Entrega 2 - Implementação da Mensagem (Envelope) e dos Argumentos de Entrada (IN) e Saída (OUT).

Entregar o código fonte das Classes que representam a mensagem (req/resp) e os dados que serão enviados e retornados pelos métodos remotos:

Integrantes: Larissa Matos e Paula Feitosa

Para este trabalho, a equipe decidiu utilizar JSON. O trabalho é sobre o joguinho Tamagotchi. Cada uma das classes a partir da segunda a seguir, vai herdar a classe “Message” para que a serialização seja feita. Para que isso ocorra, os atributos estão sendo agrupados em um dicionário.

class Message: def __init__(self, type, id, obf_reference, method_id, arguments):
self.type = type
self.id = id
self.obf_reference = obf_reference
self.method_id = method_id
self.arguments = json.dumps(arguments)

```
class FeedPetRequest(Message):
    def __init__(self, food, drink):
        super().__init__(1, None, None, "feed_pet", {"food": food, "drink": drink})

class FeedPetResponse(Message):
    def __init__(self, success, message):
        super().__init__(2, None, None, "feed_pet", {"success": success, "message":
message})

class PlayWithPetRequest(Message):
    def __init__(self, game, participate):
        super().__init__(1, None, None, "play_with_pet", {"game": game, "participate":
participate})

class PlayWithPetResponse(Message):
    def __init__(self, success, message, diamonds_earned):
        super().__init__(2, None, None, "play_with_pet", {"success": success, "message":
message, "diamonds_earned": diamonds_earned})

class ShowerPetRequest(Message):
    def __init__(self, water):
        super().__init__(1, None, None, "shower_pet", {"water": water})

class ShowerPetResponse(Message):
    def __init__(self, success, message):
```

```
        super().__init__(2, None, None, "shower_pet", {"success": success, "message":
message})


class SleepPetRequest(Message):
    def __init__(self, hours, dream):
        super().__init__(1, None, None, "sleep_pet", {"hours": hours, "dream": dream})


class SleepPetResponse(Message):
    def __init__(self, success, message, age_increment, diamonds_earned):
        super().__init__(2, None, None, "sleep_pet", {"success": success, "message":
message, "age_increment": age_increment, "diamonds_earned": diamonds_earned})


class StatusRequest(Message):
    def __init__(self):
        super().__init__(1, None, None, "status", {})

class StatusResponse(Message):
    def __init__(self, clean, energy, hungry, age, diamonds):
        super().__init__(2, None, None, "status", {"clean": clean, "energy": energy,
"hungry": hungry, "age": age, "diamonds": diamonds})
```

Diagrama UML do código:
(obs.: nós não somos muito familiarizadas com a criação de diagramas de classe)