Desenvolvimento Mobile 1
Aula 08

Prof. Me Daniel Vieira

SENAI

Agenda

1- Classe abstrata
2- Get e Setters
3-Poliformismo
4 -Exercícios

Classe abstrata

```
class Alimento{
  String nome;
  double peso;
  String sabor;
  Alimento(this.nome, this.peso,this.sabor);
}
```

```
class Fruta extends Alimento implements Descascar{
  bool? isMadura;
  Fruta(String nome, double peso, String sabor):super(nome,peso,sabor);
  void frutaMadura(isMadura)
  {
    if(isMadura==true)
    {
      print("A fruta $nome está madura");
    }
    else{
          print("A fruta $nome não está madura");
    }
  }
  @override
  void separarTalheres(){
    print("Separando talheres");
  }
  @override
  void pegarvasilha()
  {
    print("Pegando vasilha");
  };
  @override
  void comer(){
    print("Comendo a fruta $nome");
  }
}
```

Classe abstrata

```
class Uva extends Alimento implements Descascar{
 Uva(String nome, double peso, String sabor):super(nome,peso,sabor);

  @override
  void separarTalheres()
  {
    print("Não é preciso separar talheres para descascar a fruta $nome");
  }

  @override
 void pegarvasilha()
 {
  print("Não é preciso pegar vasilha para comer $nome");
 }
@override
 void comer()
 {
    print("Comendo fruta $nome");

 }
```

Classe abstrata

- **O que são Classes Abstratas:**
    - As Classes abstratas (conhecidas em outras linguagens como Interface) são como contratos pré-definidos. Elas são muito usadas para dar um caminho definido para todas as classes que a implementam. Ao criar uma classe abstrata, fazemos os seus métodos sem nenhuma ação, pois dessa forma, as ações são definidas apenas por aqueles que implementam a classe abstrata criada.
- **Polimorfismo:**
    - Agora, já sabemos extender uma classe (Mãe/Filha) e já sabemos implementar uma classe abstrata (Contrato), e começamos a notar que nem sempre os métodos herdados podem ser úteis a todo momento. Em alguns casos, precisamos alterar esses métodos sem comprometer outras classes, e para isso usamos a `Sobrescrita` com o comando `@override`. O Polimorfismo nada mais é que a habilidade das nossas classes de alterar um método recebido por herança.

Get e Setter

```
class Conta {
  double _saque = 0;
  double _saldo = 500;
  // Get
  //  Setter
  double get saque {
    return this._saque;
  }
double get saldo {
    return (this._saldo - this.saque);
  }
set saque(double valor) {
    if (valor > 0 && valor < 500) {
      this._saque = valor;
    }
  }
}
```

```
void main() {
  Conta contaRocky = Conta();
  contaRocky.saque = 100;

  print("Valor do saque R\$ ${contaRocky.saque}");
  print("Valor do saldo R\$ ${contaRocky.saldo}");
}
```

Exercícios

1) Criar uma classe mãe chamada animal com os atributos:
String nome, int idade, String cor, String raça,
Criar uma classe filha com o tipo de animal pássaro, cachorro, tigre, peixe
Atributos: peso, métodos acordou, dormiu.
Criar uma classe abstrata denominada alimentar com três métodos:
Separar ingredientes
Pegar tigela
Preparar comida
Essa classe abstrata deve ser implementada na classe filha.

Exercícios

2) Criar uma classe abstrata Máquina industrial com os seguintes métodos:
Nome - Nome da máquina
Potência da máquina
status - booleano
Métodos abstratos
ligar() - Um método abstrato que define como a máquina é ligada
desligar () - Método abstrato que define como a máquina é desligada
Criar duas subclasses concretas de máquina industrial Prensa e Robô solda.
Prensa deve ter um atributo adicional chamado pressão em toneladas e os métodos ligar e desligar deve exibir mensagens adequadas

Exercícios

Robô solda deve ter um atributo chamado tipo de solda(String) para especificar o tipo de solda que realiza, os métodos ligar e desligar devem exibir mensagens adequadas.

Criar uma classe chamada transportador que herda de MáquinaIndustrial, esta classe deve ter um atributo adicional chamado velocidade (float)m/s.
Criar instâncias das classes Prensa, RoboSolda, Transportador. Ligar e desligar cada uma delas para testar seus métodos.
Criar um lista chamada máquinas e adicionar todas as instâncias das máquinas anteriormente a essa lista. Em seguida percorra a lista e para cada máquina exibir nome, potência e status.
Crie uma classe FornecedorDeEnergia com um método fornecerEnergia(maquina: MaquinaIndustrial) que recebe uma instância de MaquinaIndustrial e liga a máquina se houver energia suficiente (suponha que a energia seja um recurso limitado). Implemente esse método de forma que ele só ligue a máquina se houver energia suficiente e exiba uma mensagem informando se a máquina foi ligada ou não.

Exercícios

3) Criar uma classe chamada produto com nome, preço e quantidade e utilizando o comando get passar o nome para o produto, o preço e a quantidade e um método para exibir as informações do produto

4) Criar classe Pessoa com getters e setters para o nome e idade da pessoa. Os setters permitem configurar esses atributos com verificações de validade. A classe também possui um método mostrar Informações para exibir as informações da pessoa.

5) Criar uma classe abstrata chamada automóveis com nome, cor, ano como atributos.
Criar uma classe chamada carro herdando características da classe abstrata automóveis.
Criar métodos abstratos na classe abstrata como colocar o cinto,ligar o carro, desligar o carro e dirigir.
Criar uma classe concreta carro implementando os métodos abstratos exibindo mensagens adequadas.

Exercícios

https://docs.google.com/forms/d/159tsKuzLJWmbKr7HNf9V-D5MJf77wlUBodUu6HU54Vc/edit

Obrigado!

Prof. Me Daniel Vieira

Email: danielvieira2006@gmail.com

Linkedin: Daniel Vieira

Instagram: Prof daniel.vieira95