Nome:_____ **Nº:**_____ **Curso:**_____

Teste de Programação Orientada aos Objectos **(A)**

MiEI e LCC - DI/UMinho

21/05/2021
Duração: **2h**

Leia o teste com muita atenção antes de começar
Assuma que gets e sets estão disponíveis, salvo se forem explicitamente solicitados.
Na Parte I não existem erros sintácticos propositados.

Parte I - 7.5 valores

1. Considere as seguintes definições das classes Jogador e Equipa:

```
public abstract class Jogador {
        private String numero;
        private String nome;
        private List<Integer> golos; // lista com os golos em cada jogo
        //...
        public double mediaGolos() {
            //...
        }
        //...
}

public class Equipa {
        private String clube;
        private Map<String,Jogador> jogadores;
        //...
}
```

   Sabendo que o método `mediaGolos(String numJogador)` da classe `Equipa` (que calcula a média de golos do jogador indicado) não deverá produzir um valor para a média caso o número de jogador indicado não exista, indique qual das seguintes implementações do método considera correcta:

   ◯
```
public double mediaGolos(String num) {
        double m = 0;
        Jogador jog = this.jogadores.get(num);
        if (jog != null) {
                m = jog.mediaGolos();
        }
        return m;
}
```

   ◯
```
public double mediaGolos(String num) throws JogadorNaoExisteException {
        return this.jogadores.get(num).mediaGolos();
}
```

◯
```
public double mediaGolos(String num) throws JogadorNaoExisteException {
        Jogador jog = this.jogadores.get(num);
        if (jog == null) {
                throw new JogadorNaoExisteException(num);
        }
        return jog.mediaGolos();
}
```

◯
```
public double mediaGolos(String num) {
        Jogador jog = this.jogadores.get(num);
        if (jog == null) {
                throw new JogadorNaoExisteException(num);
        }
        return jog.mediaGolos();
}
```

◯ Nenhuma das implementações é válida pois o método `mediaGolos()` na classe `Jogador` não é abstracto.

2. Considere que a classe Convocatoria foi declarada do seguinte modo:

```
public abstract class Convocatoria {
       private String codJogo;
       private LocalDateTime data;
       private List<Jogador> convocados;
    ...
}
```

Prestando atenção ao encapsulamento (em particular às noções de composição e agregação), indique quais dos seguintes pares de métodos estão correctamente implementados (**atenção**: indique **todos** os pares que considera correctos):

◯
```
public void setConvocados(List<Jogador> conv) {
        this.convocados = conv;
}

public List<Jogador> getConvocados() {
        return this.convocados;
}
```

◯
```
public void setConvocados(List<Jogador> conv) {
        this.convocados = new ArrayList(conv);
}

public List<Jogador> getConvocados() {
        return this.convocados.stream()
                              .map(Jogador::clone)
                              .collect(Collectors.toList());
}
```

◯
```
public void setConvocados(List<Jogador> conv) {
        this.convocados = conv.stream()
                              .map(Jogador::clone)
                              .collect(Collectors.toList());
}

public List<Jogador> getConvocados() {
        return new ArrayList(this.convocados);
}
```

◯
```
public void setConvocados(List<Jogador> conv) {
        this.convocados = conv.stream()
                               .map(Jogador::clone)
                               .collect(Collectors.toList());
    }

    public List<Jogador> getConvocados() {
        List<Jogador> conv = new ArrayList();

        for(Jogador j: this.convocados) {
               conv.add(j.clone());
        }
        return conv;
    }
```

◯
```
public void setConvocados(List<Jogador> conv) {
        this.convocados = new ArrayList(conv);
    }

    public List<Jogador> getConvocados() {
        return this.convocados.stream().collect(Collectors.toList());
    }
```

3. Considere as seguintes definições:

```
public interface I {
    public int miA();
    public int miB();
}

public class A implements I {
    ...
    public A() { .. }
    public int m1() { ... }
    public int m2() { ... }
    public int miA() { ... }
}

public class B extends A implements I {
    ...
    public B() { ... }
    public int miB() { ... }
}
```

Qual das seguintes afirmações é válida:

◯ A definição da interface I está errada pois os seus métodos têm que ser abstractos.
◯ A classe A não está correcta pois não pode definir o método miA() da interface I.
◯ A classe B não está correcta pois não define a implementação do método miA().
◯ A seguinte expressão é válida: `I i = new B();`

4. Relembre a classe DriveIt desenvolvida nas aulas, em que se armazenam Veículos (indexados pela matrícula) na seguinte estrutura:

```
private Map<String, Veiculo> viaturas;
```

Selecione o método que correctamente devolve os veículos de uma dada marca, ordenados alfabeticamente por matrícula:

◯
```
public Iterator<Veiculo> veiculosDaMarca(String marca) {
    Iterator<Veiculo> r = this.viaturas.values().iterator();
    while(r.hasNext()) {
        Veiculo v = r.next();
        if (!v.getMarca().equals(marca)) r.remove();
    }
    r.sort((v1, v2) -> v1.getMatricula().compareTo(v2.getMatricula()));
    return r;
}
```

◯
```
public List<Veiculo> veiculosDaMarca(String marca){
    Comparator<Veiculo> comp =
        (v1, v2) -> v1.getMatricula().compareTo(v2.getMatricula()));
    return this.viaturas.stream()
                  .map(Veiculo::clone)
                  .filter(v -> !v.getMarca().equals(marca))
                  .sorted(comp)
                  .collect(Collectors.toList());
}
```

◯
```
public Set<Veiculo> veiculosDaMarca(String marca){
    TreeSet<Veiculo> r = new TreeSet<>(
              (v1, v2) -> v1.getMatricula().compareTo(v2.getMatricula()));
    for (Veiculo v : this.viaturas.values()) {
        if (v.getMarca().equals(marca)) r.add(v.clone());
    }
    return r;
}
```

◯
```
public Set<Veiculo> veiculosDaMarca(String marca){
    List<Veiculo> r = new List<>();
    for (Map.Entry<String, Veiculo> e : this.viaturas.entrySet()) {
        Veiculo v = e.getValue();
        if (v.getMarca().equals(marca)) r.add(v.clone());
        r.sort((v1, v2) -> v1.getMatricula().compareTo(v2.getMatricula()));
    }
    return r;
}
```

5. Considere a seguinte estrutura de dados usada para representar uma colecção de videos numa plataforma de streaming. A cada autor está associada uma colecção com todos os seus vídeos, que por sua vez é indexada por um código único do vídeo.

```
public Map<String, Map<String, Video>> videos;
```

Considerando tudo o que aprendeu sobre o tratamento de erros, selecione o método que mais correctamente implementa a obtenção de um vídeo, dado o nome do utilizador e o código do vídeo:

◯
```
public Video getVideo(String user, String codVideo) {
    return this.videos.get(user).get(codVideo).clone();
}
```

◯
```
public Video getVideo(String user, String codVideo) {
    if (!this.videos.containsKey(user))
           throw new Exception("User " + user + " Inexistente");
    if (!this.videos.get(user).containsKey(codVideo))
```

```
            throw new Exception("Video " + codVideo + " Inexistente");
        return this.videos.get(user).get(codVideo).clone();
    }
```

◯
```
    public Video getVideo(String user, String codVideo) {
        Video v;
        try {
            v = this.videos.get(user).get(codVideo).clone();

        } catch (Exception e) {
            v = null;
        }
        return v;
    }
```

◯
```
    public Video getVideo(String user, String codVideo)
                            throws UserInexistenteException,
                                   VideoInexistenteException {
        if (!this.videos.containsKey(user))
            throw new UserInexistenteException("User "+user+" Inexistente");
        if (!this.videos.get(user).containsKey(codVideo))
            throw new VideoInexistenteException("Video "+codVideo+" Inexistente");
        return this.videos.get(user).get(codVideo).clone();
    }
```

Nome:_____ **Nº:**_____ **Curso:**_____

Parte II - 12.5 valores

Considere o exercício dos Smart Devices que foi resolvido numa das aulas práticas. De acordo com o exercício existem actualmente dois tipos de `SmartDevice`, as colunas de som (as `SmartSpeaker`) e as lâmpadas (as `SmartBulb`), com as definições que se apresentam:

```
public class SmartDevice {

    private String id;
    private boolean on;
    private double consumoPorHora;
    private LocalDateTime inicio;

    ...
    public SmartDevice( String id, double consumoPorHora) {
       this.id = id;
       this.on = false;
       this.consumoPorHora = consumoPorHora;
    }


    // devolve o consumo desde o inicio
    public double totalConsumo() {...}
    ...

    //liga o device. Se for a primeira vez inicializa o tempo de inicio
    public void turnOn() {
        this.on = true;
        if (this.inicio == null)
           this.inicio = LocalDateTime.now();
    }
}


public class SmartBulb extends SmartDevice {
    public static final int WARM = 2;
    public static final int NEUTRAL = 1;
    public static final int COLD = 0;
    private int tone;

    public SmartBulb(String id, int tone, double consumoPorHora) {
       super(id, consumoPorHora);
       this.tone = tone;
    }

    ...
    public void setTone(int t) {
        if (t>WARM) this.tone = WARM;
        else if (t<COLD) this.tone = COLD;
        else this.tone = t;
    }

    public int getTone() {
        return this.tone;
    }
}

public class SmartSpeaker extends SmartDevice {
    public static final int MAX = 20; //volume maximo da coluna
```

```
    private int volume;
    private String channel;

    public SmartSpeaker(String id, String channel, double consumoPorHora) {
       super(id, consumoPorHora);
       this.channel = channel;
       this.volume = 10;
    }
   ...
   ...
}
```

Considere que se pretende implementar uma classe `CasaInteligente` que guarda a informação dos dispositivos existentes na casa e regista também para cada divisão da casa (identificadas por Strings como "Sala Jantar", "Quarto", "Escritório", etc.) os dispositivos que nelas se encontram.

Resolva os seguintes exercícios:

Nome:_____ **Nº:**_____**Curso:**_____

6. Efectue a declaração das variáveis de instância de `CasaInteligente` e codifique o construtor que recebe uma coleção de `SmartDevice` e que assume que estamos numa estratégia de composição, `public CasaInteligente(Collection<SmartDevice> devices)`.

Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

7. Desenhe o Diagrama de Classes da solução.
   Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

8. Codifique o método `public void remove(String id) throws...`, que remove completamente do sistema o dispositivo cujo identificador é passado por parâmetro (codifique também a excepção).

   Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

9. Codifique o método `public Iterator<SmartDevice> devicesPorConsumoCrescente()`, que devolve um iterador com ordenação crescente por consumo (deve codificar e utilizar a ordem natural dos `SmartDevice`).

   Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

10. Forneça a implementação para o método `public String divisaoMaisEconomica()`, que determina a divisão da casa que apresenta o menor consumo. Se duas divisões apresentarem o mesmo consumo então deverá ser devolvida a divisão cuja designação tem o maior valor alfabético.

    Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

11. Considere que se pretende acrescentar um novo tipo de lâmpada que permita regular a intensidade da sua luz. A `SmartBulbDimmable` quando é ligada pela primeira vez fica com a intensidade da luz a 50% e gasta também metade do consumo anunciado. Crie esta classe, identificando as variáveis de instância necessárias, o construtor parametrizado e todos os métodos herdados que necessitam de ser reescritos.

    Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

12. Relembre a matéria das aulas teóricas e forneça uma implementação para o método da classe `CasaInteligente` que permita fazer alterações ao estado interno das `SmartBulbDimmable`. Esse método deve ter a assinatura `public void alteraInfo(Consumer<SmartBulbDimmable> bd)`. Forneça também a implementação para o `Consumer<SmartBulbDimmable>` que altera a luminosidade de uma `SmartBulbDimmable` para 25% do seu valor actual.

    Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

13. Codifique o método `public boolean apenasNumaDivisao()`, que dá true se não existir nenhum `SmartDevice` registado em mais do que uma divisão da casa.

    Resposta:

Nome:_____ **Nº:**_____**Curso:**_____

14. Codifique o método que grava num ficheiro de objectos, cujo nome é fornecido no parâmetro, todas os `SmartSpeaker` existentes na casa.

    ```
    public boolean gravaEmFichObjectos(String fich) throws FileNotFoundException, IOException
    ```

    Resposta: