import

As classes são referenciadas através dos seus nomes absolutos ou utilizando a primitiva import

```
import java.util.ArrayList
import java.util.*
```

A cláusula *import* deve aparecer sempre na primeira linha de um programa.
Quando escrevemos,
import java.util.*;
estamos a indicar um caminho para um pacote de classes permitindo usá-las através de nomes simples:
ArrayList al = new ArrayList();
De outra forma teríamos de escrever:
java.util.ArrayList al = new java.util.ArrayList();

Modificadores(setters) /Seletores(getters)

Após a atribuição do modificador de acesso private aos atributos, se o programador pretender permitir que estes possam ser acedidos a partir de outras classes, que não aquelas onde foram definidos, terá que recorrer aos setters e getters.

- **Getter**

método que devolve/obtém o valor atual de um atributo e é, na maioria dos casos, composto pela palavra get[nomeDoAtributo] (facilita a identificação). Quando o método devolve um valor do tipo boolean, a convenção passa a ser a palavra is[nomeDoAtributo], na Ide Eclipse.

Exemplo:

```
public float getRadius() {
return radius;
}
```

- **Setter**

método que modifica/atribui o valor de um atributo e deve ser composto pela palavra set[nomeDoAtributo] e o parâmetro do mesmo tipo do atributo (facilita a identificação).

Exemplo:

```
public void setRadius(float newRadius) {
      this.radius = newRadius;
}
```

Nota: Utiliza-se a palavra `this` para imputar ao atributo o valor que é passado como parâmetro (não é obrigatório, mas é uma boa prática usá-la, para não se confundir um atributo com outra variável definida na classe).

Boas práticas

- Devemos dar o mínimo de visibilidade pública no acesso a um objeto apenas a que for estritamente necessária;
- Por vezes, faz mais sentido criar um novo objeto do que mudar os atributos existentes

  Point p1 = new Point(2,3);☺
  p1.set(2,3); 😐

Este exemplo explica como aceder aos objetos instanciados e métodos de uma classe através de getters e setters

```
public class Puppy {
       int puppyIdade;

       public Puppy(String nome)
      {
          // Este construtor tem um parâmetro, nome.
          System.out.println("O nome escolhido foi:" + nome );
       }

       public void setIdade( int idade )
       {
          this.puppyIdade = idade;
       }

       public int getIdade( )
       {
          System.out.println("A idade do cachorro é :" + puppyIdade );
          return puppyIdade;
       }

       public static void main(String []args)
       {
          /* Criação do objeto */
          Puppy myPuppy = new Puppy( "Bobi" );

          /* Chamada do método da classe para atribuir a idade do cachorro */
          myPuppy.setIdade( 2 );

          /* Chamada do método da classe para obter a idade do cachorro */
          myPuppy.getIdade( );

          /* também se pode aceder à instância da classe a través da seguinte instrução */
          System.out.println("Valor da variável: " + myPuppy.puppyIdade );
       }
}
```

Output

```
O nome escolhido foi: Bobi
A idade do cachorro é: 2
Valor da variável:2
```

Relembra o exemplo da ficha 4 relativa aos modificadores de acesso

```
package algumPacote; // a classe definida neste arquivo pertence a algumPacote
public class NomeDaClasse
{
   public int w;            // acessível por qualquer classe
   protected int x;                // acessível pelas subclasses de NomeDaClasse
   int y;                            // acessível pelas classes deste pacote
   private int z;           // acessível somente pelos métodos de NomeDaClasse
   ...
}
```

Exercícios

Cria um novo projeto de Java de nome **java1_F8**

1. Adiciona uma nova classe de nome **Comida** no mesmo pacote, acrescentando o seguinte código à classe:

```
public class Comida {
	public final String unidade_peso ="g";
	public final String unidade_calorica ="cal";
	public String designacao;
	public int calorias;
	public int gramas;
}
```

2. Cria no mesmo pacote a classe ExecutarComida, contendo o método main, adicionando-lhe o seguinte código:

```
public class ExecutarComida {

	public static void main(String[] args) {
		Comida arroz = new Comida();
		arroz.designacao = "Arroz Carolino";
		arroz.calorias = 112;
		arroz.gramas = 100;

		System.out.println(arroz.designacao + " tem "+ arroz.calorias +
arroz.unidade_calorica +" em " + arroz.gramas + arroz.unidade_peso);
		}
}
```

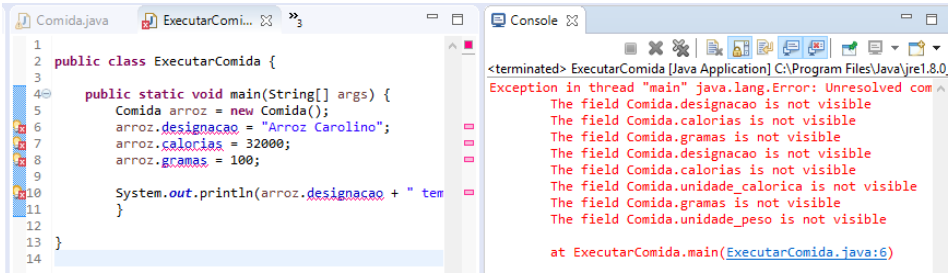
3. Executa o código
   Output: `Arroz Carolino tem 112cal em 100g`

4. Altera a visibilidade de todos os campos (propriedades ou atributos) para **private**:

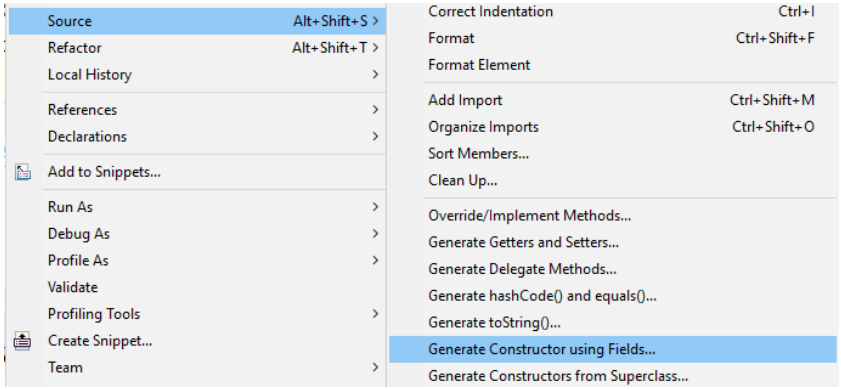
```
public class Comida {

	private final String unidade peso ="g";
	private final String unidade_calorica ="cal";
	private String designacao;
	private int calorias;
	private int gramas;

 }
```

5. Tenta executar o código.
6. Verifica que não consegues porque os campos são privados e não é possível ter acesso a estes fora da classe **Comida**. Isto é, o objeto **arroz** não tem acesso aos campos privados, como mostra a figura seguinte:

PROGRAMA OPERACIONAL CAPITAL HUMANO POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

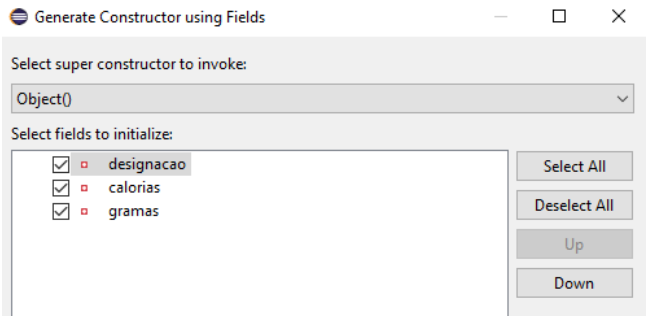


7. Coloca o cursor na classe Comida e com o botão do lado direito do rato escolhe a opção "*Source*" e seleciona a opção *"Generate Constructor using Fields… "* como mostra a figura seguinte:



Nota: podias aceder às opções de "Source" na barra de menu

8. Escolhe todos os campos/atributos como mostra a figura seguinte e pressiona o botão *OK*.




```
public Comida(String designacao, int calorias, int gramas) {
            super();
            this.designacao = designacao;
            this.calorias = calorias;
            this.gramas = gramas;
}
```

O código para o Construtor da classe Comida apresentou-se como se segue:
Recordar que o ***Construtor*** *é o método que é chamado, quando da instanciação dos objetos da classe, (com* ***new****) com os argumentos de acordo com o definido no método Construtor. Podemos ter vários construtores com diferente quantidade de parâmetros.*

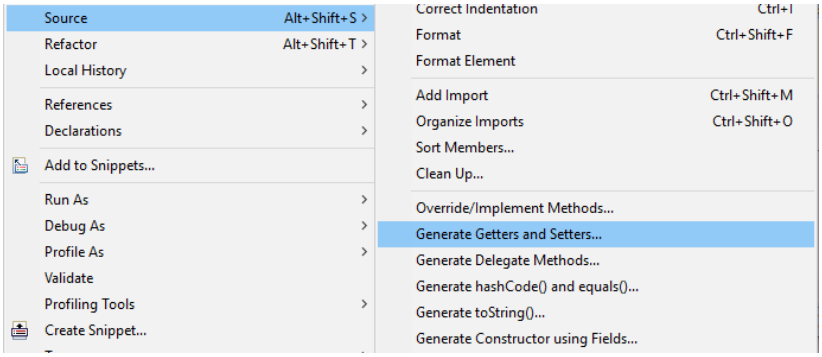
9. Na classe **ExecutarComida**, substitui o código:

```
Comida arroz = new Comida();
            arroz.designacao = "Arroz Carolino";
            arroz.calorias = 112;
            arroz.gramas = 100;
```

por

```
Comida arroz = new Comida("Arroz Carolino",112,100);
```

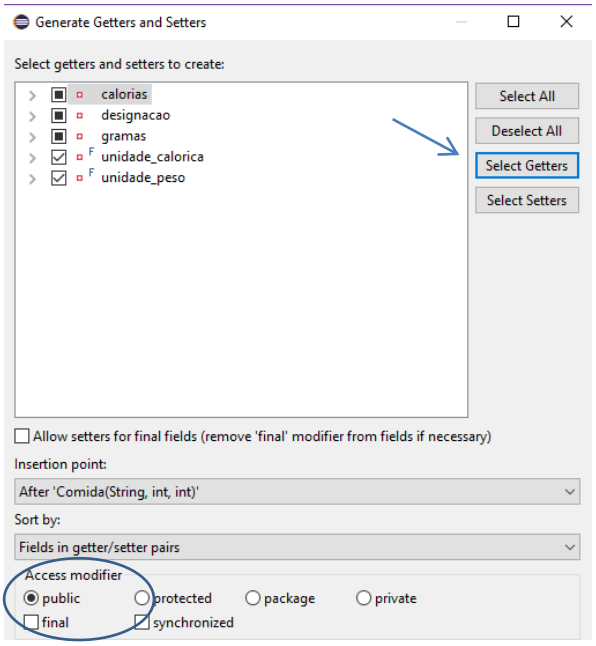
10. Repete o processo da questão 8 e escolhe "Generate *Getter and Setter…*"



11. Seleciona tudo como mostra a figura seguinte. Seleciona o botão Select Getters e clica em ok

Foi gerado o código seguinte:

```
	public String getUnidade_peso() {
		return unidade_peso;
	}

	public String getUnidade_calorica() {
		return unidade_calorica;
	}

	public String getDesignacao() {
		return designacao;
	}

	public int getCalorias() {
		return calorias;
	}

	public int getGramas() {
		return gramas;
}
```



Getters *- são os métodos que nos dão acesso para aceder (get em Inglês) aos dados privados dos campos da classe. Estamos assim a criar os nossos métodos para acesso à classe e assim encapsular a classe.*

Recorda: *encapsulamento é controlar o tipo de acesso às classes, atributos e métodos o que é conseguido através dos modificadores de acesso.*

12. Na classe **ExecutarComida**, substitui o código:

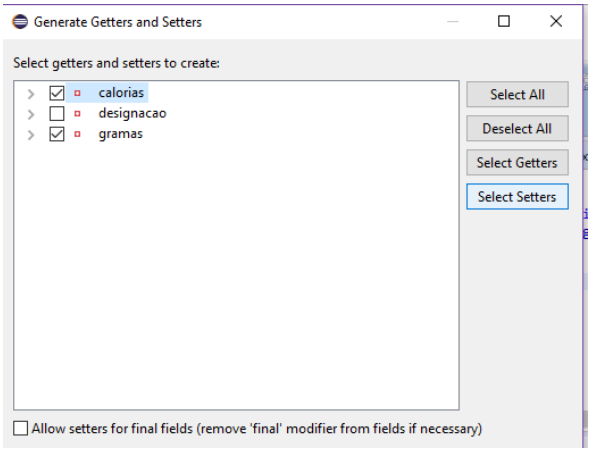
```
System.out.println(arroz.designacao + " tem "+ arroz.calorias +
arroz.unidade_calorica +" em " + arroz.gramas + arroz.unidade_peso);
```

por

```
System.out.println( arroz.getDesignacao() + " tem "+ arroz.getCalorias() +
arroz.getUnidade_calorica() +" em " + arroz.getGramas() + arroz.getUnidade_peso());
```

13. Executa o programa.

14. Introduz o método Setter, de modo a poder alterar os campos **calorias** e as **gramas** na classe Comida.



O código gerado será semelhante ao seguinte:

```
    public void setCalorias(int calorias) {
          this.calorias = calorias;
    }

    public void setGramas(int gramas) {
          this.gramas = gramas;
    }
```

15. Altera os campos do objeto arroz (no método main, da classe **ExecutarComida)**, utilizando os Setters criados na questão anterior, para:

Calorias: 56

Gramas: 50

```
…
Comida arroz = new Comida("Arroz Carolino",56,100);

//usar os setters aqui

     arroz.setCalorias(56);
     arroz.setGramas(50);
```

O output passa a ser:

```
Arroz Carolino tem 56cal em 100g
```


Bibliografia:
https://www.w3schools.com/java
https://www.tutorialspoint.com/java/
Jesus, C. (2013). Curso Prático de Java. Lisboa: FCA
Coelho, P (2016). Programação em JAVA – Curso Completo. Lisboa: FCA
https://www.devmedia.com.br/get-e-set-metodos-acessores-em-java/29241