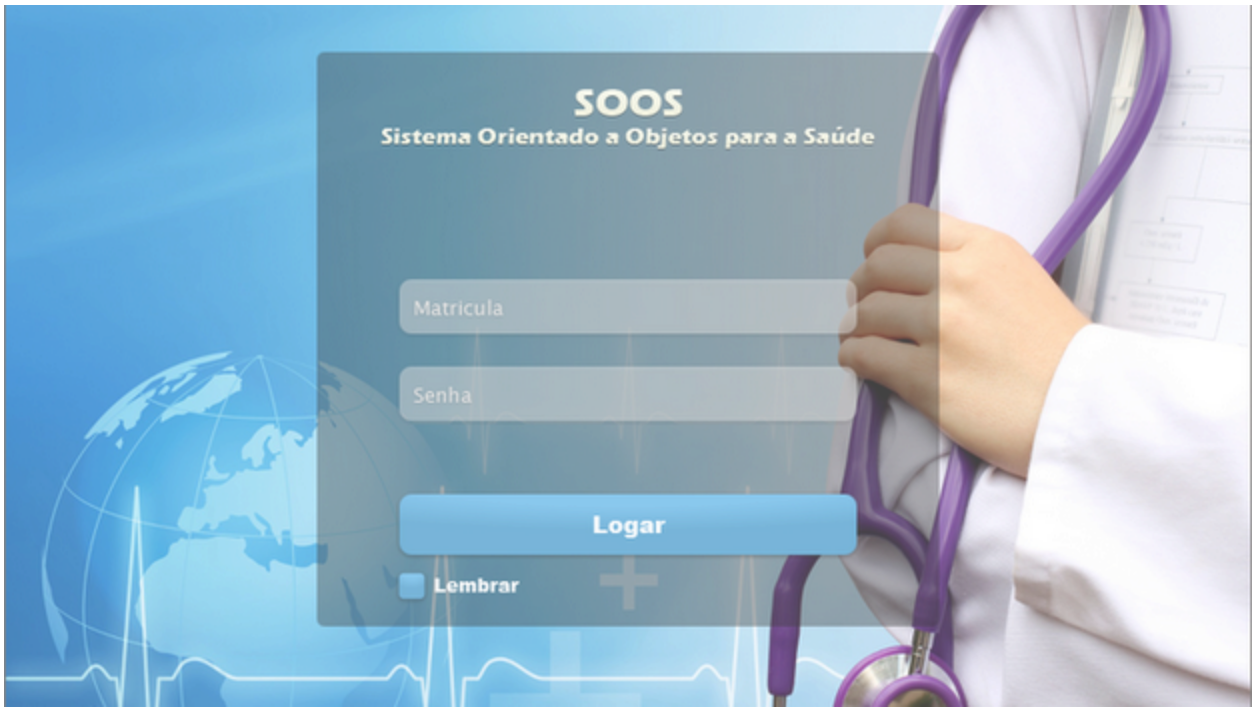
Universidade Federal de Campina Grande – UFCG
Centro de Engenharia Elétrica e Informática – CEEI
Departamento de Sistemas e Computação – DSC
Disciplina: Laboratório de Programação 2

Projeto da Disciplina - SOOS



O SOOS (Sistema Orientado a Objetos para a Saúde) é um sistema que será usado por gestores e colaboradores de clínicas particulares que possuem parceria com o Sistema Único de Saúde (SUS). O objetivo é fornecer meios que possam agilizar e conduzir a gerência dos recursos humanos e físicos do hospital, como também auxiliar nos procedimentos internos inerentes à organização e seus departamentos.

A organizaçao do hospital é dividida em 3 grupos: comitê gestor, corpo clínico e corpo profissional.

- O comitê gestor é composto por um diretor geral, cuja responsabilidade é gerir os departamentos (farmácia, área clínica e bloco cirurgico), o corpo clinico e o corpo profissional.
- O corpo clínico é composto pelos médicos.
- O corpo profissional é composto pelos técnicos administrativos e atendentes do hospital.

Para isso, teremos os seguintes níveis de acesso: Note que os níveis de acesso são diferenciados de acordo com o cargo de cada funcionário. Garanta que o seu sistema atribua aos usuários os seguintes privilégios.

Cargo	Acessos
Diretor geral	Possui acesso a todas as funcionalidades relacionadas do sistema.
Médico	Possui acesso a todas as funções do corpo clínico/cirúrgico.
Técnico Administrativo	Possui acesso ao cadastro e edição de atributos de um paciente.

Inicialmente o sistema possui apenas um único usuário que é o diretor geral do hospital, esse tem acesso a todas as funcionalidades do sistema. Esse usuário será responsável por gerenciar todos os demais usuários do sistema, e também por inserir novos gestores e colaboradores no sistema. Os usuários do sistema, inseridos pelo diretor, terão acesso apenas as opções inerentes ao seu grupo na organização.

Implementação:

Será usada uma arquitetura considerando uma Façade e um Controller. As demais entidades do sistema devem ser definidas de acordo com o seu design. Lembre de respeitar o GRASP e de utilizar o máximo de conceitos explorados na disciplina. Serão fornecidos scripts de teste de aceitação que serão usados como um dos critérios de avaliação. Utilize os scripts fornecidos pelos monitores para criar seus próprios testes de aceitação.

Importante: Como todo sistema da vida real, o cliente pode solicitar modificações. Portanto, esteja atento à modificações na especificação que podem surgir durante o acompanhamento do projeto.

Caso de uso 1:

Primeiro cadastro, login do diretor geral e cadastro de novos colaboradores.

Descrição principal: *Como diretor geral desejo iniciar e realizar o cadastro de novos colaboradores para que possam posteriormente usar o sistema.*

O sistema inicia em seu modo bloqueado, para ter acesso à todas as funcionalidades será necessário inserir uma chave de desbloqueio. Crie um metodo chamado *liberaSistema(...)* que receberá uma chave de seguranca. Este metodo será responsával por gerar o primeiro cadastro do sistema. Ao ser chamado, o método deverá retornar uma nova matrícula com privilégios de

Diretor. Use a chave "**`c041ebf8`**" para efetuar o desbloqueio, note que este método só poderá ser chamado uma única vez e a partir deste ponto apenas diretores poderão cadastrar novos funcionários.

Agora que já possui sua primeira matrícula você conseguirá conectar-se ao sistema a partir da chamada do método *realizaLogin(...)* e, consequentemente, ter acesso a todas as funções inerentes ao seu cargo. Utilize a chave como senha para acessar o sistema.

Diante disso, implemente o cadastro de novos colaboradores ao chamar o método *cadastraFuncionario(...)*. Para que o cadastro seja efetivado será necessário informar nome, cargo e a data de nascimento do mais novo funcionário. Faça tratamento para valores inválidos que possam ser especificados. Para garantir a integridade do sistema, será necessário a **geração automática** de **matrículas** e **senhas** ao cadastrar um novo funcionário.

Uma matrícula, para ser formada deve obedecer o seguinte padrão: **Prefixo + Ano + Sufixo**

- **Prefixo:** representa o ID de cada cargo, sendo Diretor (1), Médico (2) e Técnico (3).
- **Ano:** representa o ano em que está sendo realizado o cadastro.
- **Sufixo:** será um número de três dígitos que representa a quantidade de cadastros realizados.

O sistema gerará uma **senha automática** para o novo usuário, essa senha será composta pelo seu **ano de nascimento** e dos **quatro primeiros dígitos da matrícula**.

Por exemplo: Funcionário com matricula **12016001** e Data de Nascimento **26/09/1992**, a senha automática será: **19921201**

Importante: Para facilitar sua implementação envolvendo datas (dias, meses e anos) **em todo o projeto** utilize as classes LocalDate de Java 8. Note que com isso, você não conseguirá executar seu programa no Java 7.

Caso de uso 2:
Atualizar informações de usuários e Excluir usuários do sistema

Descrição principal: *Como diretor geral desejo atualizar as informações dos demais usuários cadastrados para permitir atualização dos dados. Como usuário logado desejo atualizar minhas próprias informações cadastradas para mante-las atualizadas.*

Para implementar essa funcionalidade crie uma opção onde é possível pesquisar por usuários do sistema, a busca será feita exclusivamente pela **matrícula**. O sistema permite atualizar todas as informações de um usuário (*exceto a matrícula*) e também fornece a opção de **exclusão de usuários**, porém para isso é solicitado uma confirmação da senha do diretor.

Fique atento, pois qualquer usuário que esteja logado poderá atualizar suas próprias informações. Neste caso a busca por um usuário não será necessária, uma vez que o usuário que estará logado no sistema será o próprio usuário atualizado. Cada usuário poderá atualizar suas informações básicas como nome, senha, etc. Ao tentar alterar a senha, o usuário deverá inserir sua senha antiga para que a nova seja aceita pelo sistema.

Para que atributos possam ser alterados, as novas informações devem seguir rigorosamente os seguintes padrões:

- **Senha:** conjunto alfanumérico com extensão de 6 à 12 caracteres.
- **Nome:** contem apenas letras com extensão máxima de 16 caracteres.

Caso de uso 3:
Cadastrar e atualizar prontuários dos pacientes

Descrição principal: *Como técnico administrativo desejo cadastrar novos pacientes para gerenciar suas ações na clínica.*

Para que o cadastro de um novo paciente seja efetuado será necessário fornecer as seguintes informações: nome, data de nascimento, peso, tipo sanguineo, sexo biológico e gênero. Todo paciente ao ser cadastrado no sistema recebeŕa uma *id* que serve para distinguir os mesmos. Essa *id* deve ser um número inteiro que representa a quantidade de pacientes já cadastrados. ***Por exemplo:*** A *id* do primeiro paciente deve ser **1**, do segundo paciente **2** e assim sucessivamente.

Dica: Para facilitar sua implementação, dê uma olhada na classe UUID de Java que facilita a geração e manutenção de ids únicas. Use:

```
UUID idCriada = UUID.randomUUID()
```

Para manter as informações e o histórico de tratamentos do paciente organizados, todo paciente deve estar associado a um **prontuário**. O prontuário armazenará todas as informações de um paciente, além disso, nele será possível, futuramente, manter um conjunto de procedimentos aos quais o paciente deverá ser submetido, dentre outras informações. Os prontuários devem ser organizados por **ordem alfabética do nome do paciente**

correspondente, não será permitido a exclusão de nenhum prontuario do sistema (não precisa se preocupar com os empates).

Caso de uso 4:
Farmácia e Medicamentos

Descrição principal: *Como técnico administrativo desejo gerenciar medicamentos para permitir a disponibilização dos medicamentos para tratamentos dos pacientes.*

A farmácia é o departamento do hospital responsável pela criação e gerência de medicamentos. Além disso, fornece todo o suprimento necessário para a efetivação de quaisquer procedimentos realizados pelo corpo clínico e cirúrgico em casos de usos futuros. Um medicamento é composto por um nome, preço, quantidade e categorias (uma ou mais). Existem seis categorias de medicamento: Analgésico, antibióticos, antiemético, anti-inflamatório, antitérmico e hormonais.

Além disso, temos dois tipos de medicamento: **medicamento genérico** e **medicamento de referência**. Ambos possuem as mesmas informações, e diferem apenas no preço, de forma que o medicamento genérico é mais barato. Ou seja, ao criar um remédio é especificado um preço e o tipo do remédio. Se o remédio for genérico, a farmácia irá criar um medicamento cujo preço **terá um desconto de 40% do preço especificado**. O medicamento de referência, por sua vez, será cadastrado com o preço integral especificado. A farmácia realiza dois tipos de consulta de medicamentos:

- **Categoria:** Na busca por categoria deve-se retornar uma **lista de nomes dos remédios encontrados** naquela categoria. Essa lista deve estar **ordenada pelo menor preço** dos medicamentos encontrados.
- **Nome:** Esta retorna uma representação em String com as informações básicas referentes ao remédio com o nome especificado.

Para fins de consulta, é possível verificar também **todos os remédios disponíveis** no estoque da farmácia, utilizando duas formas de ordenação: ordem alfabética, e a ordenação por menor preço de medicamento.

Fiquem atentos…
em breve teremos mais casos de uso para implementar.

Considerações Importantes:

- Independente da complexidade do sistema, lembre o **mais importante** princípio de design: ***KISS - Keep It Simple, Stupid!***
- Realize o **tratamento adequado de Exceptions**, considerando uma hierarquia de exceções voltada para o tratamento de Exceptions checked e, se for o caso, Exceptions unchecked.
- Os testes fornecidos pelos professores e monitores serão testes de aceitação. Aumente a qualidade de seu código **criando testes de unidade pelo JUnit**, e também, **seus próprios scripts de teste de aceitação**. Apesar de ser um critério **extra** para a avaliação, os testes terão um efeito importante para determinar se seu código está bem feito, e se atinge as funcionalidades de todos os casos de uso.
- O projeto foi criado para explorar **todos os conteúdos vistos na disciplina**. Portanto, fique atento a cada trecho dele. Uma **dica** é enviar com o projeto final um arquivo texto (README.txt) detalhando, em seu design, **onde está cada assunto visto na disciplina** como por exemplo: Herança, as distribuilções de responsabilidades (Expert, Creator, Alta coesão, etc.), Composição, Coleções (quais e por quê), Strategy, dentre outros.
- As informações sobre a entrega e os demais artefatos para implementação do projeto estão disponíveis no Canvas (Páginas > Projeto de LP2).