

## Tópicos de hoje

- Tipos definidos pelo usuário
- Tipos estruturados de dados
  - Tipo array
  - Tipo String
- Classes como tipos de dados
  - Composição de tipos
  - Extensão de tipos
- Exercícios

## Tipos ordinais em Java

- Em Java não existe definição de tipos como em C++ e Pascal/Delphi:
  - C++: typedef
  - Pascal/Delphi: seção type
- Enumerações são definidas através de constantes em **interfaces**. Exemplo:
 

```
public interface Meses {..
  int janeiro=1,
    fevereiro=2, ....dezembro=12;
}
```

## Tipos estruturados

- São tipos compostos a partir de outros tipos de dados
  - **homogêneos**: todos os componentes pertencem ao mesmo tipo: Ex: array
  - **heterogêneos**: os componentes podem ser de tipos de dados diferentes. Ex. struct, union
- Operações sobre elementos
- Operações mais restritas sobre o conjunto

## Tipos estruturados usuais

Tipo	Componentes
▫ registro (record, struct, class)	▫ heterogêneo
▫ array	▫ homogêneo
▫ string	▫ homogêneo
▫ registro variante (union, record/case)	▫ heterogêneo
▫ set	▫ homogêneo

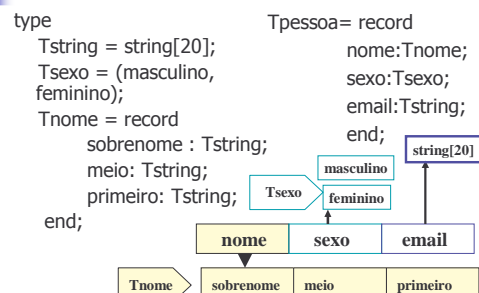
Os tipos estruturados podem ser fornecidos através de bibliotecas de tipos abstratos de dados

## Tipos definidos pelo usuário: compostos

- Associação de um sinônimo a um tipo estruturado
  - Exemplo em C/C++:
 

```
typedef struct {.....} registro;
```
- Úteis em estruturas de dados mais complexas
- **Composição de tipos**: ocorre quando um componente de um tipo estruturado é também um tipo estruturado, como por exemplo, arrays de registros ou arrays de arrays.

## Exemplo de composição



## Tipos estruturados em Java

- Homôgeneos: array e String
  - Array: elementos de mesmo tipo (primitivos, objetos)
  - String: sequência de caracteres Unicode entre ""
- Heterogêneos: registro
  - não são oferecidos através de uma 'declaração' de tipo
  - são definidos pelo usuário através de classes
  - estrutura de classes: campos de uma classe

## Arrays e Strings: declaração e instanciação

- Array

```
int numeros[ ]; int[ ] vet1, vet2, vet3;
numeros = new int[5];
```
- String

```
String s;
String s = new String ("gremio");
```
- Arrays de Strings

```
String[ ] str = new String[3];
```

## Arrays em Java

- Arrays são objetos
  - criados na memória dinâmica (*heap*) - referências
  - embora tenham tamanho fixo, pode-se alterar sua referência para um array de outro tamanho
- Declaração de arrays

```
int numero[ ]; int[ ] v1,v2; char mc[ ][ ];
```
- Depois é preciso criar (instanciar) o array, usando o operador *new*:

```
numbers = new int[5];
str = new String[3];
```

## Arrays em Java

- Declaração, instanciação e inicialização

```
int n[ ] = { 10, 20, 30, 40 };

String frase = "Uma string em Java!";
```

## Arrays com vários subscritos

- Um array bidimensional
  - é um array de arrays: cada elemento aponta para um array
  - os arrays que formam a segunda dimensão podem ter tamanhos diferentes
  - Exemplo:

```
int b[ ][ ];
b = new int [2][ ]; // instanciação de linhas
b[0] = new int [5]; // colunas da linha 0
b[1] = new int [3]; // colunas da linha 1
```

## Arrays como parâmetros

- Cada array 'conhece' seu tamanho
  - Exemplo: `int a[ ] = {1,2,3}; tam = a.length;`
- Arrays inteiros são passados por referência

```
int temperaturas[ ] = new int[31];
assinatura do método: void calcTemp(int temp[])
chamada do método: calcTemp(temperaturas);
```
- Passagem de parâmetros:
  - tipos primitivos: por valor
  - tipos compostos (objetos): por referência

## Arrays como valor de retorno

- Um método pode devolver um array como resultado. Exemplo:  

```
class Testarray{ .....
    static int [ ] devolveArray (int q) {
        return new int[q]; }
    .....
    public static void main (String args[]){
        int vetor 20[] = new int[];    // tamanho indefinido
        vetor20=devolveArray(20);    // demais comandos main
    }}
```

tipo de resultado

## Tipo String

- É um objeto da classe String (java.lang)
  - literais:** Strings anônimos
    - Exemplo: "azul"
  - variáveis:** Referências a Strings
    - Exemplo: String cor = " azul";
  - Lembrete:** cada classe define um tipo de dado (representação do tipo e operações sobre o tipo)
- Exemplos de declaração, instanciação e inicialização
  - String s1;      s1 = new String();
  - String s2 = new String("Guga");

## Strings x Arrays: exemplos

- Uso com parâmetro
  - public static void main ( String args[ ]) { }
- Array de caracteres
  - char charArray[ ] = { 'G', 'u', 'g', 'a' };  
notar tipo char
- Criando um String a partir de um array
  - s3 = new String (charArray);

## Strings x Arrays: exemplos

- Concatenação de Strings
  - saida = s3 + "venceu por " + n + "sets a" + m;  
converte int para String
- Sugestão: estudar a classe String e ver seus recursos de conversão de dados

## Classes como tipos de dados

- Uma definição de classe cria um novo **tipo de dado**
- Membros de uma classe: variáveis (atributos) e métodos (funções)
- Formato simplificado de definição de classe

```
class NomeDaClasse
{ // inicia corpo da definição
  // definição dos campos
  // definição das funções (opcionais)
} // fim da classe NomeDaClasse
```

## Classes x Registros

- Uma classe pode conter apenas definição de campos
- Os campos de uma classe formam um registro. Exemplo:

```
class Ator {
    String nome;
    int idade;
    char sexo;
}
```

Ator      nome | idade | sexo

## Classes x instâncias

- Classes**
  - definem o **tipo** de um objeto
  - não reservam **memória** para instâncias
- Instâncias = objetos**
  - Cada **objeto**, quando instanciado, ocupa espaço de memória para suas variáveis (de instância)
  - Cada objeto possui suas variáveis e seus valores (**estado**)

## Classes e objetos

- Classes**
  - são descritores (de objetos)
  - mesma classe pode se usada para instanciar inúmeros objetos
- Objetos**
  - são instâncias (de classe)
  - objetos de mesma classe: mesmo **tipo**
  - idênticas propriedades, diferentes dados
  - podem ser atribuídos e passados como parâmetros (referências!)

## Composição de tipos com classes

```
class Pessoa{
    String nome;
    int idade; }

class Novela{
    Pessoa autor;
    String titulo; }
```

## Extensão de tipos: herança

```
class Pessoa {
    String nome;
    int dia, mês, ano;
    // demais membros }

class Ator extends Pessoa {
    String contrato ;
    // demais membros }

class Aluno extends Pessoa {
    String num_matric, curso;
    // demais membros }
```

## Exemplo de classes e objetos


```
class Ator {
    String nome;
    int idade;
    char sexo;

    public static void main (String args[]) {
        Ator ana = new Ator(); // cria instância
        ana.nome= "Ana Paula";
        ana.idade= 22;
        ana.sexo= 'f';
    } // Ator
```

## Objetos: parâmetro e resultado

```
class Fruta {
    int gramas;
    int calorias;

    Fruta somaFruta(Fruta par){
        Fruta temp=new Fruta();
        temp.gramas = par.gramas +10;
        temp.calorias = par.calorias + 100;
        return (temp);
    }
}
```



## Exercícios em laboratório

- Exercitar:
  - arrays
  - Strings
  - Novos tipos de dados
  - Composição
  - Extensão

