Sistema para Gerenciamento de Projetos

Caderno:	Primeiro Caderno		
Criada em:	20/02/2019 16:29	**Atualizada ...**	20/02/2019 18:27
Autor:	paul.cavalcantesilva@gmail.com		

Funcionalidades:

O projeto tem como funcionalidades básicas:

1. Adicionar um projeto ao sistema.
2. Alterar as informações de um projeto já existente.
3. Adicionar um usuário ao sistema.
4. Adicionar uma atividade ao sistema.
5. Alterar as informações de uma atividade.
6. Consultar um projeto, uma atividade ou um usuário no sistema.
7. Associar um usuário a um projeto.
8. Associar um usuário a uma atividade.
9. Associar um responsável a uma atividade.

Essas funcionalidades com as quais o usuário pode interagir, há três funcionalidades que são uso exclusivo do sistema:

1. Mostrar Projetos existentes no sistema ou no usuário.
2. Mostrar Atividades existentes no sistema ou no usuário.
3. Buscar um projeto no sistema ou no usuário.

Classes:

- **Motivação:**

Precisava de uma classe que pudesse executar os métodos mais básicos, uma classe para armazenar minha relação de usuários, projetos e atividades, também era necessário armazenar as informações dos usuários e trabalhos(projetos e atividades).

- **Solução:**

No meu código há 9 classes, são elas:

1. Main
2. Sistema
3. Trabalho
4. Projeto
5. Atividade
6. Usuario
7. Profissional
8. Coordenador
9. Ferramentas(Interface)

São super-classes:

1. Usuario

2. Trabalho

São sub-classes da classe Usuario:

1. Profissional
2. Coordenador

São sub-classes da classe Trabalho:

1. Projeto
2. Atividade

- **Vantagens:**

A vantagem é que as funções ficaram bem definidas em cada classes, ou seja, cada classe tem relação com os métodos que ela possui, nada está atribuído por acaso. Há uma relação entre a classe Sistema e a classe Usuario, ambos utilizam métodos em comum, através da classe interface Ferramentas.

- **Desvantagens:**

O sistema ficou com um número de classes que poderia ser reduzida.

Distribuição de métodos:

- **Motivação:**

Era necessário distribuir os métodos de uma forma que todos estivessem relacionados a classe que ele está implementado de forma que um método ajudasse na execução de outro e assim por diante até executar a tarefa completamente.

- **Solução:**

No meu sistema, Três classes possuem métodos realmente importante, são eles:

1. Classe Main:

- void AdicionarProjeto(): Esse método pede ao usuário a identificação(nome) do projeto e o tipo, verifica se já existe um projeto com o mesmo nome na minha classe Sistema com o método BuscarProjeto() da minha interface Ferramentas, caso exista, ele informa ao usuário, caso não exista, ele adiciona ao sistema.

- void AlterarInfoProjeto(): Esse método pede ao usuário a identificação(nome) do projeto, verifica se já existe um projeto com o mesmo nome na minha classe Sistema com o método BuscarProjeto() da minha interface Ferramentas, caso exista, o próximo passo é alterar as informações, a cada nova informação, ele pergunta ao usuário se ele deseja alterar essa informação.

- void AdicionarUsuario(): Esse método pede ao usuário o e-mail do usuário que ele deseja adicionar, em seguida, verifica se já existe algum usuário com o mesmo e-mail na minha classe sistema, como é uma coleção Map que armazena, utilizo apenas os métodos disponíveis do Map, a chave será o e-mail e o valor será o objeto instanciado da classe Usuario.

- void AdicionarAtividade(): Esse método pede ao usuário a identificação(nome) da atividade, verifica se já existe uma atividade com o mesmo nome na classe Sistema através do método BuscarAtividade() da interface Ferramentas, caso exista, ele informa ao usuário, caso não existe, ele adiciona no sistema.

- void AlterarInfoAtividade(): Esse método pede ao usuário a identificação(nome) da atividade, verifica se já existe uma atividade com o mesmo nome na minha classe Sistema com o método BuscarAtividade() da minha interface Ferramentas, caso exista, o próximo passo é alterar as informações, a cada nova informação, ele pergunta ao usuário se ele deseja alterar essa informação.

-void Consulta(): Esse método pede ao usuário o que ele quer consultar, se a opção for usuário, o método ConsultarUsuario() na classe Sistema é chamado, se for Projeto, o método ConsultarProjeto() na classe Sistema é chamado e por último, se for Atividade, o método ConsultarAtividade() na classe Sistema é chamado.

- void AssociarUsuarioaProjeto(): Esse método pede ao usuário o e-mail do usuário que será associado ao projeto, verifica se esse usuário existe, caso exista:

1. Pede ao usuário a identificação do projeto e verifica se ele existe através do método BuscarProjeto().
2. Se ele existir, verifica se o usuário já faz parte do projeto através do método BuscarProjeto(), caso ele não faça, adiciona ele ao projeto e adiciona o projeto na lista de Projetos do usuário, se o usuário for do tipo Coordenador e não tenha nenhum coordenador associado ao projeto, esse usuário será associado como coordenador, caso contrário ele será adicionado a lista de usuários do projeto.
3. Em seguida, pergunta se ele quer adicionar uma atividade ao projeto, caso a resposta seja sim, ele vai pedir o nome da atividade e verificar se ela existe através da classe BuscarAtividade(), caso ela exista, o próximo passo é verificar se o usuário já tem essa atividade na sua lista de atividades com o método BuscarAtividade(), então se ele não fizer parte, adicionar a atividade ao projeto e ao usuário.

- void GerarRelatorio(): Esse método chama os seguintes métodos presente na classe Sistema: MostrarProjetos(), MostrarAtividades(), MostrarUsuario().

2. Classe Sistema:

- void MudarEstado(int dia, int hora): Esse método segues os passos:

1. Pede a identificação do projeto, verifica se ele existe.
2. Caso ele exista, verifica o status atual do projeto, quando o usuário escolhe o método AdicionarProjeto(), o status do projeto é setado como "criado".
3. Se o projeto tiver como status "criado", o próximo passo é verificar se todas as informações estão inseridas, caso estejam e caso o dia e hora passados batam com a data inicial e hora inicial do projeto, o status é alterado para "iniciado".
4. Caso contrário, ele informa ao usuário que não foi possível mudar o status.
5. Caso ele identifique o status do projeto como "iniciado", o status será mudado para "em andamento" e o usuário será informado.
6. Caso ele identifique o status do projeto como "em andamento", o próximo passo é verificar se há atividades relacionadas ao projeto, caso exista, o status do projeto será alterado para "concluido".

- void ConsultarProjeto(): Esse método é chamado pelo método Consulta() na classe Main, procura o projeto através do método BuscarProjeto() e mostra todas as informações do projeto, incluindo quem é o coordenador, o status atual, o tipo do projeto, as atividades atribuídas a esse projeto, os usuário envolvidos.

- void ConsultarAtividade(): Esse método é chamado pelo método Consulta() na classe Main, procura a atividade através do método BuscarAtividade() e mostra todas as informações da atividade.

- void ConsultarUsuario(): Esse método é chamado pelo método Consulta() na classe Main, mostra todas as informações do usuário.

- void MostrarUsuario(): É chamado pelo método GerarRelatorio() na classe Main e

mostra todos os usuários do sistema.

- void MostrarProjetos(): é chamado em GerarRelatorio() e mostra todos os projetos e suas informações.

- void MostrarAtividades(): é chamado em GerarRelatorio() e mostra todas as atividades e suas informações.

- Trabalho BuscarProjeto(): é chamado na classe Main() por determinados métodos e retorna null ou um projeto existente.

- Trabaho BuscarAtiviade(): é chamado na classe Main() por determinados métodos e retorna null ou uma atividade existente.

3.Classe Usuario:

- Trabalho BuscarProjeto(): é chamado na classe Main() por determinados métodos e retorna null ou um projeto existente.

- Trabaho BuscarAtiviade(): é chamado na classe Main() por determinados métodos e retorna null ou uma atividade existente.

- void MostrarProjetos(): é chamado em ConsultarUsuario() e mostra todos os projetos e suas informações.

- void MostrarAtividades(): é chamado em ConsultarUsuario() e mostra todas as atividades e suas informações.

- ## Vantagen:

Tarefas bem distribuídas ocasionando o melhor entendimento de cada método.

- ## Desvantagens:

Um número muito grande de métodos.

Herança:

- ## Motivação:

Havia classes com alguns atributos diferentes e que poderiam ser separadas em sub-classes.

- ## Solução:

Criei duas super-classes, Usuario e Trabalho, as sub-classes Profissional e Coordenadores herdam de Usuario e as sub-classes Projeto e Atividade herdam de Trabalho.

- ## Vantagem:

Há menos repetição de código, há reutilização.

- ## Desvantagem:

Algumas sub-classes poderiam ser desconsideradas pois os atributos diferentes são mínimos.

Abstrata: Não vi motivos para utilizar pois não tinha classes similares que implementavam métodos iguais.

Interface:

motivação:

As classes Sistema e Usuario tinham funções iguais que poderiam ser implementadas de formas diferentes.

Solução:

Criei a interface Ferramenta que implementa os métodos MostrarProjetos(), MostrarAtividades(), BuscarProjeto(), BuscarAtividade().

Vantagem:

Mais simplicidade em implementar esses quatro métodos.

Desvantagem:

Não pensei em nenhuma.

Polimorfismo: Não utilizei.

Tratamento de Exceções:

Motivação: o usuário lê um inteiro que vai decidir qual função chamar.

Solução: chama a exception que verifica se o valor dado é um inteiro.

Vantagem: o programa não para quando um valor diferente de inteiro é passado para uma entrada que só recebe inteiro.

Desvantagem: nenhuma.

Extensibilidade: Não utilizei.

Reuso:

Motivação: Achei necessário para classes que definiam os usuários e os trabalhos.

Solução: utilizei herança para reutilizar atributos e métodos.

Vantagem: código menor e menos clonagem de código.

Desvantagem: nenhuma.