SCC 204 – Programação Orientada a Objetos
Trabalho 1

Todos os trabalhos da disciplina são parte de um projeto que envolve o desenvolvimento de um Comunicador Instantâneo. Nesta primeira etapa deve-se desenvolver:

1) Uma classe Usuário contendo as seguintes informações:
•Email do usuário
•Senha do usuário
•Nome completo
•Apelido ou nickname para o Comunicador Instantâneo
•Cidade
•Estado
•País
•Data de nascimento
E os seguintes métodos:
•setters e getters
◦ Os setters devem ainda verificar:
▪ Se o email do usuário tem um arroba, se não tiver, não aceite o e-mail. Se o e-mail tiver um espaço, ou tab, ele também não será aceito. Nesses casos, escreva na tela

```
printf("Email invalido\n");
```

em seguida, continue a leitura do campo email até que esse seja válido.

▪ Se a senha tem de 2 a 30 caracteres. Se tiver mais ou menos caracteres, ela não será válida e deve mostrar a mensagem de erro abaixo:

```
printf("Quantidade de caracteres da senha invalida\n");
```

em seguida, continue a leitura do campo senha até que esse seja válido.

▪ Se a data de nascimento é uma data válida. Verifique apenas se os dias estão entre 1 e 31, se o mês está entre 1 e 12, e, finalmente, se o ano é menor ou igual ao atual. Caso não seja válida, escreva na tela:

```
printf("Data de nascimento invalida\n");
```

em seguida, continue a leitura do campo data de nascimento até que esse seja válido.

•insereRelacionamentoUsuario(usuario) → adicionar usuario na lista de contatos do usuário atual. Essa lista de contatos deve, também, ser mantida em um segundo arquivo binário de dados **relacionamentos.dat**
•removeRelacionamentoUsuario(usuario) → remove usuario da lista de contatos do usuário atual. Lembre-se que a lista de contatos é mantida em no arquivo binário de dados **relacionamentos.dat**

•autenticar(email, senha) → método que recebe email e senha, verifica se usuário existe por meio de busca binária em arquivo de indexação **usuarios.idx** e carrega os demais dados do usuário contidos no arquivo binário de dados **usuarios.dat** para dentro do objeto
•imprimir → imprime informações do usuário no formato:

```
printf("%s\n", email);
printf("%s\n", nome);
printf("%d\n", numeroDeContatosNaLista);
```

•imprimirContatos → lista todos os contatos do usuário no formato (segundo a ordem em que foram adicionados):

```
printf("%s\t%s\n", emaildocontato, apelido);
```

2) Cada Usuário deve conter um vetor com seus contatos. Esse vetor deve ser implementado em uma classe Vector para o caso de C++. Java já conta com uma classe Vector previamente implementada e que deve ser utilizada. Há uma classe vector na Standard Template Library (STL) do C++, no entanto, neste caso, o aluno deve criar sua própria classe Vector com base na disponível em Java.

3) Criar uma classe Comunicador com os seguintes métodos:

•inserirUsuario → que recebe um objeto do tipo Usuário, verifica se o usuário existe (via email) e o salva em um arquivo binário de dados **usuarios.dat**. Após inserir, deve-se regerar o arquivo binário de indexação dos dados **usuarios.idx**.
•removerUsuario → que recebe o e-mail do Usuário, verifica se o usuário existe (via email) e o remove do arquivo binário de dados **usuarios.dat**. Após remoção, deve-se regerar o arquivo binário de indexação dos dados **usuarios.idx**. Lembre-se de remover todos os relacionamentos existentes de outros usuário com o removido.
•buscaSeqUsuario → que recebe uma string com o email do Usuário e busca por um registro no arquivo binário de dados **usuarios.dat** de maneira sequencial. Em seguida, carrega esses dados e cria e retorna um objeto do tipo Usuario
•buscaBinUsuario → utiliza um arquivo de indexação **usuarios.idx** para buscar um usuário via seu email. Deve-se utilizar busca binária. Esse arquivo de indexação deve conter os emails e os offsets dos registros de usuários contidos no arquivo de dados. Em seguida, posiciona no registro correspondente do arquivo binário de dados **usuarios.dat**, carrega esses dados e cria e retorna um objeto do tipo Usuario

Descrição dos arquivos

Todos os arquivos serão binários.

Formato do arquivo **usuarios.dat**:

Email do usuário contendo 80 caracteres
Senha do usuário contendo 30 caracteres
Nome completo contendo 80 caracteres

Apelido ou nickname para o Comunicador Instantâneo contendo 30 caracteres
Cidade contendo 30 caracteres
Estado contendo 30 caracteres
País contendo 50 caracteres
Data de nascimento contendo 2 caracteres para dia, 2 caracteres para mês, 4 caracteres para ano

Formato do arquivo **usuarios.idx**:

Email do usuário contendo 80 caracteres
Offset do registro correspondente no arquivo de dados **usuarios.dat** (Veja mais sobre o comando fseek para usar em C++ e veja mais sobre as classes File, FileReader e FileWriter para Java)

Formato do arquivo **relacionamentos.dat**:

Email do usuário contendo 80 caracteres
Email do usuário de contato contendo 80 caracteres
Offset do usuário de contato no arquivo de dados **usuarios.dat**

Sobre este trabalho

Deve-se implementar este trabalho na linguagem C++ criando todas as classes manualmente. Deve-se, também, implementar o mesmo trabalho na linguagem Java. Haverá dois itens no sistema de submissão de trabalhos, um para submeter em linguagem C++ e outro para Java. Ambos são obrigatórios.

Para submeter o trabalho

1) Acesse o site: https://ssp.icmc.usp.br
2) Cadastre-se na opção “Cadastro”
3) Clique na opção “Login” e entre no sistema
4) Clique em “Matricular em disciplina” e selecione “SCC0204 - Programação Orientada a Objetos (Ano: 2013/Semestre: 1- Turma: B)” e matricule-se
5) Clique em “Login” novamente e, em seguida, na opção “Listar disciplinas matriculadas”
6) Clique na opção “Submeter exercícios” da disciplina
7) Escolha o exercício e, em seguida, defina a linguagem de programação como “Zip/Makefile”
8) Selecione seu arquivo zipado contendo o código e o Makefile do programa
9) Submeta seu programa
10) Clique novamente na opção “Login” e, em seguida, na opção “Listar disciplinas matriculadas” e depois em “Resultados” para tomar ciência dos resultados da correção automática

Lembre-se que as notas dependem, ainda, de uma avaliação posterior.

Para criar seu projeto em C++

O projeto zipado deve conter, em seu diretório raíz, um arquivo Makefile. Internamente podem haver outros diretórios. Por exemplo:

projetoc++.zip
./Makefile
./meuprojeto/meuprograma.h
./meuprojeto/meuprograma.cpp

Dentro do arquivo Makefile deve SEMPRE haver duas targets (all e run). LEMBRE-SE QUE O NOME DO ARQUIVO Makefile DEVE TER A PRIMEIRA LETRA EM MAIÚSCULO! Exemplo de Makefile:

Arquivo Makefile

```
all:
	@g++ -o ./meuprojeto/meuprograma ./meuprojeto/meuprograma.cpp

run:
	@./meuprojeto/meuprograma
```

Observe que há um arroba (@) antes dos comandos contidos no Makefile. Isso é obrigatório, caso contrário a linha de comando será enviada para o sistema de correção automática e ocorrerá erro.

Para criar seu projeto em Java

O projeto zipado deve conter, em seu diretório raíz, um arquivo Makefile. Internamente podem haver outros diretórios. Por exemplo:

projetojava.zip
./Makefile
./meuprojeto/Meuprograma.java

Dentro do arquivo Makefile deve SEMPRE haver duas targets (all e run). LEMBRE-SE QUE O NOME DO ARQUIVO Makefile DEVE TER A PRIMEIRA LETRA EM MAIÚSCULO! Exemplo de Makefile:

Arquivo Makefile

```
all:
	@javac meuprojeto/Meuprograma.java

run:
	@java -cp meuprojeto Meuprograma
```

Observe que há um arroba (@) antes dos comandos contidos no Makefile. Isso é obrigatório, caso contrário a linha de comando será enviada para o sistema de correção automática e ocorrerá erro.

A opção -cp (ou classpath) define o diretório em que se encontram as classes compiladas (arquivos extensão .class).

Saída esperada para o programa

Algumas opções serão encaminhadas para o programa. Essas opções são:

1) inserirusuario → ao receber essa opção, seu programa deve capturar os dados referentes ao usuário na ordem:

•Email do usuário
•Senha do usuário
•Nome completo
•Apelido ou nickname para o Comunicador Instantâneo
•Cidade
•Estado
•País
•Data de nascimento

em seguida, criar um objeto da classe Usuario e depois invocar o método inserirUsuario de um objeto da classe Comunicador. Ao final imprima:

```
printf("Usuario inserido com sucesso\n");
```

ou

```
printf("Falha ao inserir usuario\n");
```

caso um usuário com mesmo email já esteja cadastrado no sistema.

2) removerusuario → ao receber essa opção, seu programa deve ler o email do usuário e chamar o método removerUsuario do objeto da classe Comunicador. Esse método é responsável por removê-lo do sistema. Ainda dentro do método removerUsuário (classe Comunicador) faça: se usuário foi removido corretamente exiba:

```
printf("Usuario removido com sucesso\n");
```

caso contrário:

```
printf("Problemas ao remover usuario\n");
```

3) buscaSeqUsuario → ao receber essa opção, seu programa deve ler o email do usuário e chamar o método buscaSeqUsuario do objeto da classe Comunicador. Receba o objeto retornado, o qual será da classe Usuario, e chame o método imprimir e, em seguida, imprimirContatos

4) buscaBinUsuario → ao receber essa opção, seu programa deve ler o email do usuário e chamar o método buscaBinUsuario do objeto da classe Comunicador. Receba o objeto retornado, o qual será da classe Usuario, e chame o método imprimir e, em seguida, imprimirContatos

5) inserirrelacionamento → ao receber essa opção, seu programa deve ler o email do primeiro usuário e, em seguida, do segundo. Busque por ambos usuários utilizando buscaBinUsuario do objeto da classe Comunicador. Em seguida, adicione o segundo usuário na lista de contatos do primeiro e vice-versa. Se um dos usuários não existir exiba a seguinte mensagem na tela:

```
printf("Relacionamento nao pode ser criado\n");
```

Caso o relacionamento seja criado com sucesso exiba:

```
printf("Relacionamento criado com sucesso\n");
```

Exemplo de entrada para seu programa

inserirusuario
"joao@teste.com"
"senha1"
"Joao da Silva"
"jsilva"
"Sao Carlos"
"SP"
"Brasil"
"12/02/1973"
inserirusuario
"carlos@teste.com"
"senha2"
"Carlos da Silva"
"csilva"
"Recife"
"PE"
"Brasil"
"01/07/1962"
inserirrelacionamento
joao@teste.com
carlos@teste.com
buscaSeqUsuario
carlos@teste.com
buscaBinUsuario
joao@teste.com
removerusuario
joao@teste.com
removerusuario
carlos@teste.com

Exemplo de saída esperada para a entrada anterior:

Usuario inserido com sucesso

Usuario inserido com sucesso
Relacionamento criado com sucesso
carlos@teste.com
Carlos da Silva
1
joao@teste.com jsilva
joao@teste.com
Joao da Silva
1
carlos@teste.com csilva
Usuario removido com sucesso
Usuario removido com sucesso