Abstração e Encapsulamento

ECM251- Linguagens de Programação I

Murilo Zanini de Carvalho

murilo.carvalho@maua.br

abril de 2020

Atividade 1

I. Crie um projeto com o nome Atividade1 e acima de todos os arquivos fontes coloque um bloco de comentário com o nome e o RA.
II. Utilize um arquivo fonte para cada classe criada.
III. Utilize todas as padronizações de nomes e nomenclaturas apresentados pela disciplina (Camel Case para Java), isso será parte da avaliação, bem como o código identado e comentado.
IV. A atividade terá aproximadamente 200 min de duração (9h30 até 13h00).

Não é permitido a utilização da internet para consultas diferente de retirar dúvidas com o professor. Espera-se que o código de honra do aluno seja seguido (http://web.stanford.edu/class/archive/cs/cs106b/cs106b.1164/handouts/honor-code.pdf).

V. Ao termino da atividade, subir ela no Github e enviar o link (do Github) para o professor pela ferramenta de comunicação utilizada na disciplina (Slack).

1. QRCode de Texto – Atividade desenvolvida pelo prof. Murilo Zanini de Carvalho

Fomos chamados por uma grande companhia para desenvolver um protótipo de pagamento utilizando apenas QRCodes, tanto para realizar e receber pagamentos. Grandes companhias também já desenvolveram produtos similares, como o Mercado Pago. Em outros países, esse sistema é utilizado amplamente por diversas pessoas da população. Essa demanda veio como uma forma de reduzir o contato das pessoas com cédulas de dinheiro, reduzindo a propagação de doenças e a necessidade das pessoas de carregar dinheiro físico.

O pagamento por QRCode apresenta elevados mecanismos de segurança quando comparado com outros tipos de pagamento (https://digitalcommons.wou.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com.br/&httpsredir=1&article=1002&context=computerscience_studentpubs). Os usuários desse sistema apresentam auto nível de satisfação (https://www.mdpi.com/2071-1050/9/7/1186/htm).

Essa forma de transação vai permitir que cada usuário possa receber créditos e realizar pagamentos por QRCodes únicos gerados para a transação específica. A validação de que a chave criada é única ainda não será implementada.

Ficamos incumbidos de desenvolver a primeira versão do sistema em que cada usuário pode possuir conta. Cada conta pode realizar pagamentos e receber créditos de outros usuários. Como ainda estamos nas etapas inicias do projeto, todas as informações que serão embarcadas nos QRCode serão apresentadas como Strings.

Quando um usuário for realizar um pagamento, ele deve informar o usuário que vai receber esse valor, o valor e a String do QRCode. Quando ele for receber algum valor, ele deve gerar a chave de autorização da transação.

1.1 Classes

O problema deve ser modelado respeitando os dois pilares da POO já aprendidos na disciplina.

No mínimo, crie classes para Usuários, Conta e para as Transações.

Todas as classes devem estar encapsuladas e acoplamento entre as Classes devem ser evitados.

Entrada e saída de dados deve estar apenas na classe principal “Atividade1”.

1.1.1 Usuarios

As informações que a classe que contém informações dos usuários deve conter são:
- nome
- senha
- e-mail
- e outros que você julgar necessário para a classe.

1.1.2 Contas

Nas contas deve existir informações sobre a mesma:
- idConta – esse id deve ser único para cada conta, pode estar relacionado com o total de contas já criadas.
- saldo
- e outros que você julgar necessário para a classe.

1.1.3 Transações

Essa classe deve possuir informações sobre as transações. Ela não deverá possuir nenhum atributo. Ela traz a implementação dos métodos para gerar os QRCode em String para cada operação. Para gerar os QRCode, algumas regras devem ser seguidas:

- Toda transação deve ser composta pelo id da conta;
- A String gerada deve conter o nome do usuário que vai receber o valor;
- A String gerada deve conter o valor da transação;
- Ela deve conter um número aleatório no intervalo 1000 e 9999, gerado por:

```
private static int getRandomNumberInRange(int min, int max) {
                Random r = new Random();
                return r.nextInt((max - min) + 1) + min;
}
```

Adaptado de (https://mkyong.com/java/java-generate-random-integers-in-a-range/), em 05/04/2020

Por exemplo, a conta de ID 5, do usuário Perigo, vai gerar um QRCode para receber o valor de 1 dólar, String gerada:

"5;PERIGO;1;1234"

Quando algum usuário for realizar o pagamento para a conta com a String gerada, ele deve fornecer ao método de pagamento os dois objetos, o pagador e o recebedor e a String. Para realizar a transação, o ID e o usuário da conta recebedora devem ser validados. Outro ponto, a transação apenas pode acontecer se o usuário pagador tiver crédito em sua conta. Para separar os campos da String, utilizar o seguinte *snippet* de código:

```
String s = "5;PERIGO;1;1234";
String[] dados = s.split(";");
```

Para acessar os membros de dados, utilizar dados[índice], onde índice é um número inteiro que representa a informação desejada.

2. Outras informações

- Forneça formas de atualizar as informações das classes sem quebrar o pilar Encapsulamento de POO.
- Defina quais são as informações cruciais para o funcionamento de cada classe e as coloque em um construtor.
- Cada uma das classes deve ter um método para devolver informações, que irá retornar uma String formatada com as informações do objeto.
- No método main, para fins de teste:
  - Crie três usuários
  - Cada um com uma conta (valores iniciais de saldo: 1000, 250, 3000)
  - As seguintes operações, se elas foram possíveis:
    - Usuário 1 gera uma operação de recebimento de 250;
    - Usuário 2 paga a requisição do usuário 1;
    - Usuário 3 paga a requisição do usuário 1;

- Usuário 2 paga a requisição do usuário 1;
- Usuário 2 gera uma operação de recebimento de 1000;
- Usuário 3 paga a requisição do usuário 2.

3. Adicionais

- Caso necessário crie métodos, variáveis ou atributos para auxiliar as interfaces entre as classes.

4. Possível Saída Após a Execução do Sistema:

```
Informe o nome do usuário 1:
All Might
Informe o nome do usuário 2:
One For All
Informe o nome do usuário 3:
Bakugo
Estado Inicial:
Nome Usuário: All Might - Saldo: 1000,00
Nome Usuário: One For All - Saldo: 250,00
Nome Usuário: Bakugo - Saldo: 3000,00
Estado Final:
Nome Usuário: All Might - Saldo: 1500,00
Nome Usuário: One For All - Saldo: 1000,00
Nome Usuário: Bakugo - Saldo: 1750,00

Process finished with exit code 0
```