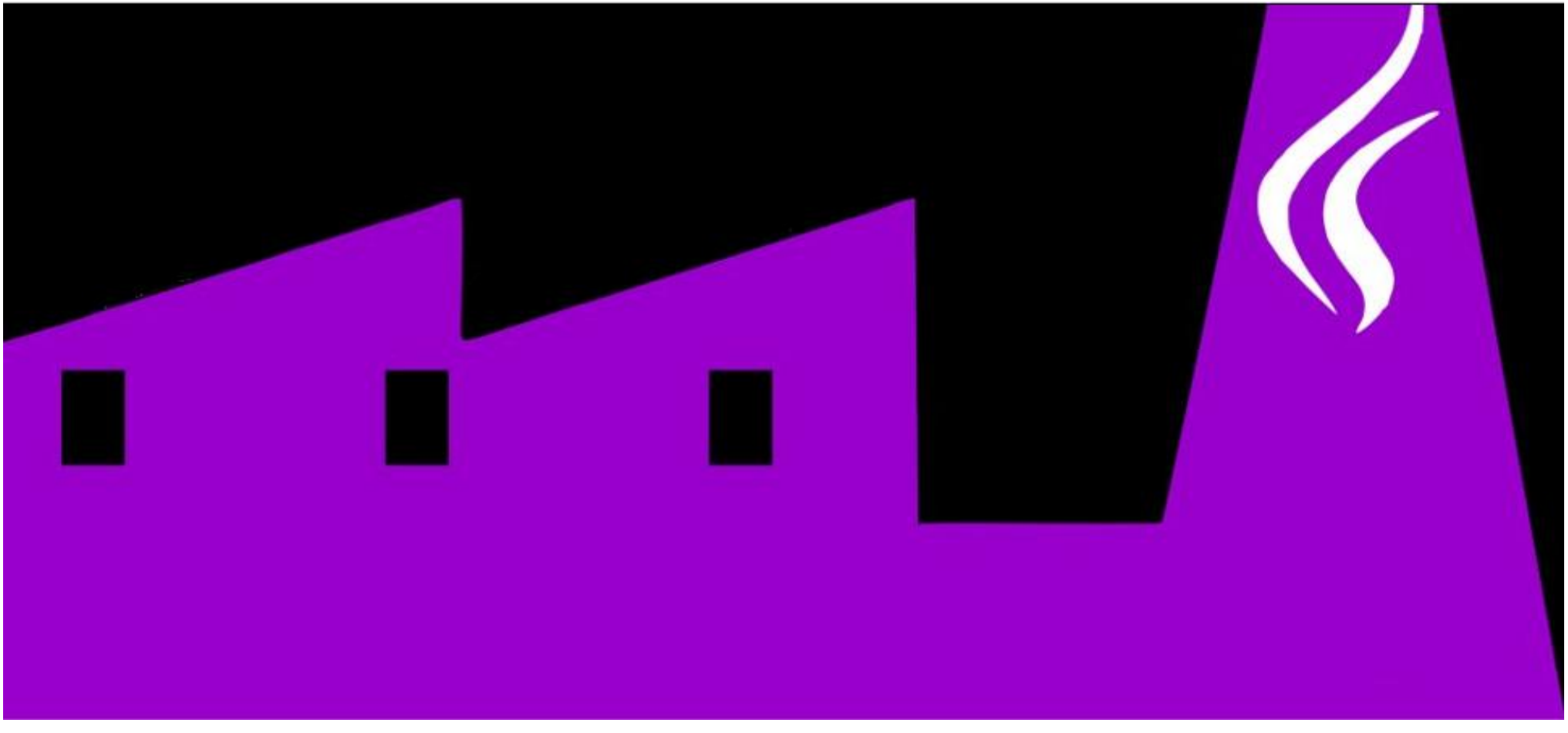
Fábrica de Software

Prof[es.] Ivan L. Süptitz e Evandro Franzen

Arquivos em Java

Tipos Enumerados



Ob et vos da aula

- Saber como manipular **arquivos** em Java:
  - não vamos detalhar pois é semelhante a manipulação vista em Python;
- Aprender a utilizar **enumeradores**;
- Preparação para a prova 2 por meio de **exercícios práticos**

IO. em Java

Ass m como todo o resto das b bl otecas em Java, a parte de controle de entrada e sa da de dados (conhec do como o) e or entada a ob etos e usa os pr nc pa s conce tos mostrados estudados ate agora: nterfaces, classes abstratas e pol morf smo.

A de a atras do pol morf smo no pacote ava. o e de ut l zar fluxos de entrada (InputStream) e de sa da (OutputStream) para toda e qualquer operacao, se a ela relat va a um arqu vo, a uma conexao remota v a sockets, ou ate mesmo as entrada e sa da padrao de um programa (normalmente o teclado e o console).

As classes abstratas InputStream e OutputStream def nem, respect vamente, o comportamento padrao dos fluxos em Java: em um fluxo de entrada, e poss vel **ler bytes** e, no fluxo de sa da, **escrever bytes**.

InputStream, InputStreamReader e BufferedReader

Para recuperar um caractere, prec samos traduz r os bytes para o respect vo cod go un code, sso pode usar um ou ma s bytes. Escrever esse decod f cador e mu to compl cado, quem faz sso por voce e a classe InputStreamReader.

```
class TestaEntrada {
    public static void main(String[] args) throws IOException {
        InputStream is = new FileInputStream("arquivo.txt");
        InputStreamReader isr = new InputStreamReader(is);
        int c = isr.read();
    }
}
```

InputStream, InputStreamReader e BufferedReader

Apesar da classe abstrata Reader a a udar no trabalho de man pulacao de caracteres, a nda ser a d f c l pegar uma Str ng. A classe BufferedReader e um Reader que recebe outro Reader pelo construtor e concatena os d versos chars para formar uma Str ng atraves do metodo readL ne:

```
class TestaEntrada {
    public static void main(String[] args) throws IOException {
        InputStream is = new FileInputStream("arquivo.txt");
        InputStreamReader isr = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(isr);
        String s = br.readLine();
    }
}
```

InputStream, InputStreamReader e BufferedReader

O metodo readL ne devolve a l nha que fo l da e muda o cursor para a prox ma l nha. Caso ele chegue ao f m do Reader (no nosso caso, f m do arqu vo), ele va devolver null. Entao, com um s mples laco, podemos ler o arqu vo por nte ro:

```
class TestaEntrada {
    public static void main(String[] args) throws IOException {
        InputStream is = new FileInputStream("arquivo.txt");
        InputStreamReader isr = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(isr);

        String s = br.readLine(); // primeira linha

        while (s != null) {
            System.out.println(s);
            s = br.readLine();
        }

        br.close();
    }
}
```

OutputStream

Escrever em um arquivo é o mesmo processo, porém de forma inversa e utilizando a classe OutputStream

```
class TestaSaida {
    public static void main(String[] args) throws IOException {
        OutputStream os = new FileOutputStream("saida.txt");
        OutputStreamWriter osw = new OutputStreamWriter(os);
        BufferedWriter bw = new BufferedWriter(osw);

        bw.write("lepo lepo");

        bw.close();
    }
}
```

O método write do BufferedWriter não insere o(s) caractere(s) de quebra de linha. Para isso, você pode chamar o método newLine.

Mais fácil

Existem duas classes chamadas java.io.FileReader e java.io.FileWriter. Elas são atalhos para a leitura e escrita de arquivos.

Escrevendo:

```
try {
    FileWriter arq = new FileWriter("saida.txt");
    PrintWriter gravar = new PrintWriter(arq);

    gravar.printf("123 testando");

    arq.close();
} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

Mais fácil

Lendo:

```
try {
    FileReader arq = new FileReader("saida.txt");
    BufferedReader lerArq = new BufferedReader(arq);

    String linha = lerArq.readLine(); // lê a primeira linha
    // a variável "linha" recebe o valor "null" quando o processo
    // de repetição atingir o final do arquivo texto
    while (linha != null) {
        System.out.printf("%s\n", linha);
        linha = lerArq.readLine(); // lê da segunda até a última linha
    }

    arq.close();
} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

Constantes

A pr nc pal d ferenca entre constantes e var ave s e que uma constante nao pode ter o seu valor atr bu do ma s de uma vez, po s ela recebe um valor f nal mutavel. Ja a var avel (atr buto) pode ser alterada d versas vezes durante a execucao do programa.

Exemplo:
Tanto um atr buto como uma var avel nterna a um metodo pode ser declarado como constante:

```
public class ExemploConstante {
    final int VALOR_MAX = 100;
}

public class Constante {

    public static void main(String[] args) {
        final Double PI = 3.14159265;
    }
```

Enumeradores ou tipo enum

O uso de constantes em Java e tao mportante que a l nguagem possu uma ferramenta espec al para manusear com fac l dade suas constantes: enum

De uma mane ra s mpl f cada, enum e uma classe ava espec al para tratar constantes.

A func onal dade pr nc pal de enum e agrupar valores com o mesmo sent do dentro de uma un ca estrutura, como por exemplo meses, d as da semana, cores, tabela per od ca, etc.

Enumeradores ou tipo enum

Sao t pos de campos que cons stem em um con unto f xo de constantes (stat c f nal), sendo como uma l sta de valores pre-def n dos.

Na l nguagem de programacao Java, pode ser def n do um t po de enumeracao usando a palavra chave enum.

Todos os t pos enums mpl c tamente estendem a classe ava.lang.Enum, sendo que o Java nao suporta heranca mult pla, nao podendo estender nenhuma outra classe.

Características

- As instâncias dos tipos enum são criadas e nomeadas junto com a declaração da classe, sendo fixas e imutáveis (o valor é fixo);

- Não é permitido criar novas instâncias com a palavra chave new;

- O construtor (se quiser usar) é sempre declarado private;

- Seguindo a convenção, por serem objetos constantes e imutáveis (static final), os nomes declarados recebem todas as letras em MAIÚSCULAS;

- As instâncias dos tipos enum devem obrigatoriamente ter apenas um nome;

- Opcionalmente, a declaração da classe pode incluir variáveis de instância, construtor, métodos de instância, de classe, etc.

Declaração Enum

Na declaração é definida uma classe chamada de tipo enum.

O corpo da classe enum pode incluir métodos e outros campos.

O compilador automaticamente adiciona alguns métodos especiais quando se cria um enum.

Dica: Declaração enum (sempre definir como letras maiúsculas).

```
public enum Semaforo {
    VERMELHO, AMARELO, VERDE;
}
```

Enumeração

```
public static void main(String[] args) {

    acaoCondutor(Semaforo.VERDE);
}

private static void acaoCondutor(Semaforo x) {
    if (null != x) {
        switch (x) {
            case VERDE:
                System.out.println("acelera");
                break;
            case AMARELO:
                System.out.println("desacelera");
                break;
            default:
                System.out.println("pára");
                break;
        }
    }
}
```

Enumeração

Como a estrutura enum é derivada da classe Enum podemos ter ainda em sua declaração outros métodos e propriedades, tornando as enumerações ainda mais úteis.

```
public enum Semaforo {

    VERMELHO, AMARELO, VERDE;

    public String getAcao() {
        switch (this) {
            case VERDE:
                return "acelera";
            case AMARELO:
                return "desacelera";
            default:
                return "pára";
        }
    }
}
```

```
public static void main(String[] args) {

    System.out.println(Semaforo.VERDE.getAcao());
}
```

Enumeração

```
public enum Dias {
    DOMINGO(1), SEGUNDA(2), TERCA(3), QUARTA(4), QUINTA(5), SEXTA(6), SABADO(7);
    int valorDia;

    Dias(int dia) {
        valorDia = dia;
    }
}
```

```
public class JavaEnum {

    public static void main(String[] args) {

        System.out.println(Dias.DOMINGO.valorDia);
    }

}
```

Conversões

```
public class App {

    public enum FaseLua {
        NOVA, CRESCENTE, CHEIA, MINGUANTE;
    }

    public static void main(String[] args) {
        FaseLua a = FaseLua.valueOf("CHEIA"); //convertendo de String para Enum
        String sA = a.toString();//convertendo de Enum para String

        FaseLua b = FaseLua.values()[3];//convertendo de Inteiro para Enum
        int iB = b.ordinal();//convertendo de Enum para Inteiro
    }
}
```

Exercício 10.1

Criar um Enumerador para os tipos de escala de temperatura possíveis: Célsius, Fahrenheit e Kelvin. Crie uma classe chamada ConversorTemperatura que recebe no construtor um float com a temperatura em graus Célcius. A classe deve ter 2 métodos em sobrecarga como a seguir:

- float converte(Escala escala);
- float converte(String escala);

Ambos retornam a temperatura que foi passada pelo construtor convertida para a escala que o usuário informou (passada no método). As fórmulas de conversão são:

K = C + 273
F = 1,8 * C + 32