

Instruções/ Tutoriais

O código desenvolvido foi no âmbito de uma cadeira de Programação Avançada no ano letivo de 2022/2023. De modo a ser utilizado é necessário inicializar uma classe de objetos, no caso do código fornecido é inicializado uma classe de objetos chamada UC, ou seja, Unidade Curricular, esta classe é composta por vários argumentos, dentro dos quais temos uma lista de alunos inscritos nessa mesma UC. De seguida é necessário transformar o objeto dado em JSONObject, esta classe transforma os valores do objeto inserido em JSON para que possa depois ser feito uma transcrição textual do mesmo.

```
val Aluno1 = Aluno(101101, "Dave Farley", true)
val Aluno2 = Aluno(101102, "Martin Fowler", true)
val Aluno3 = Aluno(26503, "André Santos", false)
val inscritos = listOf(Aluno1, Aluno2, Aluno3)
val unidade_curricular = UC("PA", 6.0, null, inscritos)

val obj = JSONObject(unidade_curricular)
```

- Através da função chamada de JSONObject_to_String é possível obter a transcrição textual típica de JSON como é esperado (Após desenvolver o modelo e passar como atributo à classe JSONObject).

```
val obj = JSONObject(unidade_curricular)
print(obj.JSONObject_to_String())
```

Output :

```
{
  "uc": "PA",
  "ects": "6.0",
  "data_exame": "null",
  "inscritos": [
    {
      "numero": 101101,
      "nome": "Dave Farley",
      "internacional": true
    },
    {
      "numero": 101102,
      "nome": "Martin Fowler",
      "internacional": true
    },
    {
      "numero": 26503,
```

```
"nome": "André Santos",
"internacional": false
}
]
}
```

- Foram desenvolvidas funções de visitantes, estas funções servem para efetuar pesquisas no modelo e verificar a estrutura de um certo modelo (Através dos testes criados é possível testar a validade de cada uma destas funções).
 - savedValues : Obter todos os valores guardados em propriedades com o identificador dado

```
@Test
fun testSavedValues() {
    val property = "internacional"
    val l = listOf(obj1, obj2, obj3, obj4)
    val expected = mutableListOf(JSONBoolean(true), JSONBoolean(true),
    JSONBoolean(false))
    assertEquals(expected, savedValues(property, l))
}
```

- typeEqualsProperty : Verificar que a propriedade dada só tem valores do tipo inserido

```
@Test
fun testOnlyIntNumbers() {
    val l = listOf(obj1, obj2, obj3, obj4)
    assertTrue(typeEqualsProperty("numero", "JSONInt", l))
}
```

- propertySameStructure : O array dado tem propriedades todas do mesmo tipo

```
@Test
fun testInscritosSameStructure() {
    val l = listOf(obj1, obj2, obj3, obj4)
    assertTrue(propertySameStructure("inscritos", l))
}
```

- allObjectsWithProperty : Obter todos os objetos que têm as propriedades dadas

```
@Test
fun testAllObjectsWithProperty() {
    val properties = listOf("numero", "nome")
    val l = listOf(obj1, obj2, obj3, obj4)
    val expected = listOf(obj1, obj2, obj3)
    assertEquals(expected, allObjectsWithProperty(properties, l))
}
```

- Para além disto ainda foram utilizadas anotações com os seguintes objetivos :
 - @JSONExcluir : Excluir propriedades da instanciação

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.PROPERTY,
AnnotationTarget.FUNCTION )
annotation class JSON_Excluir
```

- @DataClass : Utilização de identificadores personalizados (e não os das classes)

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.PROPERTY,
AnnotationTarget.FUNCTION )
annotation class DataClass(val id: String)
```

- @ForceJSONString : Força que alguns valores sejam considerados strings JSON

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.PROPERTY,
AnnotationTarget.FUNCTION )
annotation class ForceJSONString
```

- Por fim, desenvolvemos uma interface gráfica de modo a eliminar, adicionar e modificar o JSONObject, de modo a poder realizar o mesmo foram utilizados observers que através da alteração do modelo na parte gráfica altera diretamente o modelo em si e por consequência o modelo de texto localizado na parte direita.
 - Para o seguinte ser possível foi criada uma classe JSONObjectObserver que permite:

```
interface JSONObjectObserver {
```

- adicionar elementos ao modelo

```
fun propertyAdded(pair: Pair<String, Any>, index: Int?, p: JPanel, undo:
Boolean = false) {
}
```

- remover elementos do modelo

```
fun propertyRemoved(pair: Pair<String, Any>) {
}
```

- atualizar os valores presentes no modelo

```
fun updateObject(pair: Pair<String, Any>) { }
```

- Assim que a interface gráfica é criada é adicionado um observer ao modelo para fazer com que o mesmo seja observável, assim, o campo textual do lado direito é atualizado automaticamente

Josue - JSON Object Editor

ucPA

ects 6.0

data_exame N/A

inscritos

numero 101101

nome Dave Farley

internacional ☒

numero 101102

nome Martin Fowler

internacional ☒

numero 26503

nome André Santos

internacional ☐

```
{
  "uc": "PA",
  "ects": "6.0",
  "data_exame": "null",
  "inscritos": [
    {
      "numero": 101101,
      "nome": "Dave Farley",
      "internacional": true
    },
    {
      "numero": 101102,
      "nome": "Martin Fowler",
      "internacional": true
    },
    {
      "numero": 26503,
      "nome": "André Santos",
      "internacional": false
    }
  ]
}
```