Aula prática 04 – Estruturação do sistema em Camadas

Você foi contratado para criar um sistema de gerenciamento pessoal de corridas orientado a objetos, utilizando Java. O sistema permitirá guardar as seguintes informações de uma corrida: identificador (String), descrição (String), data e hora de início e fim da corrida (use java.time para esses atributos), distância percorrida em km (float), velocidade média em km/h (float), percurso executado (objeto do tipo Percurso que você irá definir). A classe Corrida, apresentará um método que permita transformar a velocidade média, expressa em km/h, em um ritmo médio em expresso em min/km (usualmente chamado de *pace*).

Para cada corrida, o percurso executado deve ser armazenado com um identificador (String) e uma URL que aponta para algum serviço de mapas como o Google Maps, por exemplo. Para facilitar, a URL deve ser somente um objeto do tipo String.

Você deve criar um Repositório de corridas que permita a execução das operações básicas de persistência: cadastrar, recuperar, atualizar e deletar uma corrida (CRUD). Antes de criar o repositório, defina com cautela todas as assinaturas dos métodos a serem implementados pelo repositório. A implementação pode usar ArrayList em seu corpo.

Implemente o padrão de projeto Singleton na classe do repositório.

Implemente uma classe chamada de `GerenciadorPessoalCorridas` que utilize os serviços do repositório. Para isso, defina um atributo nesta classe do tipo da classe do Repositório que você criou anteriormente e instancie este atributo no construtor. Este gerenciador de corridas deve ter os seguintes métodos:

- **void comecarACorrer()**: instancia uma corrida com o horário atual. Uma corrida não pode ser iniciada se alguma outra corrida estiver em andamento.
- **void pararDeCorrer()**: para a corrida que estiver em andamento e configura o seu horário final para o atual no momento da execução do método. Uma corrida não pode ser finalizada se não tiver sido inicializada. Os outros valores de atributos da corrida podem ser gerados aleatoriamente no momento da finalização da mesma.
- **List<Corrida> listaCorridasExecutadas()**: lista todas as corridas executadas, ordenadas por data e hora. Observe que, para imprimir os dados de uma corrida, você pode sobrescrever o método toString da classe Corrida para organizar todas as informações e retornar como String.

Crie uma classe Principal, que contenha um método main, e permita que o usuário final possa executar o programa, podendo iniciar e parar corridas, além de poder listar todas as corridas executadas, tudo através de um menu textual. Observe que esta classe, estando relacionada com aspectos da camada de interface com o usuário (GUI), deve deixar o uso do programa confortável, com mensagens de erro significativas que ajudem a entender o possível problema que venha a ocorrer.