Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada

Programação Orientada a Objeto

24. Manipulação de Arquivos (Prática)

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

2° Semestre de 2020

Objetivos

- Exercitar os conceitos de POO vistos nas aulas anteriores!
  - Escrevendo, compilando e interpretando (executando) aplicações em Java;
  - Leitura e escrita de arquivos (e serialização).

Exercício #1 Parte #0

- Digitar textos eficientemente (corretamente e de forma rápida) ajuda a aumentar a produtividade daqueles que trabalham e/ou estudam na frente de um computador.
- Vamos implementar uma aplicação que nos "motiva" a digitar de forma mais eficiente, mantendo um *ranking* das três maiores taxas de digitação (em "caracteres por segundo"). Essa forma de engajamento é chamada de *gamificação*.
- Com essa abstração em mente [...]

Imagens livres de https://pixabay.com/

Exercício #1 Parte #1

- Faça o *download* dos arquivos **bancodefrases.poo** e **top3ranking.poo**;
- Esses dois arquivos nos ajudam a implementar dois conceitos importantes de nossa a aplicação (discutidos melhor na sequência);
- Nossa aplicação terá, ao menos, três classes:
  - `Jogador`,
  - `ManipuladorSerializaveis` e
  - `Aplicacao`.

Exercício #1 Parte #2

- A classe `Jogador` tem um `nome` (`String`) e um `score` (`double`);
- A classe `Jogador` tem dois construtores: um que recebe apenas um valor para `nome` e o outro que recebe valores para `nome` e `score`;
- A classe `Jogador` tem `setScore(double score)` e `getScore()`;
    - O *setter* deve manter o maior valor entre `score` e `this.score` (arredondado em duas casas decimais);
- A classe `Jogador` sobrescreve o método `toString` e retorna o `nome` do jogador e, entre parênteses, seu `score` - acompanhado de “c/s”;
    - “c/s” significa, no escopo da nossa aplicação, “caracteres por segundo”.

Exercício #1 Parte #2

- A classe `Jogador` tem um `nome` (`String`) e um `score` (`double`);
- A classe `Jogador` tem dois construtores: um que recebe apenas um valor para `nome` e o outro que recebe valores para `nome` e `score`;
- A
  - o
  - decimais);
- A classe `Jogador` sobrescreve o método `toString` e retorna o `nome` do jogador e, entre parênteses, seu `score` - acompanhado de "c/s";
  - "c/s" significa, no escopo da nossa aplicação, "caracteres por segundo".

A classe `Jogador` deve ser serializável!
Os nomes dos métodos e atributos **devem** estar conforme indicado!

Exercício #1 Parte #2

- A classe `ManipuladorSerializaveis` tem apenas dois métodos estáticos que escrevem/leem objetos serializáveis para/de arquivos;
- O método `serializa` recebe `nomeArquivo` (`String`) e `objeto` (`Object`) e retorna `void`;
  - O método deve criar, instanciar e manipular um `FileOutputStream` e um `ObjectOutputStream` (ambos definidos em `java.io`);
  - Exceções devem ser "tratadas" dentro do método (imprima uma mensagem adequada);
- O método `desserializa` recebe `nomeArquivo` (`String`) e retorna um `Object`;
  - O método deve criar, instanciar e manipular um `FileInputStream` e um `ObjectInputStream` (ambos definidos em `java.io`);
  - Exceções devem ser "tratadas" dentro do método (imprima uma mensagem adequada);

Exercício #1 Parte #2

- A classe **`Aplicacao`** tem o método ***`main`***;
- O método ***`main`*** deve:
  - Criar um **`ArrayList`** de **`String`**s (**`frases`**) e usar o método **`desserializa`** para ler textos do "banco de frases" em "**bancodefrases.poo**";
  - Criar um **`ArrayList`** de **`Jogador`**es (**`jogadores`**) e usar o método **`desserializa`** para ler os (estados dos) três melhores jogadores em "**top3ranking.poo**";
  - Pedir ao usuário que informe seu **`nome`**, e instanciar **`jogador`** (**`Jogador`**) com esse valor;
  - Selecionar aleatoriamente uma **`String`** de **`frases`** (**`frase`**) e pedir que o usuário a digite o mais rapidamente possível, armazenando-a em **`jogada`** (**`String`**);
  - [...]

Exercício #1 Parte #2

- [...]
- Computar o tempo em segundos desde que `jogador` pode começar a digitar até ele apertar o ***Enter***;
- Verificar se `frase` coincide com `jogada`:
  - Se sim, atribuir a razão entre o número de caracteres digitados pelo tempo (em segundos) a `score` de `jogador`;
  - Se não, atribuir zero a `score` de `jogador`; e informar na tela que ele errou a digitação;
- Mostrar o ranking atualizado com os três melhores colocados (usar `System.out.println`)!
- Atualizar os (estados dos) três melhores `jogadores` em "**top3ranking.poo**" utilizando o método `serializa`.

Exercício #1 Parte #2

- [...]
- Computar o tempo em segundos desde que `jogador` pode começar a digitar até ele apertar o ***Enter***;
- [...]
  - Se não, atribuir zero a `score` de `jogador`, e informar na tela que ele errou a digitação;
- Mostrar o ranking atualizado com os três melhores colocados (usar `System.out.println`)!
- Atualizar os (estados dos) três melhores `jogadores` em "**top3ranking.poo**" utilizando o método `serializa`.

`ArrayList<T>` é um objeto serializável!
Em nossa implementação, é **necessário** que o considere nas chamadas de `serializa` e `desserializa` !

Exercício #1 Parte #3

- Desafios (obrigatórios):
  - obter **`nome`** e **`jogada`** da entrada padrão (pode-se usar um `java.util.Scanner`);
  - sortear **`frase`** de **`frases`** (pode-se usar `java.util.Random` como no slide 9 da Aula 19);
  - obter o intervalo de tempo (em segundos) que o **`jogador`** teve para digitar **`jogada`**;
    - pode-se usar `System.nanoTime()` - que retorna o tempo de máquina em nanossegundos;
  - fazer os *castings* necessários para "`ArrayList` de ..." nas chamadas de **`desserializa`**;
  - alterar o *ranking* dos três melhores **`jogadores`**
    - pode-se adicionar **`jogador`** a **`jogadores`**, ordenar **`jogadores`** e **`remover`** o último colocado - ver o método `compare` da interface `java.util.Comparator`;

Exercício #1 Parte #3

Se acatares a sugestão de implementar um **`Comparator`**...

```
import java.util.Comparator;

public class Ordenador implements Comparator<Jogador>
{
    public int compare(Jogador jogador1, Jogador jogador2) {
        return Double.compare(jogador2.getScore(), jogador1.getScore()); //Double.compare compara doubles
    }
}
```

```
        /* Em algum lugar no método main... */

        /* Mostra ranking */
        jogadores.add(jogador);
        jogadores.sort(new Ordenador());
        jogadores.remove(3); // Há quatro instâncias Jogador em jogadores

        /* ... */
    }
}
```

Exercício #1 - Exemplo de Execução #1

- A primeira execução de **`Aplicacao`**, conforme descrita, deve* imprimir na tela:

```
Digite seu nome:
Fulano
Digite: 'Ser ou não ser, eis a questão.'
Ser ou não ser, eis a questão.
Sua taxa de digitação foi de: 6.25 c/s!
Ranking:
Fulano (6.25 c/s)
Anônimo (0.0 c/s)
Anônimo (0.0 c/s)
```

* o **`nome`**, **`frase`** e tempos de digitação podem ser outros.

Exercício #1 - Exemplo de Execução #2

- A primeira execução de **`Aplicacao`**, conforme descrita, deve* imprimir na tela:

```
Digite seu nome:
Beltrano
Digite: 'Ser ou não ser, eis a questão.'
Ser ou não ser, eis a questão!
Você errou a digitação de 'Ser ou não ser, eis a
questão.'!
Ranking:
Anônimo (0.0 c/s)
Anônimo (0.0 c/s)
Anônimo (0.0 c/s)
```

* o **`nome`**, **`frase`** e tempos de digitação podem ser outros.

Atividades

- Entrega de um arquivo **.zip** (contendo a estrutura de diretórios da aplicação e os arquivos .java)
  - Entregue a implementação descrita no Exercício #1 Partes #2 e #3
  - Entrega até às 23:55h de 22/04/2021

Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada

Programação Orientada a Objeto

24. Manipulação de Arquivos (Prática)

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

2° Semestre de 2020