+1/1/60+

**Nome:** .....

**Curso:** .....

**Número:**

| | | | | |
|---|---|---|---|---|
| ☐0 | ☐0 | ☐0 | ☐0 | ☐0 |
| ☐1 | ☐1 | ☐1 | ☐1 | ☐1 |
| ☐2 | ☐2 | ☐2 | ☐2 | ☐2 |
| ☐3 | ☐3 | ☐3 | ☐3 | ☐3 |
| ☐4 | ☐4 | ☐4 | ☐4 | ☐4 |
| ☐5 | ☐5 | ☐5 | ☐5 | ☐5 |
| ☐6 | ☐6 | ☐6 | ☐6 | ☐6 |
| ☐7 | ☐7 | ☐7 | ☐7 | ☐7 |
| ☐8 | ☐8 | ☐8 | ☐8 | ☐8 |
| ☐9 | ☐9 | ☐9 | ☐9 | ☐9 |

# Programação Orientada aos Objetos

## Teste

LEI/LCC, Universidade do Minho

20 de Maio, 2022 – Duração: 2h

**Instruções:** *Não se esqueça de preencher o nome, curso e número. Indique o número à direita, assinalando um dígito por coluna.*

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



## Parte 1 - 7.5 valores

## Parte 2 - 12.5 valores

Considere as seguintes definições de classes de uma aplicação que implementa uma loja de livros digitais. A aplicação da `LivrosDigitais` possui a informação dos utilizadores que nela estão registados e para cada utilizador é guardada a informação respeitante à colecção de livros que adquiriu. A informação dos livros indica as páginas lidas e por ler e consequentemente em qualquer altura sabe-se sempre qual é o sítio do livro que se está a ler.

Considere os seguintes excertos de código:

```
public class Livro implements Comparable<Livro>, Serializable {
  public String codISBN;         //código ISBN do livro
  private String nomeLivro;
  private String autor;
  private String editora;
  private List<Pagina> pagLidas; // páginas já lidas
  private List<Pagina> pagPorLer; //páginas ainda por ler.
                                  //o primeiro elemento é a página a ser lida no momento
  ....

  /* método que devolve a página com o número indicado */
  public Pagina devolvePag(int numPag) throws PagInexistenteException {
   Pagina res = null;
    int numLidas = this.pagLidas.size(); //número de páginas lidas
    int porLer = this.pagPorLer.size();

    if (numPag > numLidas+porLer)
      throw new PagInexistenteException(numLidas);
    if (numPag <= numLidas )
       res = this.pagLidas.get(numPag -1);
    else
       res = this.pagPorLer.get(numPag-numLidas -1);

    return res.clone();
  }

}

public class Pagina implements Comparable<Pagina>, Serializable {
private List<String> texto;

public Pagina() {
  this.texto = new ArrayList<>();
}
...
/* método que devolve uma formatação do texto */
public String reproduzPagina() {...}
}
```

```
public class PaginaComAudio extends Pagina implements Comparable<PaginaComAudio>,
                                                    Serializable {
   private String narrador;
   private List<Byte> som;
   ...

   public PaginaComAudio(List<String> texto, String narrador, List<Byte> audio) {
        super(texto);
    this.narrador = narrador;
    this.audio = new ArrayList<>(audio);
   }
   /* método que devolve uma formatação do texto e audio */
    public String reproduzPagina() {...}

}


public class Utilizador implements Serializable {

  private String numUser;
  private String nomeUser;
  private LocalDate dataAdesao;  // data de adesão do utilizador à aplicação
  ...


}
```

Assuma, para as perguntas seguintes, que os métodos usuais (equals, clone, hashcode, ...) estão disponíveis a menos que sejam solicitados e responda às questões:

**Questão 1**

Efectue a declaração das variáveis de instância de `LivrosDigitais` e complete a declaração das variáveis de instância de `Utilizador`. Codifique o construtor parametrizado de `Utilizador` que recebe uma série de instâncias de `Livro` e que assume que estamos numa estratégia de composição, `public Utilizador(String numUser, String nomeUser, Iterator<Livro> livros)`.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1



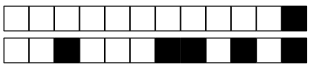
**Questão 2** Desenhe o Diagrama de Classes. ☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1

**Questão 3** Codifique o método `public void avancaPags(String codISBN, int n) throws...`, da classe `Utilizador`, que avança `n` páginas na leitura desse livro.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1

**Questão 4** Codifique o método `public Livro livroMaisLido()`, que determina o livro mais lido. Em caso de existir mais do que um livro candidato deverá ser devolvido aquele que seja alfabeticamente maior. O livro mais lido é aquele que registar mais páginas lidas.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1





**Questão 5**

Codifique o método `public Map<String,List<Livro» livrosPorEditora()`, da classe `LivrosDigitais`, que para cada nome de editora associa a lista dos livros dessa mesma editora.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1

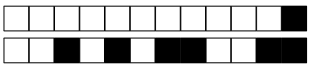
**Questão 6**

Considere que pretendemos adicionar aos livros um novo tipo de páginas que possuam texto, audio e vídeo. Temos, no entanto, uma implementação já existente da classe `PaginaMultimedia`, com a seguinte declaração:

```
public class PaginaMultimedia {
  private List<String> texto;
  private List<Byte> audio;
  private List<Byte> video;

  public PaginaMultimedia(List<String> texto, List<Byte> audio, List<Byte> video) {
    this.texto = texto;
    this.audio = audio;
    this.video = video;
  }

/**
 * método que devolve uma formatação do texto, audio e vídeo.
 * Está devidamente implementado.
 */
public String fazPagina() {
...
}

}
```

Esta é uma classe já antiga e não podemos alterar o código dela mas queremos aproveitar o comportamento que ela apresenta e especialmente o resultado do método `fazPagina` que devolve uma representação de uma página em que está colocado o texto, o audio e o vídeo. Queremos evitar ter que desenvolver uma classe de raíz e implementar novamente o método que faz a reprodução de uma página com texto, audio e vídeo.

Diga como é que podemos compatibilizar esta classe com o resto das classes existentes, mostre o que é preciso alterar ou criar, e codifique o método `public List<String> reproduzLivros()`, da classe `Utilizador`, que fornece a reprodução de todos os livros existentes.





☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1

Para o seu exame, use preferencialmente documentos compilados do auto-multiple-choice.