**Projeto-de-Software**

Descrição Projeto Isoccer

⦁ Funcionalidades:

O isoccer conta com 4 funcionalidades principais, as quais serão explicitadas e explicadas.

**-Adição de funcionários:** É solicitado ao administrador que seja digitado o tipo de funcionário, dentre os disponíveis a ser criado, assim, com o auxilio da interface, é chamada a função “creat\_funcionario”, a qual sua implementação encontra-se dentro da classe correspondente ao tipo de funcionário desejado. Posteriormente, na função, é associado ao funcionário todos os seus dados básicos e após isso, o mesmo é adicionado na em um array list de funcionários.

**-Adição de sócio torcedores/Edição de taxas:** Casos seja o primeiro acesso, será solicitado que seja inserido os valores das taxas associados a cada tipo de torcedor. Feito isso, para adicionar um torcedor, deve-se ser informado o seu tipo(júnior, sênior, elite), assim, na função “creat\_torcedor” localizada na classe “torcedor”, são associados todos os dados básicos do indivíduo, como também a definição do status adimplente ou inadimplente, feito isso, o torcedor é adicionado a um array list de torcedores. Além disso, pode-se editar as taxas dos torcedores, escolhendo esta opção, tal operação ocorre na função “edit\_taxas”, localizada na classe “torcedor”, na qual define as novas taxas a serem cobradas.

**-Gerenciar Recursos físicos:** Possibilidade de gerenciamento dos recursos físicos. Os mesmos podem ser criados e acumulados aos bens do clube, assim como detalhado seus atributos e estados de disponibilidade. Existem como bens físicos: Ônibus, estádio e centros de treinamento.

\*ônibus: é possível adicionar/editar um transporte ou verificar sua disponibilidade. Caso seja solicitado adicionar, tal operação ocorre na função “creat\_onibus” localizada na classe “ônibus”, na qual é associado os dados básicos do transporte e, após isso, o mesmo é associado a um array list dos transportes pertencentes ao clube. Já caso seja solicitado editar, esta operação ocorre em “edit\_disp”, pertencente a classe “ônibus”, em que o administrador escolhe o modelo dentre os existentes, utilizando a função “search\_onibus” presente em “ônibus”, e digita sua nova disponibilidade. Escolhendo verificar disponibilidade, são exibidos ao administrador todos os transportes disponíveis presentes no array list de transportes naquele momento, contando também com o apoio da função “check\_disp”, presente na classe “ônibus”, através de uma simples navegação na lista de transportes.

\*estádio: é possível adicionar/editar um estádio ou verificar sua disponibilidade. Caso seja solicitado adicionar, tal operação ocorre na função “creat\_estadio” localizada na classe “estádio”, na qual é associado os dados básicos do estádio e, após isso, o mesmo é associado a um array list dos estádios pertencentes ao clube. Já, caso seja solicitado editar, esta operação ocorre em “edit\_disp”, pertencente a classe “estádio”, em que o administrador escolhe o nome dentre os existentes, utilizando a função “search\_estadio” presente em “estádio”, e digita sua nova disponibilidade. Escolhendo verificar disponibilidade, são exibidos ao administrador todos os estádios presentes no array list de estádios, disponíveis naquele momento, contando também com o apoio da função “check\_disp”, presente na classe “estádio”, através de uma simples navegação na lista de estádios.

\*Centro de treinamento(ct): é possível adicionar/editar um ct ou verificar sua disponibilidade. Caso seja solicitado adicionar, tal operação ocorre na função “creat\_ct” localizada na classe “ct”, na qual é associado os dados básicos do ct e, após isso, o mesmo é associado a um array list dos centros de treinamento pertencentes ao clube. Já, caso seja solicitado editar, esta operação ocorre em “edit\_disp”, pertencente a classe “ct”, em que o administrador escolhe o nome dentre os existentes, utilizando a função “search\_centro” presente em “ct”, e digita sua nova disponibilidade. Escolhendo verificar disponibilidade, são exibidos ao administrador todos os centros disponíveis presentes no array list de centros de treinamento, contando também com o apoio da função “check\_disp”, presente na classe “ct”, através de uma simples navegação na lista de centros.

**-Exibir Relatório:** Em tal funcionalidade é possível que o administrador solicite um relatório apenas dos funcionários, apenas dos recursos físicos, apenas do torcedor ou de todos juntos. Levando em consideração que a única diferença da aplicação individual para todos juntos é o momento em que se chama as funções, será explicado o momento do relatório de todos juntos, o qual engloba todas as funções usadas.

\*todos juntos:

-Técnico: o relatório envolvendo o técnico é feito através da função “relat\_tecnico”, presente na classe “técnico”, a qual trata de percorrer a lista de funcionários e mostrar ao administrador todos os dados dos funcionários do tipo “técnico”.

-Jogadores: o relatório envolvendo o jogador é feito através da função “relat\_jogador”, presente na classe “jogador”, a qual trata de percorrer a lista de funcionários e mostrar ao administrador todos os dados dos funcionários do tipo “jogador”. Feito isso, é sugerido mostrar a lista de jogadores aptos e inaptos para jogar, ação realizada pela função “ver\_apt\_inapt” presente na classe “jogador”.

-Serviços gerais: o relatório envolvendo o funcionários de serviços gerais, por compartilharem a herança da classe “funcionário” e terem os mesmos dados, é feito através da função “relat\_func\_gerais”, presente na classe “funcionário”, a qual trata de percorrer a lista de funcionários e mostrar ao administrador todos os dados dos funcionários que tenham tipo diferentes de “técnico” e “jogador”.

-Transportes: o relatório envolvendo os transportes é feito através da função “relat\_transportes”, presente na classe “ônibus”, a qual trata de percorrer a lista de transportes e mostrar ao administrador todos os dados dos transportes, assim como sua disponibilidade.

-Estádios: o relatório envolvendo os Estádios é feito através da função “relat\_ estadio”, presente na classe “estádio”, a qual trata de percorrer a lista de Estádios e mostrar ao administrador todos os dados dos Estádios, assim como sua disponibilidade.

-Centros de treinamento: o relatório envolvendo os Centros é feito através da função “relat\_ cts”, presente na classe “ct”, a qual trata de percorrer a lista de centros e mostrar ao administrador todos os dados dos Centros, assim como sua disponibilidade.

-Dados básicos da torcida: Trata-se apenas de deixar explícito ao administrador a quantidade de torcedores, e quantos destes estão adimplentes e inadimplentes, tal operação é feita simplesmente navegando a lista de torcedores e marcando quantos são adimplentes e quanto são inadimplentes.

-Torcedores: trata-se de uma exibição individual de cada torcedor, navegando no array list de torcedores, é exibido os dados de cada um, bem como sua disponibilidade.

⦁Classes:

* Torcedor:
  + Motivação: Deixar os torcedores separados de funcionários, para trata-los e organizar seus atributos;
  + Solução: Organizou os torcedores em tipos, deixando bem mais fácil a questão de editar as taxas, criar um torcedor e exibir os dados básicos e individuais;
  + Vantagens: Conseguir armazenar e utilizar a classe da forma necessária e de forma breve, compactando as funções necessárias na classe e seus atributos;
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* CT:
  + Motivação: Deixar os recursos físicos separados de funcionários e torcedores, para trata-los e organizar seus atributos;
  + Solução: Organizou o ct de forma única, deixando bem mais fácil a questão de editar a disponibilidade, criar um ct, e exibir o relatório, assim como como buscar pelo mesmo, foram ações facilitadas pelas funções organizadas em tal classe;
  + Vantagens: Conseguir armazenar e utilizar a classe da forma necessária e de forma breve, compactando as funções necessárias na classe e seus atributos;
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Ônibus:
  + Motivação: Deixar os recursos físicos separados de funcionários e torcedores, para trata-los e organizar seus atributos;
  + Solução: Organizou os transportes de forma única, deixando bem mais fácil a questão de editar a disponibilidade, adicionar um transporte, e exibir o relatório, assim como como buscar pelo mesmo, foram ações facilitadas pelas funções organizadas em tal classe;
  + Vantagens: Conseguir armazenar e utilizar a classe da forma necessária e de forma breve, compactando as funções necessárias na classe e seus atributos;
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Estádio:
  + Motivação: Deixar os recursos físicos separados de funcionários e torcedores, para trata-los e organizar seus atributos;
  + Solução: Organizou os estádios de forma única, deixando bem mais fácil a questão de editar a disponibilidade, adicionar um estádio, e exibir o relatório, assim como como buscar pelo mesmo, foram ações facilitadas pelas funções organizadas em tal classe;
  + Vantagens: Conseguir armazenar e utilizar a classe da forma necessária e de forma breve, compactando as funções necessárias na classe e seus atributos;
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.

⦁Distribuição dos métodos:

* Motivação: A principal motivação esteve intrinsicamente relacionada a que objeto tal método se refere ou atua sobre ele, cada método que tem como parâmetros objetos de determinadas classes, foram organizados na sua determinada classe afim de facilitar a procura desses e sua aplicação.
* Solução: resolveu o problema do método ficar "perdido" e dificultar uma posterior procura, evitando que fique "bagunçado"
* Vantagens: encontrar mais rapidamente o método, facilitando a manutenção entre outras atividades no mesmo.
* Desvantagens:

Obs.: Detalhes da relação de cada método e sua aplicação e distribuição nas classes das quais fazem parte, estão organizados ao longo do relatório, referente a cada classe analisada.

⦁Herança:

* Presidente:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do presidente, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao presidente do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o presidente, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Médico:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do médico, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao médico do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o médico, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Técnico:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do técnico, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao técnico do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o técnico, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Preparador:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do preparador, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao preparador do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o preparador, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Motorista:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do motorista, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao motorista do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o motorista, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Cozinheiro:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do cozinheiro, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao cozinheiro do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o cozinheiro, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Advogado:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do advogado, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao advogado do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o advogado, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Segurança:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo apenas as variáveis já existentes pela sua classe mãe, assim, sua criação foi necessária pela questão de armazenar os dados do segurança, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao segurança do time, assim como contém uma função para criar o mesmo, armazenar os dados e lança-lo no array list de funcionários. Poupou tempo de criação das variáveis presentes na superclasse.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o segurança, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.
* Jogador:
  + Motivação: Trata-se de uma extensão da classe Funcionário, contendo as variáveis já existentes pela sua classe mãe como também a posição e aptidão do jogador, assim, sua criação foi necessária pela questão de armazenar os dados do jogador, para posteriormente lançar o conteúdo estabelecido no array list de funcionários.
  + Solução: Armazenar as informações ligadas ao Jogador do time, assim como contém uma função para criar o Jogador, para exibir ao administrador os dados do mesmo, e outra a qual exibe os jogadores aptos e inaptos. E assim armazenar os dados e lançar os dados do funcionário.
  + Vantagens: Conseguir armazenar e utilizar a função da forma necessária e de forma breve, graças a herança de funcionário e sua simples função de criar o funcionário, sendo de fácil uso, criação e manutenção.
  + Desvantagens: Aumento no número de classes, podendo deixar o desenvolvedor confuso em alguns momentos.

⦁Abstrata:

**-Funcionário:**

* + Motivação: tal classe teve como motivação a necessidade de agrupar todas as variáveis referentes aos dados dos funcionários, deixando tudo de forma mais “amarrada”, assim como, nela encontra-se a função de relatório dos funcionários gerais, comum a grade maioria dos funcionários. Além disso, trata-se de uma classe abstrata, já que nenhuma atividade será de fato executada em tal classe, apenas declarando as variáveis super.
  + Solução: Usar este tipo de classe me trouxe como solução evitar o conflito entre as diversas variáveis que teria sido necessário caso não existisse a mesma, evitando a repetição de muitos trechos de código. Fez com que as informações não ficassem dispersas no código, nem “soltas” na função principal como também em outras classes, as quais possuem outro objetivo.
  + Vantagens: Foi poupado tempo durante a criação do código, pois como dentre as soluções estão a herança. Trouxe organização ao código, deixando-o mais conciso e fácil de acessar e encontrar informações, facilitando possíveis manutenções;
  + Desvantagens: por se tratar de uma classe abstrata, para editar a informação das variáveis torna-se mais complexo pela diferença que pode surgir no uso da variável pelas classes filhas.

⦁Interface:

-Interface\_func

* Motivação: declaração dos métodos em comun usado nas classes “ônibus”, “estádio”, “ct”, método “creat\_funcionario”.
* solução: mantem a organização e facilita manutenções futuras, principalmente para quem pega seu projeto pela primeira vez;
* vantagens: organização e facilidade para encontrar funções que atuam nas classes, facilitando manutenções e analises no código, deixando explicita a função usada em comum.
* desvantagens: caso seja necessário, fica proibido uma declaração previa do corpo da função.

⦁Polimorfismo:

Ex.: método “creat\_funcionário” na “Interface\_func”

* Motivação: definição de método com o mesmo nome, porém atuando em diferentes classes para objetos diferentes.
* solução: manteve a organização e facilitou manutenções futuras, principalmente irá ajudar caso alguém pegue o projeto pela primeira vez, minimizando informações e o que aquilo esta responsável;
* vantagens: organização e facilidade para saber o que tais funções fazem, principalmente na aplicação do polimorfismo em métodos.
* desvantagens: pode causar confusão caso sege mal implementada.

⦁Tratamento de exceções:

* Motivação: evitar com que o programa pare inesperadamente, e dessa forma perca todas as entradas já atribuídas, assim como mostrar para um usuário leigo o erro que aconteceu, foi aplicado no código posteriormente a entrada de números inteiros, pois era causado erros no momento em que símbolos diferentes de inteiros eram digitados, pode ser encontrado um exemplo logo no início do projeto, quando se é pedido a senha para o administrador realiza o login;
* solução: faz com que o programa não pare em alguma exceção, sendo possível corrigir e seguir em frente com o processo, não havendo prejuízos;
* vantagens: organização e facilidade para lhe dar com os possíveis problemas que podem acontecer, como por exemplo nas entradas de símbolos diferentes do tipo especificado, mantendo seu programa ativo e possível de execução contínua, não perdendo tempo e dando ao usuário passos de como proceder na situação.
* desvantagens:

⦁Extensibilidade:

- Como foi visto em heranças temos 9 classes onde se aplicam o conceito, sendo assim é evidente a aplicação de extensibilidade.

⦁Reuso:

-Função search\_onibus, search\_centro, search\_estadio

* Motivação: foram funções solicitadas em mais de uma função, aos quais foram importantes para procurar o objeto desejado.
* solução: criação de funções por exemplo, fizeram com que o código não precisasse sempre ser reescrito para fazer certa operação;
* vantagens: evitar linhas desnecessárias e fazer com que o código fique mais limpo, sendo assim facilitando manutenções e melhorando a organização das classes;
* desvantagens: pode-se sobrepor alguma informação;