Classes abstractas

- As classes até agora apresentadas definem completamente a estrutura e o comportamento das suas instâncias (objectos)
- Há situações em que não é fácil determinar qual o código a colocar numa superclasse, mesmo sabendo que existe um dado conjunto de subclasses
- Por exemplo, considere-se um programa que necessite de manipular formas, nesta fase triângulos, quadrados e círculos

Classes abstractas

- É importante que estas formas "falem a mesma linguagem", ou seja implementem os mesmos métodos, nomeadamente (por exemplo) area () e perimetro ()
- Se as três classes implementam estes métodos faz sentido criar uma superclasse, seja Forma, para estas três classes
  - Desta forma será possível posteriormente criar classes que representem outras figuras (rectangulo por exemplo), sem que isso implique alterações no código existente e garantindo-se que a nova classe "fala a mesma linguagem" que as anteriores (pois herda os mesmos métodos)

Classes abstractas

- Se area () e perimetro () estiverem definidos em Forma garante-se que todas as suas subclasses, actuais ou futuras, terão estes métodos definidos
- No entanto, não é claro qual o código que estas classes devem conter, pois a forma de calcular a área e o perímetro varia entre as várias formas
  - Na verdade não há qualquer código comum possível na implementação destes métodos para as várias figuras
  - Portanto, apesar de ser correcto incluir estes métodos em Forma, não há qualquer código que faça sentido incluir nesses métodos
  - É uma situação em que faz sentido ter uma classe, mas ela não tem necessidade de ser codificada

Classes abstractas

- Classes abstractas são classes em que pelo menos um dos seus métodos de instância não se encontra implementado, mas apenas declarado
  - No exemplo, faz sentido colocar em Forma a declaração dos métodos area() e perimetro(), obrigando assim todas as suas subclasses a responderem a estes métodos, mas deixando para cada uma delas a obrigação de implementar a sua versão dos métodos
  - Dado que não implementam todos os seus métodos, as classes abstractas não podem ser usadas para criar instâncias
  - Em Java as classes e métodos são declarados como abstractos através da utilização do modificador abstract

Classes abstractas

- A classe Forma pode ser:

```
public abstract class Forma {
    public abstract double area();
    public abstract double perimetro();
}
```


Classes abstractas

- No exemplo, esta classe pode ser herdada pelas classes específicas de cada figura. Por exemplo:

```
public class Rectangulo extends Forma {
    private double comp, larg;
    public Rectangulo (double c, double l) {
            comp = c; larg = l;
    }
    public double area () {
            return comp * larg;
    }
    public double perimetro () {
            return 2*(comp+larg);
    }
}
```

Classes abstractas

- Uma subclasse de uma classe abstracta herda os seus métodos, podendo implementá-los ou não
  - Se implementar todos eles será uma classe concreta
  - Se deixar algum por implementar será também uma classe abstracta
- A relação de herança entre a classe abstracta e as suas subclasses é semelhante à relação entre qualquer duas classes, pelo que se pode fazer:

  Forma f = new Rectangulo (5,2);
- No entanto, f não poderá ser usado para chamar um método definido apenas em Rectangulo, pois f é do tipo estático Forma e o método não está definido nesta classe

Classes abstractas

- Resulta que é conveniente declarar o máximo de métodos abstractos, já que esta será a linguagem comum a todas as classes que venham a ser criadas
  - Este conjunto de métodos é o que garante compatibilidade a nível de polimorfismo
- As classes abstractas podem também ter variáveis de instância que são herdadas pelas suas subclasses
  - Isto implica que, apesar de não poderem gerar instâncias, as classes abstractas podem ter construtores, cuja função é inicializar as variáveis e que são chamados pelos construtores das subclasses

Classes abstractas

- Uma classe abstracta, ao não implementar todos os seus métodos, delega (ou força) as suas subclasses a fornecer a sua implementação particular desses métodos, facilitando o aparecimento de implementações diferentes nas várias subclasses
- Na herança entre classes concretas a redefinição de métodos é opcional
- Na herança a partir de classes abstractas a redefinição de métodos é obrigatória, de modo a haver uma implementação

Classes abstractas

- Em suma, as classes abstractas permitem:
  - Escrever especificações sintácticas para as quais são possíveis múltiplas implementações, de momento ou no futuro
  - Normalizar a linguagem a partir de certo ponto na hierarquia
  - Introduzir flexibilidade e generalidade no código
  - Tirar partido do polimorfismo
  - Realizar classificações mais claras de subclasses

Interfaces

- A palavra interface tem sido usada para representar o conjunto de serviços fornecidos por um objecto
- Ou seja, o conjunto de métodos que um objecto oferece define a forma como o resto do sistema interage com ele
- A linguagem Java tem um meio de formalizar este conceito
- Em Java, uma interface é uma colecção de métodos abstractos, ou seja é um conjunto de especificações sintácticas de métodos que representam um dado comportamento que qualquer classe pode implementar

Interfaces

- O conceito de interface permite definir um tipo de dados abstracto e também garantir que classes sem relações de herança apresentem comportamentos comuns
- As interfaces especificam um tipo de dados (conjunto de métodos abstractos a implementar) e qualquer classe que os implemente passa a ser compatível com esse tipo de dados
- Uma classe que implemente uma dada interface terá que fornecer uma implementação para todos os métodos definidos nessa interface

Interfaces

- Esta relação é especificada no cabeçalho da classe:

```
class NomeClasse implements NomeInterface {
}
```

- Por exemplo, definindo a seguinte interface:

```
// Interface Filosofo lista os métodos que têm que ser
// definidos por qualquer classe que o implemente
interface Filosofo {
    //Devolve uma citação de um dado filósofo
    String citacao();
} //interface Filosofo
```


Interfaces

- Podemos agora criar uma classe que implemente esta interface:

```
//A classe Descartes representa um seu pensamento
//Implementa a interface Filosofo, logo tem que definir um
//método citacao
class Descartes implements Filosofo {
    public String citacao() {
            return "Penso, logo existo";
    }
} // classe Descartes
public class Ideias {
    public static void main (String[] args){
            Descartes pessoa = new Descartes ();
            System.out.println (pessoa.citacao());
    }
} //class Ideias
```

Interfaces

- Uma interface pode ser implementada por muitas classes, mesmo que não sejam relacionadas por herança
- Cada uma dessas classes terá que fornecer a sua implementação dos métodos da interface
- Uma interface não é uma classe e, portanto, não pertence à hierarquia de classes e não pode ser utilizada para criar objectos
- Uma classe pode ser derivada a partir de uma super classe e implementar uma ou mais interfaces
- Uma interface pode ser derivada a partir de outra utilizando a palavra reservada extends
- A sub interface herda todas as definições da super interface
- Uma classe que implemente a sub interface tem que definir todos os seus métodos e os da super interface

Interfaces

- Se Filosofo é o nome de uma interface, pode ser utilizado como o tipo de um parâmetro de um método
- Um objecto de qualquer tipo que implemente Filosofo pode ser passado a esse método
- Uma classe pode implementar várias interfaces, devendo indicá-las separadas por vírgulas
- A possibilidades de implementar várias interfaces fornece muitas das características de herança múltipla (a capacidade de uma classe herdar várias classes)
- A herança múltipla não é suportada em Java

Classes abstractas e Interfaces

- Apesar de à primeira vista existirem semelhanças entre classes abstractas e interfaces, os dois conceitos são bastante diferentes:
  - Uma classe abstracta pode não ser 100% abstracta, mas uma interface é sempre 100% abstracta
  - Uma classe abstracta não impõe às suas subclasses a implementação obrigatória dos seus métodos abstractos, mas uma interface impõe a qualquer classe que declare implementá-la a implementação de todos os seus métodos abstractos
  - As classes abstractas pertencem à hierarquia de classes, enquanto as interfaces têm a sua própria hierarquia