teiacoltec.org

TECNOLOGIAS DE PROGRAMAÇÃO - DESENVOLVIMENTO DE SISTEMAS

Orientação a Objetos com Dart

- Ao término desta aula, você será capaz de:
  - entender como o Dart trabalha com orientação a objetos.

Classes

- Classe: O modelo para nossos objetos.

```
class Animal{

}

//Testando no DartPad, nosso main pode ficar aqui solitario
void main(){

}
```

Atributos

- Atributos: as características da nossa classe.

```
class Animal{
    String nome;
    int populacao;
    bool quaseExtinto;
}

//Testando no DartPad, nosso main pode ficar aqui  solitario
void main(){

}
```

Métodos

- Métodos: as ações da nossa classe.

```
class Animal{
    String nome;
    int populacao;
    bool quaseExtinto;

    void emiteSom(){
        print("$nome gritou agora");
    }
}

//Testando no DartPad, nosso main pode ficar aqui  solitario
void main(){

}
```

Objetos

- Objetos: a instância é feita com ou sem new;

```
class Animal{
    String nome;
    int populacao;
    bool quaseExtinto;

    void emiteSom(){
        print("$nome gritou agora");
    }
}

//Testando no DartPad, nosso main pode ficar aqui  solitario
void main(){
    Animal lemure = Animal(); //o new e opcional
    lemure.quaseExtinto = true;
    lemure.emiteSom();
}
```

Construtores

- Construtores: o padrão é vazio, senão fazemos o nosso próprio.

```
class Animal{
    String nome;
    int populacao;
    bool quaseExtinto;

    Animal(String nome){//um novo construtor
        this.nome = nome;
    }
    void emiteSom(){
        print("$nome gritou agora");
    }
}
```

Getters e Setters

- O encapsulamento do Dart.

```
class Animal{
    String _nome; //protegendo meu atributo
    int _populacao; //protegendo meu atributo
    bool quaseExtinto;

    // ... outros codigos

    int get populacao(){
        return _populacao;
    }

    set populacao(int populacao){
        _populacao = populacao;
    }
}
```

No Dart não é padrão colocar todos os atributos privados.

Getters e Setters

- O encapsulamento do Dart.

```
class Animal{
    String _nome; //protegendo meu atributo
    int _populacao; //protegendo meu atributo
    bool quaseExtinto;

    //... outros codigos

    int get populacao => _populacao; //outra forma de get

    set populacao(int populacao){
        _populacao = populacao;
    }
}
```

Herança

- Herança simples.

```
class Animal{
 //...
}

class Cachorro extends Animal{
    int coeficienteFofura ;
}

class Lemure extends Animal{
    int coeficienteMemes;
}
```

Herança

- Herança simples.

```
class Animal{
// ...
}

class Cachorro extends Animal{
    int coeficienteFofura ;

    //criando um construtor e usando o da classe mae
    Cachorro(String nome, int coeficienteFofura ) : super(nome);
}
```

Reescrita de métodos

```
class Animal{
// ...

    void emiteSom(){
        print("$nome gritou agora");
    }
}

class Cachorro extends Animal{
    int coeficienteFofura ;

    Cachorro(String nome, int coeficienteFofura ) : super(nome);

    @override
    void emiteSom(){
        print("Au au au, au");
    }
}
```

Static, Final, Const

- *static*: modificador para que o atributo seja da classe e não do objeto.
- *const*: modificador para que o atributo seja constante, não altera nunca (em tempo de compilação).
- *final*: modificador para que o atributo seja constante (em tempo de execução).

Classes Abstratas

- No nosso exemplo, a classe Animal serve como generalização mas não necessariamente precisa ser instanciada.
- Assim, ela pode ser uma Classe Abstrata.

```
abstract class Animal{


void emiteSom(); //se meu metodo nao tem corpo, ele se torna um metodo abstrato.

}
```

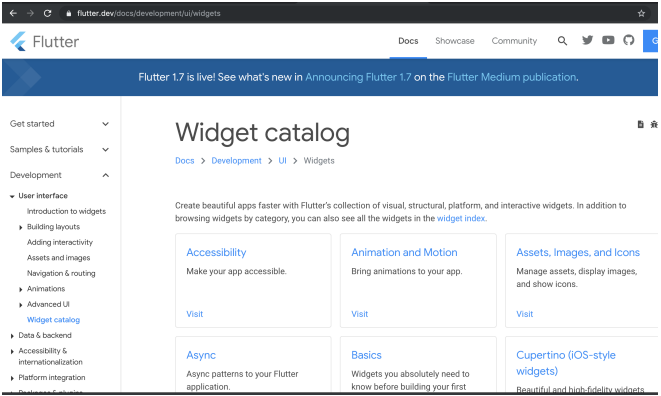
Listas

```
void main(){
    List<String> turmas = ["303", "203A", "203B"];
    turmas.add("ModIII");
    print(turmas);

    turmas.removeAt(0);
    turmas.insert(1, "106");
    bool tem = turmas.contains("101");

    List<Cachorro> pets = List();
    pets.add(Cachorro("toto", 8000));
    pets.add(Cachorro("judite", 10));
    for (Cachorro c in pets){
        print(c); //basta implementar o toString
    }

}
```

Dart e Flutter

- Vimos então que Dart é uma linguagem orientada objetos muito próxima ao Java e C#.
- Vamos então começar a parte diferente: dispositivos móveis.

Widgets



`flutter.io/widgets`

O que são Widgets?

Material Design

- `https://material.io/design/`
- Material Design é uma linguagem de design desenvolvida pela Google.
- Faz um uso mais liberal de layouts baseados em grids, animações e transições responsivas, preenchimentos, e efeitos de profundidade como luzes e sombras.

Na próxima aula...

Fazendo um aplicativo com Flutter