prova1-poo

April 8, 2023

[ ]:
```python
class Corrida ():
    "Classe contém valores relacionados a corrida (forma de pagamento, tempo,␣
 ↪distancia) e métodos de cálculo do valor de cada corrida"

    def __init__(self, forma_pagamento: str, tempo_minutos: float = 0.00,␣
 ↪distancia_km: float = 0.00) -> None:
        self.forma_pagamento = forma_pagamento
        self.tempo_minutos = tempo_minutos
        self.distancia_km = distancia_km

    def valor_base(self) -> float:
        "Apura o valor base de cada corrida, antes dos descontos"

        preco_km = float(1.60)  # valor cobrado por km rodado

        preco_tempo_min = float(0.60)  # valor cobrado por hora rodada

        #calculo do valor final
        preco_base = float((preco_km * (abs(self.distancia_km)*2)) +
                           ((abs(self.tempo_minutos))/2)*preco_tempo_min)

        return preco_base

    def percentual_desconto(self) -> float:
        "Determina o percentual do desconto de acordo com metodo de pagamento"

        if self.forma_pagamento == "dinheiro":
            percentual_desconto = 0
            return percentual_desconto

        elif self.forma_pagamento == "cartao":
            percentual_desconto = 0.05
            return percentual_desconto

        else:
            percentual_desconto = 0.1
            return percentual_desconto
```

```
    def valor_final(self) -> float:
        "Metodo que apura o valor final (líquido de eventual desconto)da␣
↪corrida"

        cobrar_cheio = self.valor_base()

        valor_desconto = cobrar_cheio * self.percentual_desconto()

        valor_cobrar = cobrar_cheio - valor_desconto

        return valor_cobrar
```

[ ]:
```
class Viagem():
    "Classe que encapsula dados que são relacionados a cada corrida"

    def __init__(self, distancia_km, valor_viagem, origem: str = None, destino:␣
↪str = None,) -> None:
        self.origem = origem
        self.destino = destino
        self.distancia_km = distancia_km
        self.valor_viagem = valor_viagem
```

[ ]:
```
class Pagamento():
    "Classe com atributos relacionados a metodo e valores dos pagamentos"

    def __init__(self, forma_pagamento: str, valor_pago: float = 0) -> None:
        self.forma_pagamento = forma_pagamento
        self.valor_pago = valor_pago
```

[ ]:
```
import random

class Motorista():
    "Classe com atributos e metodos pertinentes ao objeto motorista"

    registro_corridas = {} # dict atributo de classe para armazenar os dados␣
↪das corridas

    def __init__(self, nome: str, dados_corridas: list) -> None:
        self.nome = nome
        self.dados_corridas = dados_corridas  # lista que contem objetos de␣
↪outras classes

    def adicionar_corrida(self) -> dict:
        "Método que registra em um dicionário os dados pertinentes a uma␣
↪corrida"
```

```
        id_corrida = random.randint(1, 1000)

        self.registro_corridas[id_corrida] = {'Motorista': self.nome,␣
↪'Distancia_km': self.dados_corridas[0].distancia_km,
                                              'Metodo_Pagamento': self.
↪dados_corridas[0].forma_pagamento,
                                              'Tempo_Corrida': self.
↪dados_corridas[0].tempo_minutos,
                                              "Origem": self.dados_corridas[1].
↪origem,
                                              'Destino': self.dados_corridas[1].
↪destino, 'Valor_Recebido': self.dados_corridas[2].valor_pago}

        return ('Registro inserido com sucesso.')

    def total_motorista(self, nome_motorista) -> float:
        #Método para levantar os valores totais recebidos por um motorista
        soma = 0
        for valor in self.registro_corridas.values():
            if valor['Motorista'] == nome_motorista:
                soma += valor['Valor_Recebido']
        return soma
```

[ ]:
```
def valida_metodo():
    "validar o metodo de pagamento"
    while True:
        forma = input("Insira a forma de pagamento: dinheiro, cartao, pix")
        try:
            if forma == 'dinheiro' or forma == 'cartao' or forma == 'pix':
                return forma
        except:
            valida_metodo()

def validar_minutos():
    "validar entrada com minutos rodados"

    while True:
        valor = input("Insira a duracao em minutos: ")
        try:
            valor = float(valor)
            if valor > 0:
                return valor
        except:
            validar_minutos()

def validar_km():
```

```
    "validar entrada com km rodado"

    while True:
        valor = input("Insira a kilometragem rodada: ")
        try:
            valor = float(valor)
            if valor > 0:
                return valor
        except:
            validar_km()
```

[ ]:
```
# objeto da classe Corrida

forma = valida_metodo()
minutos = validar_minutos()
kmrodado = validar_km()
corrida = Corrida(forma, minutos, kmrodado)

# objeto da classe Viagem
origem = str(input("Insira a rua de partida: "))
destino = str(input("Insira a rua do destino: "))
viagem = Viagem(corrida.distancia_km, corrida.valor_final(),
                origem, destino)

# objeto da classe Pagamento
pagamento = Pagamento(corrida.forma_pagamento, corrida.valor_final())

# objeto da classe Motorista
profissional = str(input("Insira o nome do motorista: "))
motorista = Motorista(profissional, [corrida, viagem, pagamento])

# chama metodo que adiciona corrida no dicionario
registro = motorista.adicionar_corrida()
print(registro)
```

[ ]:
```
import json

consulta_base = motorista.registro_corridas
print(json.dumps(consulta_base, indent=4))
```


```
{
    "217": {
        "Motorista": "Zoe",
        "Distancia_km": 25.0,
        "Metodo_Pagamento": "dinheiro",
        "Tempo_Corrida": 32.0,
        "Origem": "rua o",
```

```
        "Destino": "rua a",
        "Valor_Recebido": 89.6
    },
    "560": {
        "Motorista": "Zoe",
        "Distancia_km": 23.0,
        "Metodo_Pagamento": "pix",
        "Tempo_Corrida": 31.6,
        "Origem": "pp",
        "Destino": "qq",
        "Valor_Recebido": 74.772
    },
    "744": {
        "Motorista": "Paulo",
        "Distancia_km": 5.0,
        "Metodo_Pagamento": "cartao",
        "Tempo_Corrida": 12.0,
        "Origem": "rua dos cachorros",
        "Destino": "rua dos bezerros",
        "Valor_Recebido": 18.62
    }
}
```

[ ]:
```
consulta = 'Zoe'
total = motorista.total_motorista(consulta)
print(f'O motorista {consulta} recebeu o valor total liquido de R$ {total:.2f}.
  ↪')
```

```
O motorista Zoe recebeu o valor total liquido de R$ 164.37.
```