Criação de classes

- Para além de utilizar classes pré-definidas, num projecto é normal que seja necessário definir novas classes que ajudem a atingir o fim pretendido
- A sintaxe para definir uma classe é a seguinte:

```
class nome_da_classe {
    declarações de atributos
    construtores
    métodos
}
```

- As variáveis, construtores e métodos de uma classe são genericamente chamados membros dessa classe

Criação de classes

- Considere-se que se pretendia um programa que leia os dados de um conjunto de estudantes (nome e um conjunto de notas), calcule a sua média e ordene os estudantes por ordem decrescente das médias
- O primeiro passo será definir uma nova classe, a classe Estudante, para representar um aluno, que terá como atributos o nome, as notas e a média e como comportamentos a inicialização, o cálculo da média, o acesso à média e a escrita dos seus dados
- Terá que haver uma outra classe capaz de guardar o conjunto de alunos e ordená-los de acordo com a sua média

Criação de classes

```
class Estudante {
    //Atributos
    private String nome;
    private int [ ] notas;
    private float media;

    //Construtor da classe, promove a inicialização dos atributos
    public Estudante() {
    ...
    }
    //Comportamentos
    ...
}
```


Criação de classes

- Construtores
  - Cada classe tem pelo menos um construtor
  - É um tipo especial de método utilizado na criação de objectos dessa classe
  - Muitas vezes são usados para inicializar as variáveis (atributos)
  - Têm o mesmo nome da classe
  - Não têm valor de retorno, nem mesmo void
  - Não podem ser invocados como os restantes métodos
- Os construtores são invocados pelo operador new
- Assim, new String ("Olá"); implica a invocação a um construtor da classe String
- A criação de um novo objecto da classe Estudante pode ser conseguida por:

```
Estudante est = new Estudante();
```

Criação de classes

- O construtor da classe Estudante poderá ser:

```
public Estudante() {
    System.out.print("Nome do estudante: ");
    nome = User.readString();
    System.out.print("Quantas notas? ");
    int numNotas = User.readInt();
    //Cria a tabela notas com a dimensão necessária
    notas = new int [numNotas];
    for (int i = 0; i < numNotas; i++) {
        System.out.print("Nota "+(i+1)+" deste aluno: ");
        notas[i] = User.readInt();
    }
    media = calculaMedia();
}
```


Criação de classes

- Os comportamentos definidos podem ser implementados por:

```
//Escreve os dados de um estudante
public void imprimeEstudante() {
    System.out.print("As notas de "+nome+" são: ");
    for (int i = 0; i < notas.length; i++) {
            System.out.print(notas [i]+ " ");
    }
    System.out.println();
    System.out.println("A média é "+media);
}
```

Criação de classes

```
//Método de acesso externo à média
public float getMedia() {
    return media;
}
//Método para cálculo da média
private float calculaMedia() {
    float soma = 0;
    if (notas.length > 0){
        for (int i = 0; i < notas.length; i++) {
            soma += notas[i];
        }
        return soma / notas.length;
    } else return -1;
}
```


Criação de classes

- Os atributos (nome, notas e média neste exemplo) chamam-se variáveis de instância (instance variables) porque pertencem ao objecto como um todo e não a um método em particular
- As declarações de variáveis de instância começam geralmente com a palavra private, significando que são privativas do objecto, não sendo acessíveis directamente do seu exterior

Criação de classes

- Uma classe define o tipo de dados para um objecto, mas não armazena valores
- Cada objecto tem o seu único espaço de dados em memória
- Todos os métodos de uma classe têm acesso às variáveis de instância dessa classe
- Os métodos de uma classe são partilhados por todos os objectos dessa classe

Encapsulamento

- Visto do exterior, um objecto é uma entidade encapsulada, fornecendo um conjunto de serviços
- Estes serviços são conseguidos à custa dos métodos públicos do objecto
- Ao conjunto de serviços fornecidos por um objecto chama-se interface desse objecto
- Um objecto deve ser auto-contido, ou seja qualquer alteração do seu estado (das suas variáveis) deve ser provocada apenas pelos seus métodos
- Um objecto não deve permitir que outro objecto altere o seu estado
- O cliente de um objecto deve poder requisitar os seus serviços, mas sem saber como isso será conseguido

Encapsulamento

- O encapsulamento consegue-se à custa da utilização dos modificadores de visibilidade disponíveis na linguagem
- Em Java há três modificadores de visibilidade: public, private e protected
- O modificador protected será visto um pouco mais tarde
- Os membros de uma classe que forem declarados com o modificador public podem ser acedidos a partir de qualquer ponto
- Os membros declarados com private só podem ser acedidos de dentro do objecto

Encapsulamento

- Os membros declarados sem qualquer modificador têm visibilidade por defeito e podem ser acedidas a partir de qualquer classe dentro da mesma package
- De uma forma geral, as variáveis de instância dos objectos não devem ser declaradas com visibilidade public (antes com private)
- Os métodos que implementam os serviços fornecidos pelo objecto são declarados como public, por forma a poderem ser chamados a partir de outros objectos
- Os métodos públicos chamam-se serviços
- Os restantes métodos são métodos de suporte, servem como auxiliares dos serviços e não devem ser declarados como públicos

Variáveis e métodos de classe

- Em Java existe o modificador static que pode ser aplicado a variáveis ou métodos
- Serve para associar a variável ou método à classe e não a cada objecto da classe
- Se uma variável é declarada como static, apenas existe uma cópia para todos os objectos dessa classe
- Isto implica que alterar essa variável num objecto, provoca a sua alteração também nos restantes objectos dessa classe (só existe uma variável em memória)
  - private static int conta;
- Estas variáveis chamam-se também variáveis de classe

Variáveis e métodos de classe

- Normalmente chamamos um método através de uma instância da classe (objecto)
- Se um método é declarado como static pode ser chamado através do nome da classe, não sendo necessário que exista um objecto dessa classe
- Por exemplo, a classe Math tem vários métodos estáticos que podem ser chamados sem que seja criado um objecto desta classe:
  - Math.abs (num) - valor absoluto
  - Math.sqrt (num) - raiz quadrada
  - Math.pow (x, y) - potência

Variáveis e métodos de classe

- O método main é estático. É chamado a partir do sistema sem que este tenha que criar um objecto
- Os métodos estáticos não podem aceder a instance variables, uma vez que estas apenas existem nos objectos
- Podem utilizar variáveis declaradas com static (variáveis de classe) e também variáveis locais ao método
- Estes métodos são normalmente chamados métodos de classe