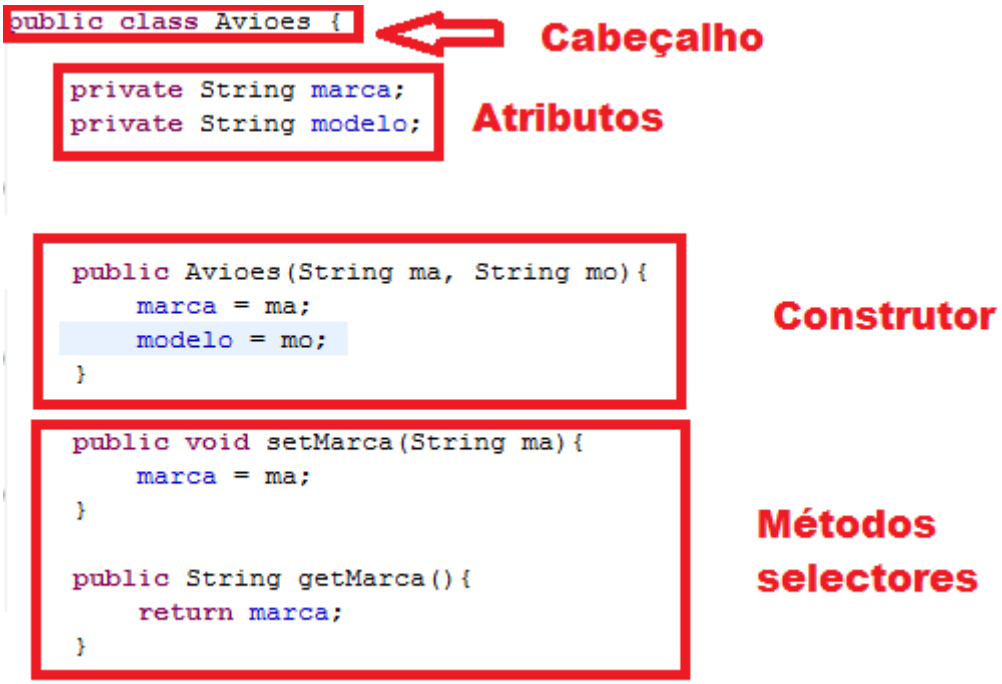
Programação II (Resumo)

- 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 índex).

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 índex 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 "Eaemplo", tira-se 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 FIXAS ao array
    array1[0] = 1; //A primeira posição do array irá ter 1
    array1[1] = 2; //A segunda posição irá ter 2
    for (int i=2;i<array1.length;i++){//Preencher o array
        array1[i]=i+1;//Vai preencher os indexes do array com o indice seguinte
    }
    int array2[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0};//Outra maneira de declarar arrays

    System.arraycopy(array1, 2, array2, 2, 7);//Copia do array1 para array2 tudo o que está
    //a partir do index 2 mas apenas entre 2 e 7 (segundo array), ou seja, o segundo array
    //irá conter {0,0,3,4,5,6,7,8,0,0}
    Arrays.fill(array1, 0);//Esvazia 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 á 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 ás 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 entretanto muda de valor
    switch (x){
        case 0: break;//Faz alguma cena
        case 1: break;//Faz alguma cena
        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 á 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, consoante 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 cena 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 {


    //Excepção do menu
    public MenuException() {
        super();

    }

}
```

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

```
if (x <= 0 || x >= 8) {
    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");
}
```