Sistemas Embarcados

Aula 3 - Continuação de Orientação a Objeto e Introdução ao Android Studio

Herança

Herança é uma maneira de aproveitar o código descrito em outras classe e expandir suas funcionalidades. Por exemplo, pense que temos uma classe que representa o objeto Televisão. Esta classe já possui um método `turnOn()` com a funcionalidade de ligar a Televisão.

```
class Television {
  void turnOn() {
    _illuminateDisplay();
    _activateIrSensor();
  }
  // ...
}
```

Imagine agora que desejamos expandir estas funcionalidades para uma Smart TV, que possui as mesmas funcionalidades da TV normal, porém implementa mais alguns métodos para tratar a conectividade e atualizar os aplicativos instalados.

```
class SmartTelevision extends Television {
  void turnOn() {
    super.turnOn();
    _bootNetworkInterface();
    _initializeMemory();
    _upgradeApps();
  }
  // ...
}
```

Polimorfismo

O polimorfismo é uma maneira de implementarmos um memso método em classes diferentes. Neste caso, os métodos teram funcionalidades diferentes, que depende do objeto que a classe representa. Vamos ver um exemplo:

```
abstract class Sensor{
  var value = 0;

  void read(){
    value += 1;
    print(value);
  }
}
```

Polimorfismo

```
class DHT22 extends Sensor{
  int DHTSensor = 22
  int DHTPin = 9

  @override
  void read() {
    var humidityAndTemperature = []
    humidityAndTemperature = AdafruitDHT.readRetry(this
    if (humidity =≠ None and temperature =≠ None){
        return humidityAndTemperature
    } else {
        print("Falha ao receber os dados do sensor DHT2
    }
  }
}
```

```
class PIR extends Sensor{
  int PIRPin = 9
  int LAST_READ_TIME = 0
  int INTERVAL = 5000 // intervalo entre medições em mi

  @override
  void read() {
    dataToSend = []
    if (
        DateTime.now().millisecondsSinceEpoch - this.LA
        and GPIO.input(this.PIN)
    ){
        dataToSend.add(this.formatData("MOVIMENTO", 1))
        self.LAST_READ_TIME = DateTime.now().millisecon
    }
    return dataToSend.length > 0 ? dataToSend : Null
  }
}
```

Polimorfismo

Como é implementado o polimorfismo em Dart?

```
void main(){
  DHT22 dht = DHT22();
  PIR pir = PIR();
  var sensors = [dht, pir];

  for (final sensor in sensors){
    print( sensor.read() );
  }
}
```

Encapsulamento

O encapsulamnto é uma maneira de proteger as propriedades de objeto de forma que não aconteçam efeitos colaterais imprevisíveis durante a execução do código. Por exemplo, considere a classe PIR apresentada anteriomente, nós sabemos que o intervalo entre medições foi definido para que o banco de dados não fique cheio de informações repetidas em um curto período de tempo. Desta forma, devemos proteger o acesso às propriedades `INTERVAL` e `LAST_READ_TIME`, e torná-las privada:

```
class PIR extends Sensor{
  int PIRPin = 9
  int _LAST_READ_TIME = 0
  int _INTERVAL = 5000 // intervalo entre medições em milisegundos

  void read() {
    dataToSend = []
    if (
        DateTime.now().millisecondsSinceEpoch - this._LAST_READ_TIME > this._INTERVAL
        and GPIO.input(this.PIN)
    ){
        dataToSend.add(this.formatData("MOVIMENTO", 1))
        self._LAST_READ_TIME = DateTime.now().millisecondsSinceEpoch
    }
    return dataToSend.length > 0 ? dataToSend : Null
  }
}
```

Exercício

Vamos aplicar os conceitos anteriores ao exercício de figuras geométricas.

https://replit.com/@paulormnas/figuras-geometricas#main.dart

Android Studio

IDE desenvolvida pela Jet Brains, em parceria com a Google, para impulsionar o desenvolvimento de aplicações para smartphones que rodam o sistema operacional Android. Esta IDE facilita muito o processo de desenvolvimento devido às diversas ferramentas, como debug, hot reload, emulador de smartphones, por exemplo. Contudo, nos últimos anos foi adicionado o suporte à linguagem Dart e ao framework Flutter, permitindo desenvolver apps também para a plataforma iOS.

A partir daqui vamos fazer a configuração no PC que vocês estão utilizando.