Parte 2 - 12.5 valores

Considere que se pretende criar um sistema de registo de actividades desportivas, a exemplo do que foi feito na Ficha 6 das aulas práticas. A classe `Fitness` guarda a informação dos seus utilizadores e estes registam as actividades que efectuaram.

A classe `Atividade` representa o topo da hierarquia das actividades, sendo que neste momento se apresenta o código das subclasses `Canoagem` e `Corrida`. Ambas as classes declaram as suas variáveis de instância e implementam, da forma adequada, o método `caloriasGastas(Utilizador u)`.

Considere que se pretende acrescentar agora a possibilidade de se registarem planos de treino para os utilizadores que assim o desejem. Os utilizadores poderão ter registo de actividades associadas a um plano de treino ou actividades sem associação (actividades isoladas). Não é obrigatório que todos os utilizadores tenham um plano de treino definido.

Um *Plano de Treino* de um utilizador deverá ser composto por uma sequência de *sessões*. Cada *Sessão* terá uma sequência de actividades, sendo que neste momento está definido que não será possível a um utilizador ter mais sessões do que dias da semana (só poderá ter no máximo 7 sessões no plano de treino). Não existe contudo restrição ao número de actividades que fazem parte de uma sessão.

Considere os seguintes excertos de código:

```
public class Fitness {
    private Map<String, Utilizador> users;
      ...
}

public class Utilizador {
    private String email;  // chave que identifica um utilizador
    ...
    private Map<String, Atividade> atividades; // codigo actividade --> Atividade
    ...
}

public abstract class Atividade implements Serializable {
    private String codigo;
    private String descricao;
    private LocalDate data;
    private int duracao;
    private Utilizador user;
    ...

    public abstract double caloriasGastas(Utilizador u);
    ...
}


public class Canoagem extends Atividade implements Comparable<Canoagem>, Serializable {
  private double vento;
  private double distancia;
   ...
}

public class Corrida extends Atividade implements Comparable<Corrida>, Serializable {
   private double distancia;
   private double altimetria;
   ...
}
```

Considere que a estratégia de associação relativa aos utilizadores e planos de treino é de **composição**, mas tal já não é necessário na relação entre a `Sessão` e a `Atividade` a que se refere.

Antes de começar a responder **leia todas as questões desta parte** e assuma, para as perguntas seguintes, que os métodos usuais (get, set, equals, clone, hashcode, ...) estão disponíveis, a menos que sejam solicitados, e responda às questões:

Questão 6

1) Efectue a declaração das variáveis e da(s) classe(s) necessária(s) para criar a noção de *Plano de Treino* 2) **justifique brevemente** a sua resposta, nomeadamente referindo onde se devem alterar e acrescentar classes.

3) Codifique também o método `public List<Utilizador> utilizadoresComPlano`, da classe `Fitness`, que devolve uma lista com todos os utilizadores que possuem um plano de treino definido.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 7

Codifique o método `public void adicionaActividade(Atividade a, int sessaoTreino) throws...`, da classe `Utilizador`, que adiciona uma actividade ao plano de treino do utilizador no final da sessão número *sessaoTreino*. Considere que deverá validar se o plano de treino existe e se tem essa sessão (considere que no plano de treino a sessão de domingo está na posição 0).

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 8

Codifique o método `public Map<String,Integer> caloriasPorUtilizador()`, da classe `Fitness`, que devolve uma relação entre o código do utilizador e o número de calorias que ele dispendeu nas actividades que executou.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 9

Considere que a aplicação Fitness tem uma vertente de *gamificação* e que por cada metro percorrido atribui pontos. Esses pontos só são atribuídos nas actividades que possuam distância (neste caso apenas a `Corrida` e a `Canoagem`) e cada uma das actividades terá uma fórmula diferente para calcular o valor de pontos resultante. Por exemplo na corrida o número de pontos é multiplicado pela distância acumulada e na canoagem é 1.5 vezes o valor do vento. **1)** Desenvolva a interface `TemMetros`, que deverá possibilitar especificar a API necessária para definir o total de pontos a atribuir por cada metro e a calcular o total de pontos que uma determinada actividade originou, e **2)** Actualize a classe `Corrida` por forma a que implemente correctamente a interface.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 10

Considere agora que se pretende criar um *plano de treino de alta intensidade* que é composto apenas por actividades que se classificam como sendo de `AI` (alta intensidade). Tendo isso em conta, **1)** explique o que alteraria na sua arquitectura para garantir este requisito e **2)** codifique o método `public void adicionaActividade(...) throws...` que permite adicionar uma actividade de alta intensidade a um plano de treino, efectuando as validações necessárias.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*