**Comparação de Strings**

1 - Valores constantes do tipo String têm a mesma referência.
"XPTO" = = "XPTO" ➔ expressão com valor true

2 – Strings construídas em tempo de compilação são tratadas como valores constantes do tipo String.

String s1 = "XPTO";
String s2 = "XPTO"

s1 = = s2 ➔ expressão com valor true

3 – Strings construídas em tempo de execução têm referências distintas:

```
String s1 = new String( “XPTO”);
String s2 = new String (“XPTO”);
```

s1 = = s2 ➔ false

mas

s1.equals(s2) ➔ true

Ex.lo 1

```
public class Exemplo {

    static String s0;

    public static void setS0 (String s){

            s0 = s;

     }
```

```
// ainda na classe  exemplo.

   public static void main (String [] args) {
      String s = "XPTO";
      setS0 (s);

     System.out.println ( s + "    "  +  s0);

     System.out.println ( s = =  s0);

     s = "XX";            //é criada uma nova instância da String s;

     System.out.println ( s + "    "  +  s0)
   }
} // fim da classe exemplo
```

Output:

```
XPTO   XPTO
true
XX  XPTO
```

E se no método **setS0**

substituirmos

**s0 = s**

por **s0= new String (s);**

O que acontece?

s = = s0 ???

s.equals (s0) ????

Exercício: testar exemplo anterior

Qual o output do seguinte programa:

```
public static void main(String[] args) {
    String s1 = new String ("XPTO");
    String s2 = "XPTO";
    System.out.println ( s1 == s2);
    System.out.println ( s1.equals(s2) );

    String s3 = "XPTO";
    System.out.println ( s2 == s3);
    System.out.println ( s2.equals(s3) );
}
```

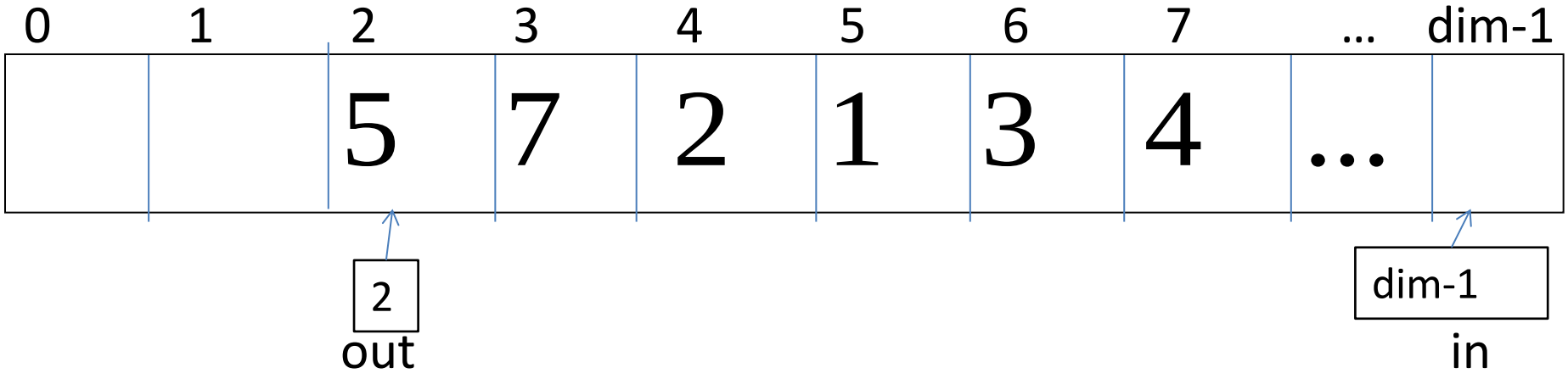
false
true
true
true

**Regra:**

- comparar Strings sempre com o método equals definido na classe String.

## Métodos parciais e pré-condições

Suponhamos uma classe que implementa o tipo abstracto de dados Fila (First In First Out) usando um array circular de valores inteiros.




out – posição do próximo elemento a sair
in – posição para onde entra o próximo elemento

out – posição do próximo elemento a sair

in – posição para onde entra o próximo elemento

Fila cheia: in = = out

Fila vazia: in == out

Para distinguir: *variável com o número de elementos da fila.*

n_elementos

```
public class Fila {

  // variáveis de instância

  private int [] fila ;
  private int n_elementos, dim, in, out;

  // construtor
  public Fila (int n){
    dim = n;
    fila = new int[dim];
    in = 0;
    out = 0;
  }
```

**// métodos de instância**

```
public void inserir ( int e ){
    fila [in] = e;
    n_elementos ++;
    in = (in + 1) % dim;
  }

  public int retirar (){
    int e = fila [out];
    n_elementos --;
    out = (out + 1) % dim;
    return e;
  }
}
```

Operações condicionadas

- nem sempre podem ser realizadas

**Como resolver?**

Hipótese 1:

- colocar um parâmetro de retorno que indica o sucesso da operação

```
public boolean inserir ( int e ){

  if ( n_elementos != dim ) {
            ...

     return true;
  } else {

     return false;
     }
}
```

Ao invocar o método:

*...main ....*

```
Fila filaA  = new Fila(10);

boolean sucesso = filaA.inserir(123);

if ( ! sucesso )
    System.out.println ( “Erro: Fila Cheia” );
```

O que fazer no caso da operação retirar? Já temos um resultado!
Podemos definir um novo tipo de objecto ...

Hipótese 2:

- implementar métodos que funcionam como pré–condições das operações condicionadas.

definir na classe Fila:

```
public boolean cheia () {
   return (n_elementos == dim);
 }

 public boolean vazia () {
   return ( n_elementos == 0);
 }
```

Ex.lo de utilização:

```
...main ....

Fila filaA = new Fila (10);

if (   ! filaA.cheia() ) {

       filaA.inserir(123);

}else {
       System.out.println ( “Erro: Fila Cheia” );
}
...
```

```
...

if (   ! filaA.vazia() ) {

      int i = filaA.retirar();

}else {
       System.out.println ( “Erro: Fila vazia” );
}

…
```

Construa a classe Pessoa. Uma pessoa tem um nome, um número de identificação fiscal, um conjunto de contactos (array de objetos do tipo telefone) – 3 no máximo: “fixo”, “móvel” e “emprego”).

- Defina um construtor que receba o nome como parâmetro;
- Defina os getters;
- Defina os setters;
- Defina o método toString;
- **Defina um método que consulte o número do telefone móvel da pessoa**

- Construa uma classe de teste, que crie duas pessoas e teste todos os métodos da classe Pessoa
- **Construa um método de classe (public static) que dado um array de pessoas devolva o número de pessoas que têm telemóvel.**

Exercícios (*1 ª Frequência 2017/18, parte prática*)

**1-** Uma Pergunta de um teste pode ser representada por um identificador único (atributo **numero**), um **texto** com o conteúdo da pergunta, e um **valor** entre 0 e 20 correspondente à cotação da pergunta. A listagem abaixo apresenta o cabeçalho da classe Pergunta e a declaração dos seus atributos. Pretende-se que o número das perguntas seja gerado de forma automática usando o atributo **ultimo** para armazenar o último número de pergunta gerado.

```
public class Pergunta {
   private static int ultimo =0;
   private int numero;
   private String texto;
   private double valor;
```

Exercícios (*1ª Frequência 2017/18, parte prática*)

**a)** Para a classe Pergunta defina o construtor sem parâmetros.

**b)** Para cada atributo da classe Pergunta defina os getters e os setters.

**c)** Para a classe Pergunta defina o método toString.

Exercícios (*1 ª Frequência 2017/18, parte prática*)

**2 –** Uma frequência pode ser representada pelo nome da respectiva **disciplina**, por uma lista de perguntas (atributo do tipo array de objectos do tipo Pergunta) e ainda um atributo com a **data** da frequência (atributo do tipo LocalDate (pode consultar a API da classe LocalDate no final do teste). Supondo a declaração abaixo,

```
import java.time.LocalDate;

public class Frequencia {
   private String disciplina;
   private LocalDate data;
   private Pergunta [] perguntas;
```

Exercícios (*1 ª Frequência 2017/18, parte prática*)

**a)** Defina um construtor que receba como parâmetros o nome da disciplina e o número de perguntas do teste. Deve inicializar o atributo data com o valor da data do sistema operativo.

**b)** Defina o get e o set do atributo perguntas. Para o resto do teste, supomos que os restantes getters e setters estão definidos.

**c)** Defina o método "to String" para a classe Frequencia.

**d)** Para a classe Frequencia defina um método, **totalCot**, que calcule a soma das cotações de todas as perguntas do teste.

## Classe LocalDate:

**Nota:** a classe LocalDate é imutável, não possui um construtor público. Para criar um objeto do tipo LocalDate pode usar entre outros os métodos **now** e **of** descritos abaixo.

public static LocalDate **now**()
- devolve a data actual do sistema operativo da sua máquina, no formato: aaaa-mm-dd;

public static Localdate **of** (int aaa, int mm, int dd)
permite criar um novo objeto do tipo LocalDate com o valor aaa-mm-dd;

**Métodos de instância:**

public String toString()
- devolve uma String com a data no formato aaaa-mm-dd. ...

**e)** Para o programa abaixo, desenhe **as variáveis que existem**, indique o seu **valor** e o **output** do programa:

```
public static void main(String[] args) {
        Pergunta p1, p2;
        p1 = new Pergunta();
        p1.setTexto("O que é uma variável de instância");
        p2 = p1;
        p2.setTexto("O que é um construtor");
        p2.setValor(6.0);
        Pergunta p3 = new Pergunta();
        p3.setTexto("O que é um modificador de acesso");
        p3.setValor(8.0);
        Pergunta [] pp = new Pergunta[3];
        pp[0] = p1;    pp[1] = p2;   pp[2] = p3;
        LocalDate dataFreq = LocalDate.of(2017, 12, 15);
```

**e)** Para o programa abaixo, desenhe **as variáveis que existem**, indique o seu **valor** e o **output** do programa:

```
public static void main(String[] args) {
    …
    Frequencia f = new Frequencia ("POO", 3);
    f.setPerguntas(pp);
    f.setData(dataFreq);
    System.out.println(f);
    System.out.println(f.totalCot());
  }
```

**Resolução:**

**1- a)**

```
public class Pergunta {
   private static int ultimo =0;
   private int numero;
   private String texto;
   private double valor;


public Pergunta (){
      ultimo++;
      numero = ultimo;
      texto = "";
   }
```

**Resolução:**

**1- b)**

**b)**

```
 public static int getUltimo() {
       return ultimo;
   }
   public static void setUltimo (int ultima) {
       Pergunta.ultimo = ultimo;
   }

   public int getNumero() {   return numero; }

   public void setNumero(int numero) { this.numero = numero;    }
```

**Resolução:**

**1- b) …**

```
   public String getTexto() {   return texto;  }

    public void setTexto(String texto) {  this.texto = texto;     }

   public double getValor() {  return valor;     }

   public void setValor(double valor) {    this.valor = valor;   }

    public String toString() {
      return "Pergunta{" + "numero=" + numero + ", texto=" + texto + ",
                                                          valor=" + valor + '}';
   }
 } // fim da classe
```

**Resolução:**

**2- a)**

```
public class Frequencia {
   private String uc;
   private LocalDate data;
   private Pergunta [] perguntas;

   public Frequencia (String uc, int dim){
      this.uc = uc;
      data = LocalDate.now();
      perguntas = new Pergunta[dim];
      for (int i = 0; i < perguntas.length; i++) {
          perguntas[i] = new Pergunta();
     }
   }
```

**Resolução:**

**2- b)**

```
 public String getUc() {     return uc;     }

public void setUc(String uc) {      this.uc = uc;      }

public LocalDate getData() {       return data;     }

public void setData(LocalDate data) {
                this.data = data;  // !! só copia o endereço endereço   }

public Pergunta[] getPerguntas() {      return perguntas;       }
```

**Resolução:**

**2- b)**

```
public void setPerguntas(Pergunta[] perguntas) {
    for (int i = 0; i < perguntas.length; i++) {
        this.perguntas[i].setNumero ( perguntas[i].getNumero() );
        this.perguntas[i].setTexto ( perguntas[i].getTexto() );
        this.perguntas[i].setValor ( perguntas[i].getValor() );
    }
}
```

**Resolução:**

**2- b)**

```
public void setPerguntas(Pergunta[] perguntas) {
    for (int i = 0; i < perguntas.length; i++) {
        this.perguntas[i].setNumero ( perguntas[i].getNumero() );
        this.perguntas[i].setTexto ( perguntas[i].getTexto() );
        this.perguntas[i].setValor ( perguntas[i].getValor() );
    }
}
```

**Resolução:**

**2- c)**

```
public String toString() {
    String s=  "Frequencia:" + "uc=" + uc + ", data=" + data + ",
           perguntas=" ;
           for (int i = 0; i < perguntas.length; i++) {
                                  s = s + ","  + perguntas[i];
           }
    return s;
 }
```

**Resolução:**

**2- d)**

```
public double totalCot (){
    double total = 0;

    for (int i = 0; i < perguntas.length; i++) {
     total= total +  perguntas[i].getValor();
    }

    return total;

}
```