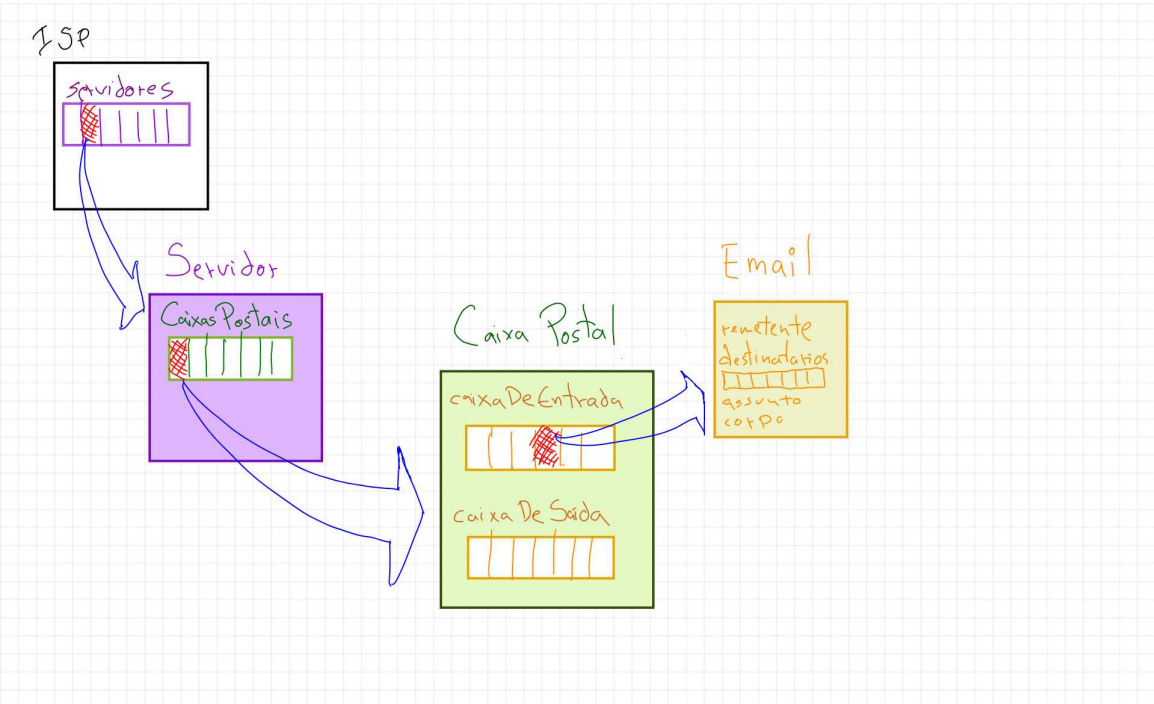
Universidade do Vale do Rio dos Sinos - UNISINOS
Algoritmos e Programação: Fundamentos - Trabalho Grau B - 2020/1

Elaborar um programa que implemente um Provedor de Serviços de Internet (**ISP** – Internet Service Provider), que irá ser responsável pelo envio e recebimento de emails dos seus usuários. Para isto, definir as seguintes classes:

- Email
  - Atributos: ***remetente***, (array de String) de ***destinatário(s)***, ***assunto*** e ***corpo do email*** (todos do tipo String)
  - Métodos:
    - getters e setters
    - construtor que recebe destinatário(s), assunto e corpo do email
    - construtor que recebe o remetente, destinatário(s), assunto e corpo do email;
    - Obs.: array de destinatários pode ter tamanho fixo de 10 elementos
- EmailComAnexo : subclasse de Email
  - Atributos ***anexo*** (é uma String)
  - Métodos:
    - getter e setter
    - construtor
- ISP
  - Atributos:
    - ***servidores***: array de Servidor Ver observações sobre o envio/recebimento de emails.
    - *totServidores*: int que representa número total de servidores inseridos no ISP
  - Métodos:
    - *boolean* **insereServidor**(Servidor servidor)
    - *boolean* **removeServidor**(String nomeServidor)
    - *void* **sendReceive**: este método "sincroniza" todas as caixas postais, realizado a distribuição de todos emails, isto é, realiza o envio e o recebimento de todo emails; para isto:
      - para cada Servidor**,**
        - para cada CaixaPostal de um servidor:
          - retira todos emails da *caixaDeSaida* e insere cada um deles na(s) respectiva(s) *CaixaDeEntrada* da(s) CaixaPostal(is) do(s) destinatário(s) de cada um destes emails (caixa postal esta que pode estar num servidor diferente que o do remetente). Ver observações sobre o envio/recebimento de emails.
    - construtor: recebe o tamanho do array de servidores
    - *Servidor* **getServidor**(String nomeServidor): retorna objeto Servidor; nome do objeto servidor é o argumento do método;
    - *String* **showAll**(): retorna uma String com a informação de todas caixas de entrada e saída de todas as caixas postais de todos servidores
- Servidor
  - Atributos:
    - *nomeServidor*: String ⇨ exemplo: "*kmail.com*"
    - ***caixasPostais***: array de CaixaPostal
    - *totCaixasPostais*: int que representa número total de caixas postais inseridas no servidor
  - Métodos:
    - *boolean* **addCx**(CaixaPostal caixapostal): insere uma CaixaPostal no array *caixasPostais* (parâmetro recebido é uma caixa postal)
    - construtor: recebe o nome do servidor e o tamanho do array caixasPostais
    - getters e setters
    - *CaixaPostal* **getCx**(String nome): retorna a caixa postal cujo dono corresponde ao argumento nome
    - *String* **showCxsPostais**(): retorna uma String com a informação (caixas de entrada e saída) de todas caixas postais deste servidor
- CaixaPostal
  - Atributos:
    - *nomeDono* : é a identificação do dono da caixa postal (String). Ex: "carlos" ⇨ o endereço de email completo é obtido pela concatenação do nome do dono da caixa postal com o nome do servidor ao qual sua caixa postal pertence. Ex.: "*carlos@kmail.com*"
    - ***caixaDeSaida***: array de emails a serem enviados
    - ***caixaDeEntrada***: array de emails recebidos
    - *totEmailsCxSaida*: int que armazena o número total de emails para envio

        - *totEmailsCxEntrada*: int que armazena o número total de emails recebidos
    - Métodos:
        - construtor, que recebe a identificação do dono da caixa postal (o *remetente*) e o tamanho dos arrays caixaDeSaida e caixaDeEntrada
        - *boolean* **send**(Email email): recebe um objeto Email e o insere em seguida na *caixaDeSaida* desta caixa postal (o parâmetro é o email criado pelo usuário); retorna ***true*** se conseguiu realizar a inserção
        - *boolean* **receive(**Email email**):** recebe um objeto Email e o insere em seguida na *caixaDeEntrada* desta caixa postal (este método pode ser chamado pelo método sendReceive do ISP); retorna ***true*** se conseguiu realizar a inserção
        - *String* **showInbox**(): retorna uma String com a lista de todos emails que estão na *caixaDeEntrada* (remetente/assunto/corpo (e anexo, se for o caso)) desta caixa postal
        - *String* **showOutbox**():retorna uma String com a lista de emails que estão na *caixaDeSaida* (destinatário/assunto/corpo (e anexo, se for o caso)) desta caixa postal
        - *void* **clearInbox**(): remove todos e-mails da caixaDeEntrada
        - **getters** dos atributos totEmailsCxSaida e totEmailsCxEntrada

Um exemplo de estrutura gerada a partir destas classes é ilustrada na imagem a seguir:



<u>Observações sobre envio/recebimento de emails</u>:

- No ISP, o primeiro servidor (índice zero) do array de servidores é de uso restrito do sistema, isto é, não pode ser usado para cadastrar caixas postais dos usuários; este primeiro servidor contém somente uma caixa postal, cujo nome do dono é *Postman* e que representa o gerente virtual do sistema de entrega de emails; **mensagens de erro** que possam porventura ocorrer durante a distribuição de emails são geradas pelo *Postman* através de emails enviados por ele para os usuários que originaram o problema;
- Se um email não pode ser entregue ao destinatário, o *Postman* envia um ***email de ocorrência de erro*** para o remetente, explicando o motivo da não entrega e concatenando as informações do email não entregue ao corpo do email enviado pelo *Postman*; possíveis <u>motivos para que a entrega de um email não possa ser efetuada</u>:
    - Endereço(s) do(s) email(s) do(s) destinatário(s) incorreto(s) ou inexistente(s);
    - Caixa de entrada do destinatário está cheia.
- No momento em que o método sendReceive não consegue realizar uma entrega, ele monta um *email de ocorrência de erro* e insere na caixa de saída do *Postman*; após o método sendReceive ter distribuído todos emails, os emails de erro que estão na caixa de saída do *Postman* são entregues.

<u>Exemplo de classe de teste</u>:

```
ISP isp = new ISP(10);  //suporta até 10 servidores
isp.insereServidor(new Servidor("kmail.com", 20));  //suporta até 20 caixas postais
isp.insereServidor(new Servidor("tierra.com.br", 50));
isp.insereServidor(new Servidor("oi.com", 15));
```

```
CaixaPostal  cpCarlos  = new CaixaPostal (“carlos”, 10, 10); //10 caixas de entrada e 10 de saida
CaixaPostal  cpEduardo = new CaixaPostal (“eduardo”, 10, 10);
CaixaPostal  cpJonas   = new CaixaPostal (“jonas”, 10, 10);

isp.getServidor(“kmail.com”).addCx(cpCarlos);
isp.getServidor(“kmail.com”).addCx(cpEduardo);
isp.getServidor(“kmail.com”).addCx(cpJonas);
isp.getServidor(“oi.com”).addCx(new CaixaPostal (“luis”, 20,20));
isp.getServidor(“oi.com”).addCx(new CaixaPostal (“marcio”, 10,10));
```

remetente | destinatários | assunto | corpo do email

```
Email em = new Email(“carlos”,new String[]{“Eduardo@kmail.com”,“luis@oi.com”},”preciso de ferias”,”agora.”);

cpCarlos.send (em);
cpEduardo.send (new Email(new String[]{“carlos@kmail.com”}, “pergunta”, “oi, tudo bem?”));

isp.getServidor(“oi.com”).getCx(“luis”).send(new EmailComAnexo(new String[]{“carlos@kmail.com”},“projeto”,”alo!”,
“este eh o anexo));

isp.sendReceive();        //envia e recebe emails entre todas caixas postais

System.out.println(cpCarlos.showInbox()); //mostra caixaDeEntrada do Carlos, listando emails recebidos
(remetente/assunto/corpo e anexo se for o caso)

System.out.println(isp.showAll());
```

Obs.: exemplo de como criar um *array literal* de strings ➡ `new String[] {“palavra”, “nome”, ”texto”};`

A figura a seguir ilustra o caminho realizado por um objeto Email enviado para ***bia@kmail.com*** saindo da caixa de saída de ***edu*** (cuja caixa postal esta no servidor ***oi.com***) e alcançando no seu destino a caixa de entrada da caixa postal pertencente a ***bia*** no servidor ***kmail.com***