

```

1 Scanner scanner = new Scanner (System.in);
2 try {
3     while (scanner.hasNext()) {
4         int num = scanner.nextInt ();
5         if (num < 1) {
6             throw new Exception ("conta"); Contar o salário com a menor
7         }
8         System.out.println (num + " fixe.");
9     }
10 } catch (InputMismatchException e) { Não é um número
11     System.out.println ("não pesca.");
12 } catch (Exception e) {
13     System.out.println ("não " + e.getMessage ());
14 } finally {
15     System.out.println ("pois.");
16 }

```

(a) (4v) Diga, pelas suas palavras, em 5 ou menos linhas, o que o programa faz, e em que condições termina.

(b) Diga qual o output para os seguintes inputs:

i. (2v) 10 20 30

ii. (2v) 2 1 0

iii. (2v) 123 xpto123 -10

*(erro para true)*

→ `Scanner.hasNext()` enquanto existir input produzido pela classe `Scanner`

→ `Scanner.nextInt();` pretendendo um input inteiro  $\geq 1$  para imprimir output (`num + "fixe!"`)

→ se o input introduzido for  $< 1$  é gerada uma exceção que é apanhada no `catch (Exception e)` gera output "não conta" *↳ finally "Pois."*

- se o input introduzido for uma letra carata string ou outro diferente de inteiro é criada uma exceção que é apanhada por `catch (InputMismatchException e)` output "não fixa" *↳ finally "Pois."*

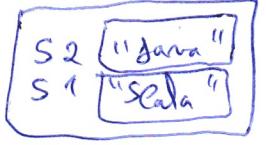
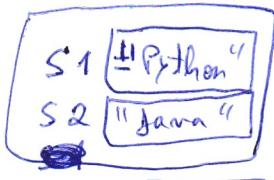
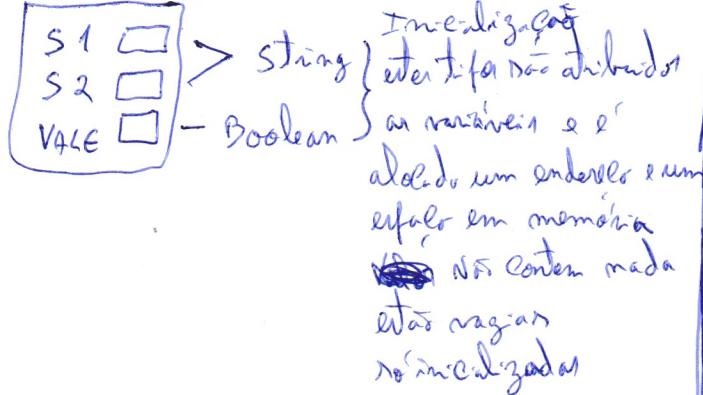
①

	Input	Output
①	10, 20, 30	10 fixe 20 fixe 30 fixe
②	2, 1, 0	2 fixe 1 fixe "não conta" "Pois."
③	123, xpto123, -10	123 fixe "não fixa" "Pois." -10 não é! Todavia foi intencionalmente com xpto123

2. (2v) Considerando o seguinte troço de programa, faça o diagrama de estado de memória correspondente:

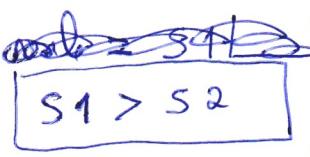
```

1 String s1, s2;
2 Boolean vale;
3 s1 = "Python";
4 s2 = "Java";
5 s1 = "Scala";
6 vale = s1 > s2;
    
```



Elocado espaço em memória para conter duas strings.

Aqui a string S1 é modificada para "SCALA".



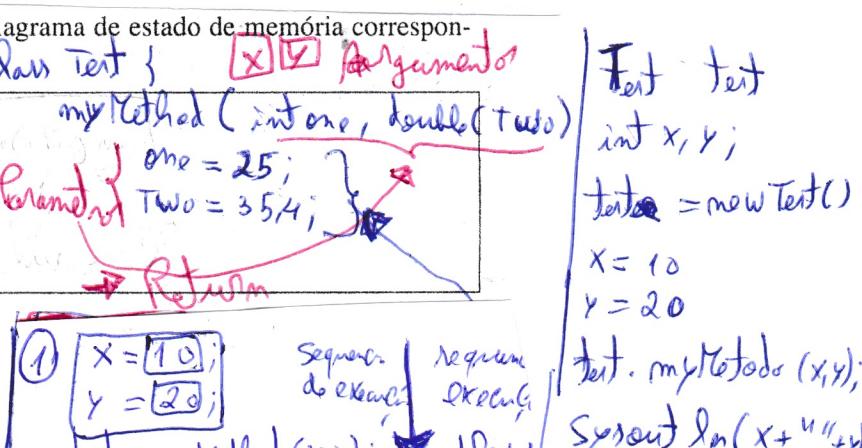
Elocado espaço em memória para Vale e os valores são copiados.

0	1	2	3	4
S	C	A	L	A

0	1	2	3
J	A	V	A

Vale  True

O valor True é guardado na variável Vale.



Nenhuma locais são executados dentro do método.

Locais dentro do método são executados.

3. As unidades curriculares da Universidade de Évora são identificadas por um código de exatamente 3 letras maiúsculas seguido de um ou mais algarismos. Por exemplo, P2 é INF0881.

(a) (2v) Defina uma expressão regular que descreva genericamente os códigos de UC.

(b) (2v) Restringindo o código de departamento ao conjunto dado pelos prefixos de 3 letras dos nomes de departamento (Informática, Gestão, Matemática, Física, Química e Pedagogia), considerando também que os números são especificados com exatamente 4 algarismos, apresente uma nova expressão regular que traduza essas limitações.

Pattern de chamada de expressões regulares as regras são:

[ ] representam sequências

\* zero ou mais ocorrências

{} n vezes

- alcance

( ) alcance de sequências para multiplas combinações

+ uma ou mais ocorrências

"([A-Z]{3}[0-9]{4})"

(c) (4v) Escreva um programa que leia palavras do input, recorrendo ao método next(), copiando-as para o output (com o método print()).

Sempre que ocorrer um padrão que possa ser o código duma UC (para determinar isso pode usar o método matches()), o programa deverá substitui-lo pelo texto: uma UC do Departamento de XXX, em que XXX é o nome do departamento.

Por exemplo, se tivermos o input "Inscrito a GES123 mas contente", o resultado será o output "Inscrito a uma UC do Departamento de Gestão mas contente". Pode usar métodos da classe String, como por exemplo substring(int beginIndex, int endIndex).

```
1 import java.util.*;
2 import java.util.regex.*;
3 class UCMain {
4     public static void main (String[] args) {
5         UC exe = new UC();
6         System.out.println(exe.encontraUC());  }}
7
8 class UC {
9     String word ;Matcher ma; String frase ;
10    void UC(){
11        String word ; Matcher ma; String frase ;
12    }
13    public String encontraUC () {
14
15        Scanner s = new Scanner(System.in);
16        System.out.println("Introduza o codigo UC");
17        String input = s.nextLine();
18        Pattern pat = Pattern.compile("( [A-Z] [A-Z] [A-z] [0-9]+ )"); /
19        frase = " ";
20        Scanner i = new Scanner(input);
21        while(i.hasNext()){
22            word = i.next();
23            if (word.length()>3){
24                ma = pat.matcher(word.substring(0,7));
25                if(ma.matches()){
26                    if (word.substring(0,7).equals("INF0881"))
27                        frase += " uma UC do Departamento de Informatica ";
28                }
29                else{
30                    frase += " " + word + " ";
31                }
32            }
33            frase += " " + word + " ";
34        }
35    }
36    return (frase); }}
```

Byte [-128 → 127] int [-2147483648 → 2147483647] Integer Default = 0

Short [-32768 → 32767] Long [-9 × 10<sup>19</sup> → 9 × 10<sup>19</sup>]

Double { Range { Default = 0.0 } } -3.14 × 10<sup>38</sup> → 3.14 × 10<sup>38</sup>  
 Float { } -1.7 × 10<sup>308</sup> → 1.7 × 10<sup>308</sup>

DecimalFormat df = new DecimalFormat ("0.###"); System.out num = 123.45789345;  
 System.out (df.format (num)) = 123.458 //

### Scanner

nextByte() byte b = scanner.nextByte();  
 nextDouble() double d = scanner.nextDouble();  
 nextFloat() float f = scanner.nextFloat();  
 nextInt() int i = scanner.nextInt();  
 nextLong() long l = scanner.nextLong();  
 nextShort() short s = scanner.nextShort();  
 next() String str = scanner.next();

Math  
 sqrt(a) = Math.sqrt (Math.max(x,y)+12.4);  
 log(a) = log<sub>e</sub> a  
 floor(a) = numero ≤ a  
 max(a,b) = maior de a ou b  
 pow(a,b) = a<sup>b</sup>

$$\sqrt{a} = \sqrt{a}$$

$$\log(a) = \ln(a)$$

$$\text{floor}(a) = \lfloor a \rfloor$$

### Import Statements

#### Class Comments

class Name { }

#### Data Members

#### Constructor & Method

< modifiers > < dataType > < name >;  
 private String ownerName;

Method	Modifier	Return Type	Method Name	Parameters
public	void		setOwnerName (String Name)	
			ownerName = name;	#statements

Constructor

import java.util.Random;

class dado {

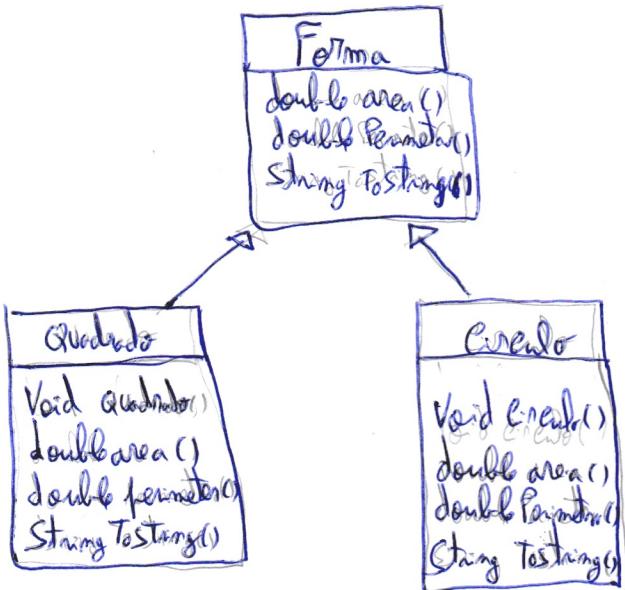
```
private static final int Max_number = 6
private static final int Min_number = 1
private static final int No_Number = 0
private int number;
private Random random;
public dado () {
    random = new Random ();
    number = No_Number;
}
//Random
public void roll () {
    number = random.nextInt (Max_number - Min_number + 1) + Min_number;
}
public int getNumber () {
    return Number;
}
```

```

1 abstract class Forma {
2     double area () { return 0; }
3     double perimetro () { return 0; }
4     String toString ();
5 };
6
7 class Quadrado extends Forma {
8     double lado;
9     void Quadrado (double lado) { this.lado = lado; }
10    double area () { return lado * lado; }
11    double perimetro () { return 4 * lado; }
12    String toString () { return "Quadrado_de_lado_" + lado; }
13 };
14
15 class Circulo extends Forma {
16     double raio;
17     void Circulo (double raio) { this.raio = raio; }
18     double area () { return Math.PI * raio * raio; }
19     double perimetro () { return 2 * Math.PI * raio; }
20     String toString () { return "Circulo_de_raio_" + raio; }
21 };

```

- (a) (1v) Explicite a relação entre as 3 classes mencionadas, i.e. herança, características das classes, etc. Para isso, desenhe um diagrama com estas classes.



- (b) (2v) Defina uma classe Retangulo com o posicionamento nesta hierarquia e comportamento esperados.

O construtor a definir deverá aceitar dois parâmetros do tipo `double`: um para a largura e outro para a altura. Inclua definições para os métodos `area()`, `perimetro()` e `toString()` apropriados para um retângulo.

```

class Retangulo extends Forma {
    double lado1;
    double lado2;
    void Retangulo (double lado1, double lado2) {
        this.lado1 = lado1;
        this.lado2 = lado2;
    }
    double area () { return lado1 * lado2; }
    double perimetro () { return 2 * (lado1 + lado2); }
    String toString () { return "lado1" + lado1 + "lado2" + lado2; }
}

```

- (c) (1v) Se quisesse definir o Quadrado em termos do Retangulo, o que é que faria?

*Basta trocar form Quadrado que herda de Retangulo que herda de Forma*

- (d) (1v) Considere que `f` é uma Coleção de formas, e que pretendemos saber qual delas a que tem a maior área. Para isso definimos um método `maiorForma()`. Este método poderá recorrer a todas mensagens que se podem enviar a objetos da classe `Forma`.  
 Sabe-se que a classe de `f` permite aceder a todas as formas que integram a coleção, recorrendo ao método `nextForma()`, que retorna cada uma das formas, em chamadas sucessivas.  
 Pretendemos que `maiorForma()` retorne a Forma do seu conjunto que tiver maior área, por exemplo, se tivermos:

```

1 Coleção f;
2 Forma maior;
3 f.acrescenta (new Quadrado (3.0)); // quadrado de lado 3
4 f.acrescenta (new Circulo (2.0)); // circulo de raio 2
5 f.acrescenta (new Retangulo (1.5, 2.5)); // retangulo de 1.5 x 2.5
6 maior = f.maiorForma ();
7 System.out.println ("A_maior_forma_é_ " + maior);
  
```

Diga qual será o output deste troço de código. Assuma que as declarações de classe estão todas feitas e que o método `toString()` foi definido para as subclasses de `Forma`.

$$\begin{aligned}
 \text{Área do quadrado} &= 3 \times 3 = 9 \\
 \text{Área do círculo} &= \pi \times 2 \times 2 = 12,566 \\
 \text{Área do retângulo} &= (2 \times 1.5) + (2 \times 2.5) = 8 \\
 \text{Logo o output é} &\quad \text{a área do círculo} \\
 \text{"A maior forma é"} &\quad \text{" + maior} \\
 &\quad \text{Círculo @) } \underline{12,566} //
 \end{aligned}$$

- (e) (2v) Complete a definição da função `maiorForma`.

```

1 class Coleção {
2   ...
3   Forma nextForma ();
4
5   Forma maiorForma () {
6     Forma aForma;
7     ...; // RESPONDA AQUI
8     return aForma;
9   }
10 }
  
```

Forma maiorForma()  
 Forma aForma;  
 $\text{maiorDe2} = \text{Math.max}(\text{Quadrado}(3.0), \text{Circulo}(2.0));$   
 $\text{maiorDe3} = \text{Math.max}(\text{maiorDe2}, \text{Retangulo}(1.5, 2.5));$   
 $\text{if} (\text{maiorDe3} == \text{Quadrado}(3.0))$   
      $\text{aForma} = \text{"Quadrado";}$   
      $\text{return aForma;}$   
 $\text{else if} (\text{maiorDe3} == \text{Circulo}(2.0))$   
      $\text{aForma} = \text{"Círculo";}$   
      $\text{return aForma;}$   
 $\text{else if} (\text{maiorDe3} == \text{Retangulo}(1.5, 2.5))$   
      $\text{aForma} = \text{"Retângulo";}$   
      $\text{return aForma;}$

```

(2) int f = 0;
if (f < 10)
    while (f < 10)
        System.out.println(f);
        f++;
}
}

```

```

Import java.util.*;
Import java.lang.Math;

public class Fermi {
    int a;
    int b;
    int c;
}

public Fermi() {
    a = getRandom();
    b = getRandom();
    c = getRandom();
}

int getRandom() { // classe private
    Random rand = new Random();
    int geranium = rand.nextInt();
    return geranium;
}

public int checkAll(int m, int n) {
    .....
    return 0;
}

```

```
public static void main (String [ ] args) {
    Scanner scan = new Scanner ( );
    Fermi fe = new Fermi ();
    System ("Input : ");
    int num1 = scan.nextInt ();
    System (fe. checkKA (num1, num2));
```

```

for (j = 0; j < 10; ++j)
    Extra.function(j);
}

```

```

// do while
int K = 0;
do {
    System.out.println(K);
    ++K;
} while (K < 10);

```

public class Thread {

```
public String matra;  
public String modelo;  
final double fatorger = 200;
```

public String toString() {

$$m_{\text{area}} = m_a;$$

public void ~~set~~ setMarea (String marea)  
    marea = marea;

```
public String getMarea() {  
    return marea;
```

inttate void Main  
main (String args) {

Amides  $\text{AV}_1 = \text{new Amides}$  (mg/mg)

Eighteen (av. - get them 11);

AV<sub>1</sub>. netMareca ("F<sub>1</sub> N")

→  $F_{1//}$

8

↳ substituting  
length  
indexof

Concatenation # string operation

String name;  
name = newString ("jon jana");

0 7 13

Index string

0	1	2	3	4	5	6	7
E	S	P	N	O	S	S	O

\* str. substring (i, j)

----- (6,7) = "so"

----- (0,8) = "Espreso"

(1,5) = "s/p/e"

(3,3) = //

(4,2) = error

Formula

#  $i \text{ to } j-1$

3  
4  
S  
6  
18

\* str. length () retorna o numero de caracteres na str

String str1, str2, str3, str4

str1 = "Hello" str2 = "jana" str3 = " " str4 = "a"

str4.length()  $\Rightarrow$  1

[str1.length()  $\Rightarrow$  5]

[str2.length()  $\Rightarrow$  4]

[str3.length()  $\Rightarrow$  0]

\* str. indexOf (subt)

retorna o indice onde comeca a palavra "Fazer, P2"

str.indexOf("Fazer")  $\Rightarrow$  0

str.indexOf("P2")  $\Rightarrow$  6

\* concatenation

(str1 + "Agora" + str2)

import java.util.\*;

class ch2Monogram {

public static void main (String [] args) {

String name, first, middle, last,  
spare, monogram;

spare = " ";

// Input the full name

Scanner scanner = new Scanner (System.in);

scanner.useDelimiter (System.getProperty ("line.separator"));

System.out.print ("Enter your full name (first, middle, last): ");

name = scanner.next();

// Extract first, middle, and last names

name Ivo M-gel Pago

spare = " ";

initials = name.substring (0, name.indexOf (spare))

12345.50

12.75

15

282,85

3884,46

Person [ ] person = new Person[20];

Person [0] = new Person(); // cria no indice 0 uma instância para person  
uma instância do objeto

// sóta é criado um array de Person do [0] a [20] e a unica [0] instancia nessa person  
uma instância de objeto da classe Person

x. charAt(0) # indica o inicio da string

① indica começo em zero mas faria comparação ainda começo em 1  
o indice zero só é ocupado

if { if (it == 0) - errado faria verificar se faria em base mecanismo  
utilizar um assert() ???

Verificar For (Person p: Person){ # faz utilização de coleções  
System.out.println(p.getName()); }

For-Each - diz reflexo sobre um Array

declarar Fazer um array com a tabuada <Type> [1] [5] < variável>

Criar < variável> = new <Type> [size 1] [5]      linha      coluna

Ler extrair objetos de uma lista

ArrayList  
LinkedList

interface List

interface Map

EDAI

Hash Map → não mantém a ordem

Tree Map → mantém a ordem

containsKey faz parte da interface Map

Capítulo 10

step 2 Capítulo 10

Search Result encontra o indice onde se encontra algo

public int linearSearch (int[] numbers, int searchValue)  
int loc = 0; while (loc < numbers.length) 88 mar ...

# o indice (-1) determina o final do array se tivermos x = numbers.length  
não retorna "not found"

pergunta linear - menor - nem menor

Codigr & pede: imprimir search procurar numero num intervalo de numeros  
num array e percorre todo o array ate encontrar Cap. 11. 15

```

double [ ] tempt = new double [ 366 ];
double hot = tempt [ 0 ];
for (int i = 1 ; i < tempt.length ; i ++ ) {
    if ( tempt [ i ] > hot );
        hot = tempt [ i ];
}

```

Permiter uma linha com palavras

```
class Bikeleta {  
    private String OwnerName; // variable  
    public void Bikeleta () { // constructor  
        OwnerName = "Unknown";  
    }  
}
```

Array

```
double [ ] x = new double [ 127 ]
```



Emersonia de AMay

$\text{Int}(\mathbb{Z}) = \{1, 2, 3, 4, \dots\}$ ; complement do anel

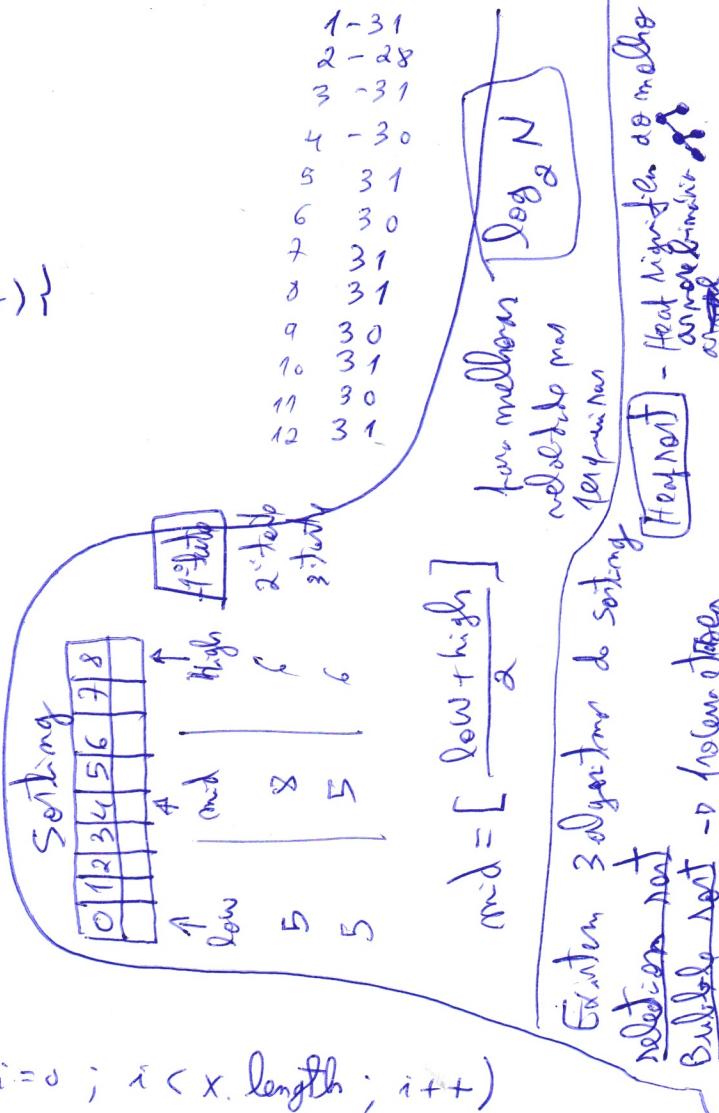
Double[] Y = { 62, 23, 3, ... }

String[1] mē = { "Jan", "Fene" }

~~ff~~ Tno Ca de aula

Aula 11/5 a[14h] Sd. 139

en vez de una dia 12/15



```
for(i=0 ; i < x.length ; i++)
```

Syment

$X[i] = \text{Planner.nextDouble}();$  (System.in) do  
Scan the input

`Sum += x[i]`

# initialize string [i]  
initially Num C<sub>i,j</sub>

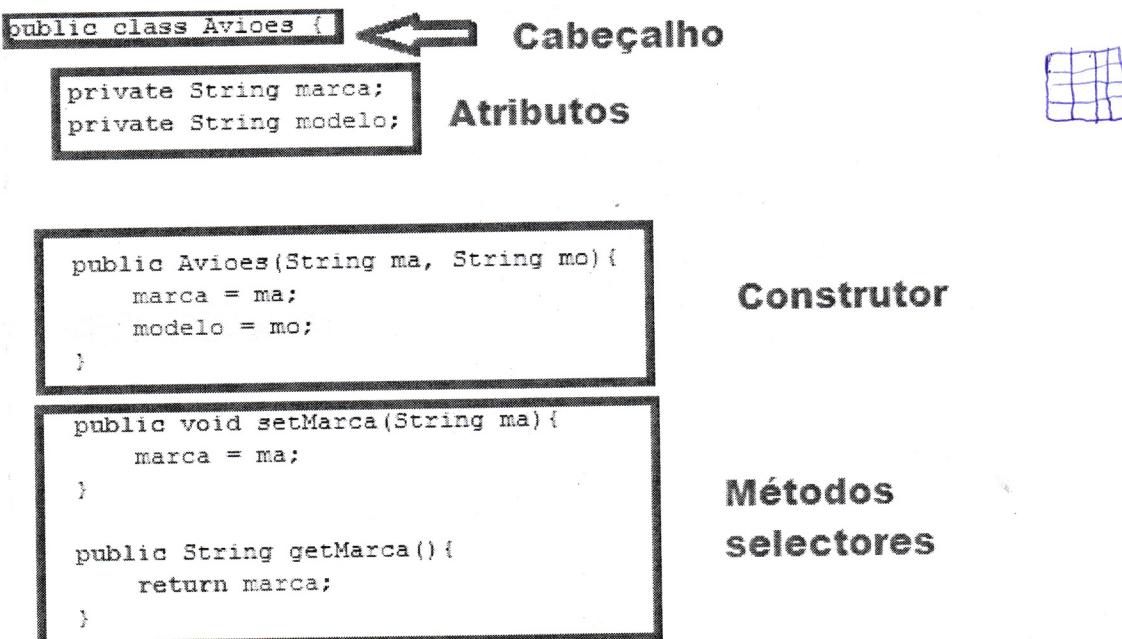
$\begin{matrix} x \\ y \end{matrix} \rightarrow$  a length  
Zoney

de Otamano do array

## - Programação Orientada a Objectos

Em Java, um Objecto é basicamente uma classe, possui atributos (variáveis de classe), métodos (conhecidos como funções em outra linguagens) e um Construtor onde a classe vai ser instanciada (instanciar significa criar um novo objecto).

Exemplo:



Por convenção, em java utilizam-se métodos selectores para alterar o valor a um atributo (set) ou descobrir o seu valor (get), as variáveis de classe (atributos) geralmente são declaradas como "private", e para aceder-lhes (noutras classes) deve-se utilizar os gets e sets.

Nota: Na própria classe, para fazer referência a um atributo utiliza-se `this.nome_do_atributo`.

A primeira parte da matéria é esta e não é difícil de compreender.

## - Arrays e Strings

É importante saber fazer a manipulação das Strings, os métodos mais importantes são o `length()`, `replace(char letraAntiga, char letraNova)`, `substring(int índice)`.

O `length` é fácil perceber (retorna o tamanho da String), o `replace` substitui todos os caracteres que sejam iguais a "letraAntiga" por "letraNova" e o `substring` devolve-te a o caractere no índice que especificares, é possível ainda no `substring` definires um "begin Index" e um "end Index".

Exemplos:

```
public void exemploStrings(){
    String palavra = "Exemplo";
    palavra.length(); // Retorna 7
    palavra.replace('x','a');// "Exemplo" passa a "Exempla", substitui o 'x', mete-se o 'a'
    palavra.substring(3); // Retorna 'm', uma String é um array de chars, logo existem indexes
    palavra.substring(0,4); // Retorna "Exemp"
}
```

Quanto aos arrays, os métodos mais importantes são o `arraycopy(arrayDePartida, indexDeInicioArrayDePartida, arrayDeChegada, indexDeInicioArrayDeChegada, indexDeFimArrayDeChegada)`, o `sort(array, indexInicio, indexFim)` e o `fill()`.

```
public void exemploArray() {
    int[] array1;//Declara o array
    array1 = new int[10];//Atribui 10 posições FIKAS ao array
    array1[0] = 1; //A primeira posição do array já tem 1
    array1[1] = 2; //A segunda posição já tem 2
    for (int i=2;i<array1.length;i++){//Preencher o array
        array1[i]=i+1;//Vai preencher os indexes do array com o índice seguinte
    }
    int array2[] = {0,0,0,0,0,0,0,0,0,0}; //Outra maneira de declarar arrays
    System.arraycopy(array1, 2, array2, 2, 7); //Cópia do array1 para array2 todo o que está
    //a partir do index 2 mas apenas entre 2 e 7 (segundo array), ou seja, o segundo array
    //já contém {0,0,3,4,5,6,7,8,0,0}
    Arrays.fill(array1, 0); //Zeraiza o primeiro array
}
```

### - Métodos que geralmente são pedidos para definir

Geralmente são pedidos os métodos `clone()`, `toString()` e `equals()`.

O `clone` faz-te uma cópia de um objecto para o outro e os 2 objectos NÃO SÃO IGUAIS mas possuem os mesmos valores.

O `toString` serve para definir uma forma de representar um Objecto e a partir do momento em que é declarado, quando fazes `print` a um Objecto ele utiliza esse método para esse efeito.

O `equals` verifica se a tua String é igual à dada por parâmetro, há que distinguir “`equals()`” de “`==`”, este último compara OBJECTOS, o “`equals()`” compara o VALOR de um Objecto.

Exemplo:

```
public Avioes clone(){
    return new Avioes(this.marca, this.modelo);
}

public String toString(){
    return "A marca do Avião é "+marca+ " e o modelo é "+modelo;
}

public boolean equals(Avioes av){
    if (this.marca == av.marca && this.modelo == av.modelo)
        return true;
    else return false;
}

public void testCloneToStringEquals(){
    Avioes av = new Avioes("Avenger", "F-10");
    Avioes av2 = av.clone(); //Copia para av2 o que está em av
    System.out.println(av); //Retorna o que está em toString()
    System.out.println(av.equals(av2)); //Retorna true
}
```

### - Sintaxe

Há algumas que devem ser relembradas:

public, private, protected – Os níveis de acesso em java.

public consegues aceder em qualquer classe, private só na própria classe e protected no mesmo package, os níveis de acesso referem-se a constructores, métodos, variáveis de classe e até às próprias classes. Nota: o protected não é muito usado em exames/frequências.

Os métodos são declarados como void caso não retornem nada ou então int, String, Object, etc., sempre que retornem alguma coisa adiciona-se “return”. Nota: Sempre que fazes um return, fazes um break; ao método e todo o código abaixo não irá ser executado.

Operadores de comparação:

== -> verifica se dois objectos/atributos são iguais

!= -> verifica se dois objectos/atributos são diferentes

&& -> “and” lógico

|| -> “or” lógico

>=, <=, >, < ->>> self-explanatory

## - Convenções

Em java por norma, os nomes das classes começam com letra Maiúscula, os nomes dos métodos escrevem-se desta forma “oMeuMetodo()” e os atributos escrevem-se “o\_meu\_atributo”.

Os métodos das classes são sempre declarados como private e são acedidos por outras classes através dos getAtributo() e setAtributo.

## - While e case

Exemplo:

```
public void exemploWhileCase() {
    boolean condicao = true;
    while(condicao){
        //Código
    }

    int x = 0;
    //O x enquanto muda de valor
    switch (x){
        case 0: break; //Faz alguma coisa
        case 1: break; //Faz alguma coisa
        default: //Caso não seja 0 nem 1, executa o que está depois dos ":".
    }
}
```

## - Estrutura ArrayList e Iterator

A estrutura ArrayList é basicamente um array, mas possui métodos próprios que são muito úteis e o seu tamanho não é fixo. Os métodos mais importantes são o add(), remove(), clear(), isEmpty(), indexOf(), size() e toArray().

O add, remove, clear, isEmpty e size penso que dá para perceber, o indexOf() retorna-te o índice do objecto que deres como parâmetro, se houverem 2 iguais ele refere-se ao primeiro; o toArray() converte de ArrayList para array.

Exemplo:

```
public void arrayListExemplo(){
    //Declaração do ArrayList, entre <> está o tipo de dados que a lista vai conter
    ArrayList<Avioes> lista = new ArrayList<Avioes>();

    Avioes av1 = new Avioes("Avenger", "Space");
    lista.add(av1); //Adiciona um novo objecto à lista
    lista.remove(0); //Remove o objecto no índice 0
    lista.remove(av1); //Remove o objecto específico (primeira ocorrência)
    lista.clear(); //Limpa a lista
    lista.isEmpty(); //Retorna true ou false, dependendo a lista esteja vazia ou não
    lista.indexOf(av1); //Retornaria 0(primeira posição) caso eu não
    //tivesse feito o remove/clear
    lista.size(); //Retorna 1 antes do remove/clear e 0 depois

    Avioes[] arrayAv = (Avioes[]) lista.toArray(); //Mete este ArrayList num
    //array normal, "(Avioes[])" corresponde a um "cast" para o objecto
    //"Avioes"
}
```

Os Iterators são classes específicas para percorrer uma ArrayList (ou outra estrutura que faça um implements Iterable), geralmente são criados com o método iterator() dos ArrayLists. Os métodos mais importantes são o hasNext() e o next(). O hasNext() é um boolean, caso haja um próximo elemento na lista ele retorna true e o next() avança e retorna o próximo elemento.

### Exemplo:

```
public void exemploIterator(){
    ArrayList<Avioes> lista = new ArrayList<Avioes>();
    lista.add(new Avioes("Avanger", "F-11")); //Inserido primeiro avião
    lista.add(new Avioes("Avenger", "F-12")); //Inserido segundo
    java.util.Iterator<Avioes> it = lista.iterator();
    it.hasNext(); //Retorna true
    it.next(); //Segue para a próxima posição e retorna o primeiro avião inserido
    it.hasNext(); //Retorna true
    it.next(); //Segue para a próxima posição e retorna o segundo avião inserido
    it.hasNext(); //Retorna false, fim do Array
    //Nota: o hasNext() pode ser utilizado dentro de um while para iterar a lista:
    while(it.hasNext()){
        it.next();
        //Faz outra cosa qualquer
    }
}
```

### - Exceptions

As exceptions em java servem para controlar inputs/valores indesejados nas variáveis/métodos.

Declaram-se assim(Nova classe):

```
public class MenuException extends Exception {

    //Exceção do menu
    public MenuException() {
        super();
    }
}
```

"Mandam-se" assim caso a condição se verifique:

```
if (x <= 0 || x >= 5) {
    throw new MenuException();
}
```

Capturam-se assim:

```
try {
    menu(); //Método onde fizeste "throw"
}
catch (MenuException e) {
    //Executa caso ele ocorra a exception
    System.out.println("Tua mensagem de erro");
}
```