Code Challenge: Autorizador

Você deverá implementar uma aplicação que autoriza transações para uma conta específica seguindo uma série de regras predefinidas.

Por favor leia as instruções abaixo e sinta-se à vontade para fazer perguntas caso ache necessário.

⚠️ **IMPORTANTE:** Por favor remova toda informação que possa lhe identificar nos arquivos do desafio antes de enviar a solução. Atenção especial para os seguintes pontos:

- Arquivos da solução como código, testes, namespaces, binários, comentários, e nomes dos arquivos;
- Comentários automáticos que seu editor de código pode ter adicionado aos arquivos;
- Documentação do código como annotations, metadata, e README.MD;
- Informações de autoria do código e configuração do versionador de código.

Se você planeja utilizar git como sistema de controle de versões, execute o seguinte comando na raíz do repositório para exportar a solução anonimizada:

```
git archive --format=zip --output=./authorizer.zip HEAD
```

Preparando seu desafio para envio

Sua solução deve conter um arquivo de README com:

- Uma explicação sobre as decisões técnicas e arquiteturais do seu desafio;
- Uma justificativa para o uso de frameworks ou bibliotecas (caso sejam usadas);
- Instruções sobre como compilar e executar o projeto;
- Notas adicionais que você considere importantes para a avaliação.

O processo de build e execução da aplicação deve ser possível num sistema operacional Unix ou Mac. Builds conteinerizadas são bem vindas.

Exemplo de uso do Autorizador

Como o programa deve funcionar?

Seu programa receberá como entrada linhas em formato `json` na entrada padrão (`stdin`) e deve fornecer uma saída em formato `json` para cada uma das entradas, imagine isso como um fluxo de eventos chegando ao autorizador.

Como o programa deve ser executado?

Dado um arquivo chamado `operations` que contém diversas linhas descrevendo operações no formato `json`:

```
$ cat operations
{"account": {"active-card": true, "available-limit": 100}}
{"transaction": {"merchant": "Burger King", "amount": 20, "time": "2019-02-
13T10:00:00.000Z"}}
{"transaction": {"merchant": "Habbib's", "amount": 90, "time": "2019-02-
13T11:00:00.000Z"}}
{"transaction": {"merchant": "McDonald's", "amount": 30, "time": "2019-02-
13T12:00:00.000Z"}}
```

A aplicação deve ser capaz de receber o conteúdo do arquivo via `stdin`, e para cada operação processada fornecer um output adequado de acordo com a lógica de negócio:

```
$ authorize < operations

{"account": {"active-card": true, "available-limit": 100}, "violations": []}
{"account": {"active-card": true, "available-limit": 80}, "violations": []}
{"account": {"active-card": true, "available-limit": 80}, "violations": ["insufficient-
limit"]}
{"account": {"active-card": true, "available-limit": 50}, "violations": []}
```

Operações do Autorizador

O programa deve lidar com dois tipos de operações, decidindo qual delas executar de acordo com a linha que estiver sendo processada:

1. Criação da conta
2. Autorização de uma transação na conta

Para simplificar o programa, você pode assumir que:

- Todos os valores monetários são inteiros positivos, portanto é uma moeda sem centavos;
- As transações na conta chegarão no autorizador em ordem cronológica.

---

1. Criação da conta

Entrada

Cria a conta com os atributos `available-limit` (limite disponível) e `active-card` (cartão ativo). Para simplificar o programa, vamos assumir que o Autorizador lidará com apenas uma conta.

Saída

O estado atual da conta criada junto de quaisquer violações da lógica de negócios. Se não houverem violações no processamento da operação, o campo `violations` deve retornar um vetor vazio `[]`.

Violações da lógica de negócios

Uma vez criada, a conta não deve ser atualizada ou recriada. Se o aplicativo receber uma segunda operação de criação de conta, ele deve retornar a seguinte violação: `account-already-initialized` (Conta já foi inicializada).

Exemplos

Criando uma conta com sucesso

Criando uma conta com cartão inativo (`active-card: false`) e limite disponível de 750 (`available-limit: 750`):

```
# Input
    {"account": {"active-card": false, "available-limit": 750}}

# Output
    {"account": {"active-card": false, "available-limit": 750}, "violations": []}
```

Criando uma conta que viola a lógica do Autorizador

Dado que há uma conta com cartão ativo (`active-card: true`) e limite disponível de 175 (`available-limit: 175`), tenta criar outra conta:

```
# Input
    {"account": {"active-card": true, "available-limit": 175}}
    {"account": {"active-card": true, "available-limit": 350}}

# Output
    {"account": {"active-card": true, "available-limit": 175}, "violations": []}
    {"account": {"active-card": true, "available-limit": 175}, "violations": ["account-already-initialized"]}
```

---

2. Autorização de transação

Entrada

Tenta autorizar uma transação para um determinado `merchant` (comerciante), `amount` (valor da transação) e `time` (horário da transação) de acordo com o estado da conta criada e **as últimas transações que foram autorizadas**.

Saída

O estado atual da conta junto de quaisquer violações da lógica de negócios. Se não houverem violações no processamento da operação, o campo `violations` deve retornar um vetor vazio `[]`.

Violações da lógica de negócios

Você deve implementar as seguintes regras, tendo em mente que **novas regras aparecerão no futuro**:

- Nenhuma transação deve ser aceita sem que a conta tenha sido inicializada: `account-not-initialized`
- Nenhuma transação deve ser aceita quando o cartão não estiver ativo: `card-not-active`
- O valor da transação não deve exceder o limite disponível: `insufficient-limit`
- Não deve haver mais que 3 transações de qualquer comerciante em um intervalo de 2 minutos: `high-frequency-small-interval`
- Não deve haver mais que 1 transação similar (mesmo valor e comerciante) no intervalo de 2 minutos: `double-transaction`

Exemplos

Processando uma transação com sucesso

Dado que há uma conta com cartão ativo (`active-card: true`) e limite disponível de 100 (`available-limit: 100`):

```
# Input
    {"account": {"active-card": true, "available-limit": 100}}
    {"transaction": {"merchant": "Burger King", "amount": 20, "time": "2019-02-13T11:00:00.000Z"}}

# Output
    {"account": {"active-card": true, "available-limit": 100},"violations": []}
    {"account": {"active-card": true, "available-limit": 80},"violations": []}
```

Processando uma transação para uma conta que não foi inicializada

Quando uma operação de transação é processada e não há uma conta criada previamente, o Autorizador deve retornar a violação `account-not-initialized`:

```
# Input
    {"transaction": {"merchant": "Uber Eats", "amount": 25, "time": "2020-12-01T11:07:00.000Z"}}
    {"account": {"active-card": true, "available-limit": 225}}
    {"transaction": {"merchant": "Uber Eats", "amount": 25, "time": "2020-12-01T11:07:00.000Z"}}
# Output
    {"account": {}, "violations": ["account-not-initialized"]}
    {"account": {"active-card": true, "available-limit": 225}, "violations": []}
    {"account": {"active-card": true, "available-limit": 200}, "violations": []}
```

Processando uma transação que viola a lógica do Autorizador

Dado que há uma conta com cartão ativo (`active-card: true`), limite disponível de 100 (`available-limit: 100`), e 3 transações ocorreram com sucesso nos úlltimos 2 minutos, o Autorizador deve rejeitar uma nova tentativa de transação e retornar a violação `high-frequency-small-interval`:

```
# Input
    {"account": {"active-card": true, "available-limit": 100}}
    {"transaction": {"merchant": "Burger King", "amount": 20, "time": "2019-02-13T11:00:00.000Z"}}
    { "transaction": {"merchant": "Habbib's", "amount": 20, "time": "2019-02-13T11:00:01.000Z"}}
    {"transaction": {"merchant": "McDonald's", "amount": 20, "time": "2019-02-13T11:01:01.000Z"}}
    {"transaction": {"merchant": "Subway", "amount": 20, "time": "2019-02-13T11:01:31.000Z"}}

# Output
    {"account": {"active-card": true, "available-limit": 100},"violations": []}
    {"account": {"active-card": true, "available-limit": 80},"violations": []}
    {"account": {"active-card": true, "available-limit": 60},"violations": []}
    {"account": {"active-card": true, "available-limit": 40},"violations": []}
    {"account": {"active-card": true, "available-limit": 40},"violations": ["high-frequency-small-interval"]}
```

Estado da aplicação

O programa **não deve depender** de nenhum banco de dados externo, e o estado interno da aplicação deve ser gerenciado em memória explicitamente por alguma estrutura que achar adequada. O estado da aplicação deve estar vazio sempre que a aplicação for inicializada.

As operações do Autorizador que houverem violações não devem ser salvas no estado interno da aplicação. Por exemplo, as operações a seguir não devem acionar a violação `high-frequency-small-interval`:

```
# Input
    {"account": {"active-card": true, "available-limit": 1000}}
    {"transaction": {"merchant": "Vivara", "amount": 1250, "time": "2019-02-
13T11:00:00.000Z"}}
    {"transaction": {"merchant": "Samsung", "amount": 2500, "time": "2019-02-
13T11:00:01.000Z"}}
    {"transaction": {"merchant": "Nike", "amount": 800, "time": "2019-02-
13T11:01:01.000Z"}}
    {"transaction": {"merchant": "Uber", "amount": 80, "time": "2019-02-
13T11:01:31.000Z"}}

# Output
    {"account": {"active-card": true, "available-limit": 1000},"violations": []}
    {"account": {"active-card": true, "available-limit": 1000},"violations":
["insufficient-limit"]}
    {"account": {"active-card": true, "available-limit": 1000},"violations":
["insufficient-limit"]}
    {"account": {"active-card": true, "available-limit": 200},"violations": []}
    {"account": {"active-card": true, "available-limit": 120},"violations": []}
```

Lidando com erros

- Por favor assuma que não ocorrerão erros na conversão do `json` de entrada. Na avaliação da sua solução nós não vamos utilizar entradas que contenham erros, estejam mal formatadas, ou que quebrem o contrato.
- As violações das regras de negócios **não são consideradas erros** porque se espera que elas ocorram e devem ser listadas no campo `violations` (violações) das saídas conforme descrito nos esquemas de `output` (saída) dos exemplos. Isso significa que a execução do programa deve continuar normalmente após qualquer tipo de violação.

Nossas Expectativas

Nós no Nubank valorizamos as seguintes qualidades:

- **Simplicidade**: espera-se da solução um projeto pequeno e de fácil entendimento;
- **Elegância**: espera-se da solução facilidade de manutenção, uma separação clara das responsabilidades e uma estrutura de código bem organizada;
- **Operacional**: espera-se da solução a resolução do problema, seus casos de borda ou extremos e a capacidade de extensão para futuras decisões de design.

Desta forma, procuraremos avaliar:

- Uso adequado de trasparência referencial quando aplicável;
- Testes de unidade e integração de qualidade;
- Documentação onde for necessário;
- Instruções sobre como executar o código.

Por último, porém não menos importante:

- Você pode utilizar bibliotecas de código aberto (open source) que acredite serem adequadas para

ajudar na solução do desafio, por exemplo analisadores de json; Por favor tente limitar o uso de frameworks e boilerplate code desnecessários.
- O desafio espera uma aplicação de linhas de comando **independente**; Por favor evite adicionar infraestrutura desnecessária e/ou dependências externas. É esperado que você seja capaz de identificar as ferramentas necessárias para resolver o problema apresentado sem adicionar camadas extras de complexidade.

Notas gerais

- Esse desafio poderá ser estendido por você e por outra pessoa engenheira do Nubank durante uma outra etapa do processo;
- Você deve entregar o código fonte de sua solução para nós em um arquivo comprimido (zip) contendo o código e toda documentação possível. Favor não incluir arquivos desnecessários como binários compilados, bibliotecas, etc;
- Não faça o upload da sua solução em nenhum repositório público como GitHub, BitBucket, etc;
- O Autorizador deve receber as operações através da entrada padrão (`stdin`) e retornar o resultado do processamento através da saída padrão (`stdout`), ao invés de uma API REST.