Teste

Programação C# (Parte 2)

**Regras**

Este teste tem como intuito avaliar o teu conhecimento e capacidades acerca das bases de programação em C#. Deverás, até as 12h50 de hoje, entregar a sua resolução por email, ou até por repositório até ao final, ou não será contabilizada. Durante este período, poderás consultar os materiais lecionados (slides), tendo sempre em consideração o tempo que te resta até à entrega. Se sentires que consegues escrever mais depressa em papel, poderás então escrever as tuas respostas numa folha, tirar foto ou digitalizar, enviando juntamente com a solução. Caso tenhas algum problema notifica-o no Slack. Um dos teus colegas poderá estar na mesma situação! Assim que entregares, notifica-me.

**Atenção :** o email enviado com a solução para a ficha deverá conter apenas um anexo (zip) com o nome FICHA\_CSHARP\_PrimeiroNome\_UltimoNome (ex: FICHA\_CSHARP\_Fabio\_Jesus.zip)

A ficha é composta por 4 (quatro) grupos:

1. **Verdadeiros e Falsos (40 pontos)**

As respostas deverão ser colocadas nos respetivos campos da grelha.

1. **Desenvolvimento (60 pontos)**

A resposta deve ser colocada abaixo da pergunta, ou se escreveres a resposta numa folha, marca apenas o número da questão. (ex: 1))

1. **Prático (100 pontos)**

As respostas devem ser colocadas no código fonte. Caso ocorra algum erro que cause o teu projeto a funcionar, respira fundo, comenta o código e continua. Não deixes que um erro mínimo te impeça de continuar o teste, pois todo o código comentado será avaliado, e caso esteja parcialmente correto, será atribuída essa pontuação. Perguntas que tenham cotações diferentes apenas totalizam a pontuação total se forem apresentadas soluções para cada uma (ex: [iterativa 5pts / recursiva 10pts] resulta em 15 pontos se entregares ambas).

1. **Extras (50 pontos)**

São pontos extra, por isso, tal como nas outras fichas e testes, nunca contam para além de demonstrar o teu esforço. Aconselho-te a resolver estes exercícios assim que acabares o teste.

Boa sorte!

# **Grupo I – Verdadeiros e Falsos (40 pontos)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **V** | **F** | **F** | **F** | **V** | **F** | **V** | **V** |

* 1. É possível, através de um índice, alterar um caracter de uma string.
  2. Os vetores são de dimensão fixa.
  3. A class Path permite criar e eliminar pastas.
  4. Na especificação das funções que arredondam valores, o floor e o ceiling arrendondam de forma diferente com base no seu sinal
  5. Os métodos de extensão são utilizados para adicionar novas funcionalidades a classes existentes.
  6. As exceções são lançadas com o launch
  7. Os tipos genéricos permitem o desenvolvimento de funções que podem ser utilizadas em diferentes tipos, que serão posteriormente especificados.
  8. Os enums permitem valores repetidos

# **Grupo II – Desenvolvimento (60 pontos)**

1. Descreve o namespace. Apresenta exemplos. **[10 pontos]**

É um diretório que fornece contexto para os itens armazenados permitindo a desambiguação dos mesmos, mantendo organizadas as classes e ajuda no controlo do escopo da classe e métodos de grandes projetos.

O uso do termo “using” permite a importação de outros namespaces.

O namespace é construído de forma hierárquica:

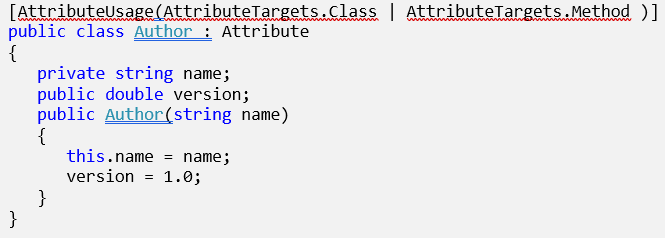
1. Build-in : é relacionado com soluções, (ex.Recodme.Formacao - onde ficam todas as soluções relacionadas com a formação).
2. Global namespace: relacionado com projetos, (ex. Recodme.Formacao.Worksheets - que armazenam todas as fichas do curso).
3. Local namespace: namespaces definidos nas classes atribuído pelo programador.
4. Descreve o atributo. Apresenta exemplos **[4 pontos]**

São tags usadas para associar o código de forma declarativa, sendo reutilizáveis. Apos associado é acedido por técnicas de reflexão.

Os atributos identificam métodos específicos(endpoints e controllers)

Descrevem a forma como o código é operado nativamente

Descreve um assemby em termos



1. Distingue delegate, event, action e func. **[8 pontos]**

Delegate – delega uma ação que vai ocorrer

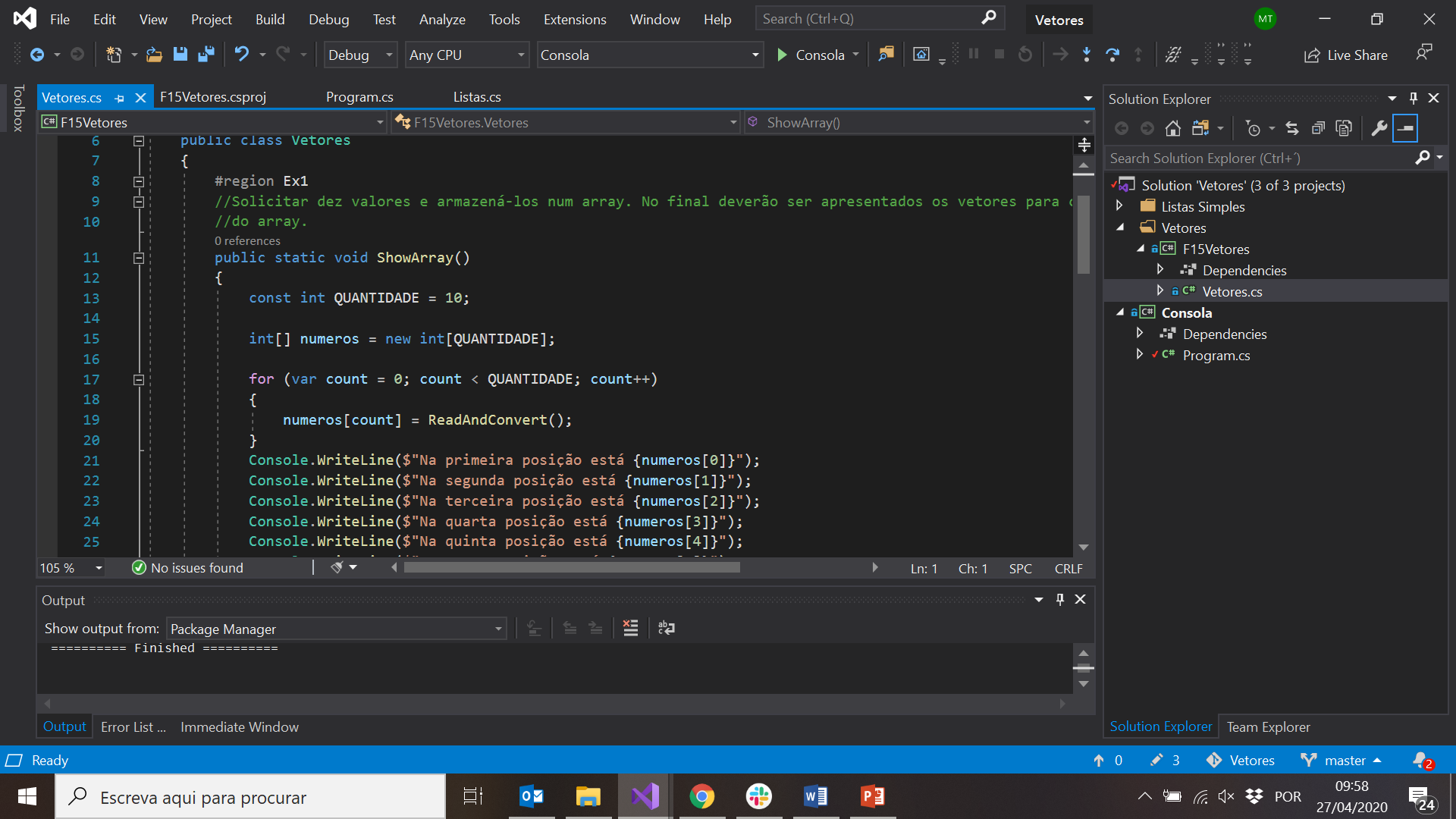
Event – executa a ação

Action – faz alguma coisa, mas não retorna um valor

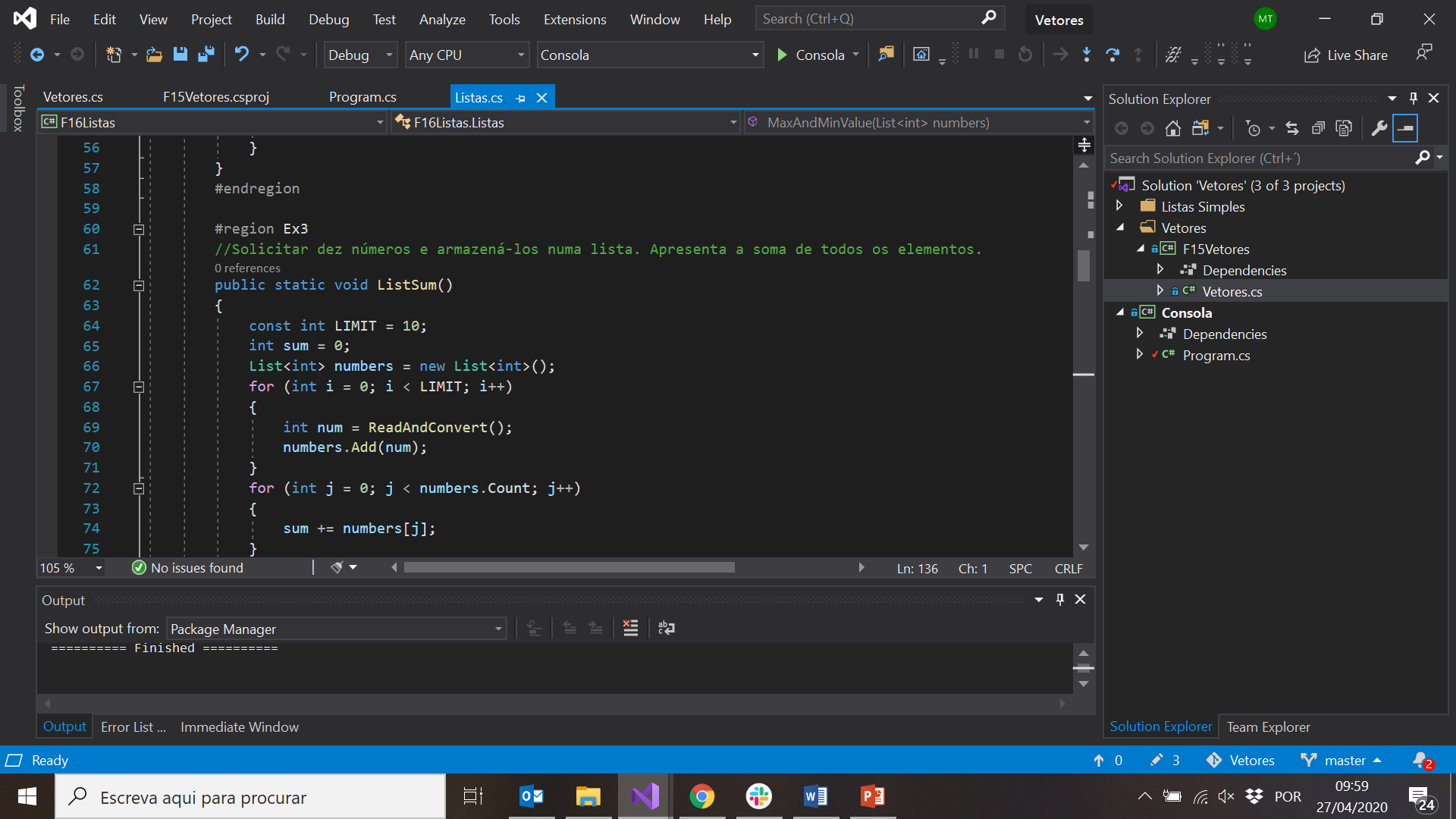
Funcs – retorna um valor

1. Distingue listas de arrays. Exemplifica **[8 pontos]**

Arrays é uma estrutura de dados que armazena uma coleção de elementos de tal forma que cada um dos elementos possa ser identificado por, pelo menos, um índice ou uma chave.

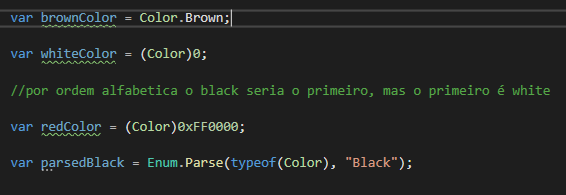


Listas é uma estrutura de dados semelhante a um array que se expande ou encolhe conforme a necessidade, aceitando valores duplicados.



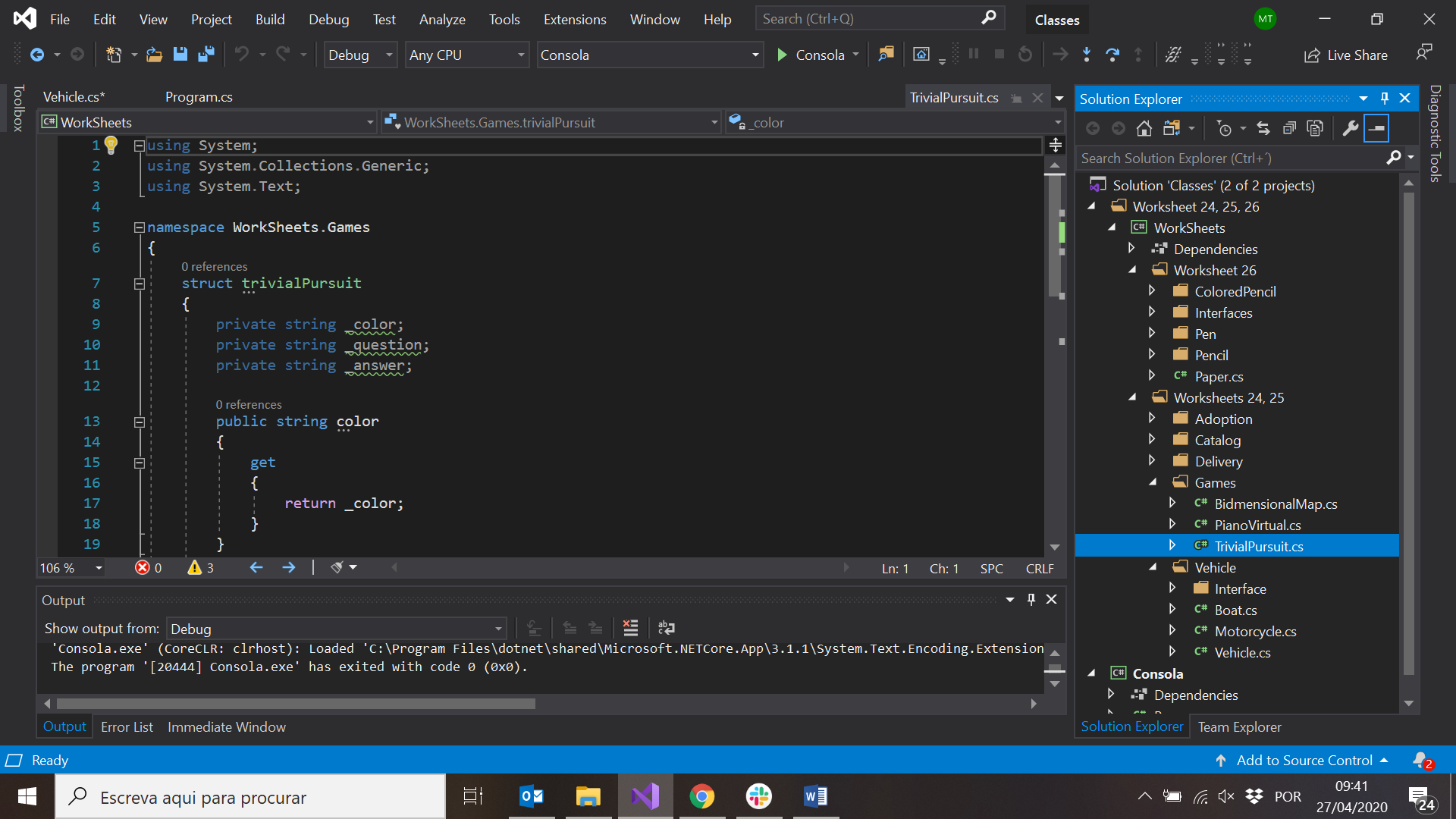
1. Descreve o tipo enum. Exemplifica **[6 pontos]**

É um tipo de valor definido através de um conjunto de constantes. Os nomes são distintos e escondem um valor numérico. Após sua implementação seus valores podem ser acedidos através de [nome].[valor] e casting

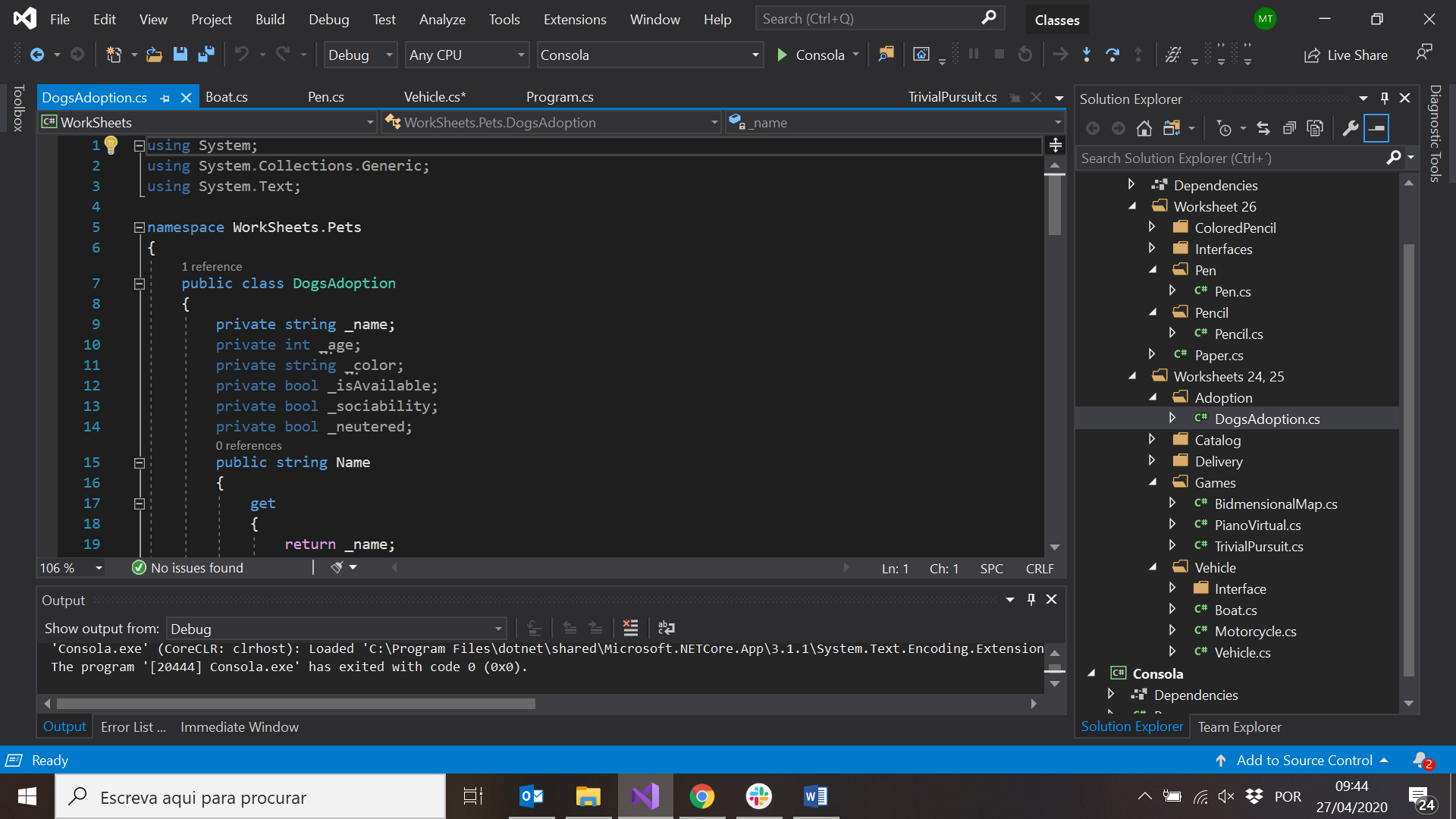


1. Distingue classes de structs. Demonstra **[8 pontos]**

Structs são utilizadas para criar estruturas de dados, onde as instancias são pequenas e imutáveis. Não podem conter construtor sem parâmetros, a copia da estrutura é feita por valor, podem ser instanciadas sem o uso do termo “new”, não é submetida a herança, não implementa interface e não pode ser nula.



A classe é composta por características(propriedades) e comportamentos(ações), funciona como um molde que cria objetos referenciados da classe em memória, cada objeto tem um estado próprio, cada propriedade tem um tipo e um nome que a identifica, cada comportamento é descrito por um verbo e realiza uma ação sobre o objeto.



1. O que é açúcar sintático? Dá exemplos do seu uso. **[10 pontos]**

São características da linguagem capazes de facilitar o processo.

Getters e Setters para acessar uma propriedade

Uso de Using para limpar a memória

Chamada através do método this.method()

1. Descreve o LINQ. Exemplifica **[6 pontos]**

É um dos elementos ligados ao conceito de açúcar sintático. Efetua operações de pesquisa sobre dados ou informações de coleções. Cada operação é executada em três passos:

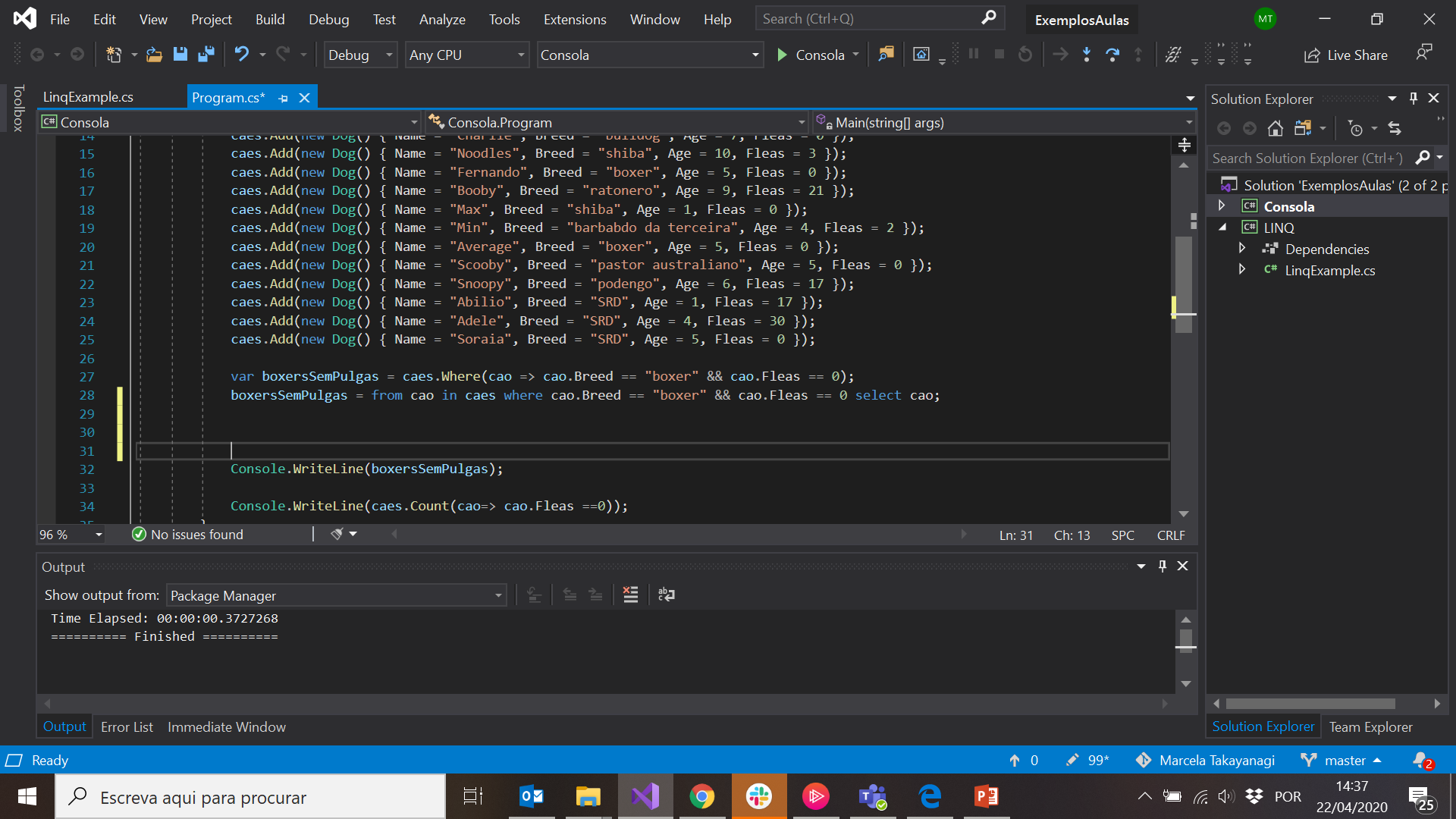
Obter informações

Criar operações

Executar operações

É mais comum na aplicação à base de dados e ficheiros.

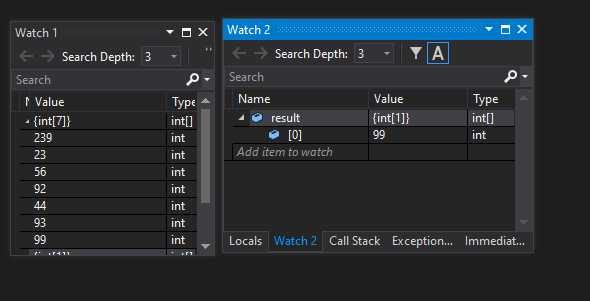
Ex. Contar um número específico de elementos com base em uma característica.



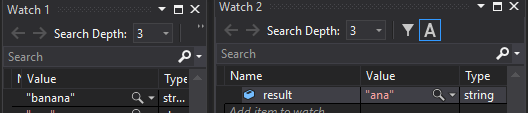
# **Grupo III – Desenvolvimento (100 pontos)**

1. Avaliação do código em geral **[10 pontos]**
   * Disposição de elementos nos ficheiros adequados
   * Respeito às normas de programação definidas
   * Uso do conteúdo lecionado de forma correta e responsável
   * Legibilidade e açúcar sintático
2. Os exercícios deverão ser colocados na pasta FirstExercise:
   1. No ficheiro OneA, sem recorrer a LINQ, cria uma função que receba um conjunto de inteiros e retorne apenas os valores > 50, ímpares e divisíveis por 9; **[6 array / 4 list]**

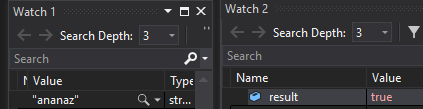
No seguinte exemplo, temos um conjunto original com 7 numeros à esquerda, e o resultado da operação à direita.



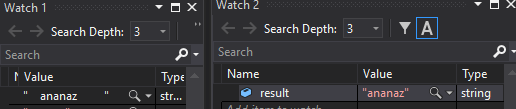
* 1. No ficheiro OneB cria funções que recebam uma string e: **[5 pontos]**
     1. Retorne apenas a metade (arredondada para baixo) da string



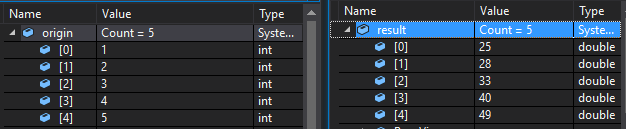
* + 1. Valide se a string começa com “a” e acaba com “z”



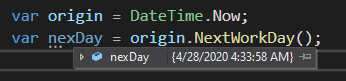
* + 1. Remova os espaços à volta da string e retorne a string “limpa”



* 1. No ficheiro OneC cria uma função que receba uma lista de inteiros e retorne a lista com o resultado da operação 22+8\*3 **[3 pontos / 2 pontos LINQ]**



* 1. Cria um método de extensão que permita obter a próxima data útil (se amanhã for sábado, ou domingo, retorna a data da próxima segunda). **[15 pontos]**



1. Os exercícios deverão ser colocados na pasta SecondExercise:

Tenho uma gaveta com várias divisórias. Como o seu interior é sensível, não quero colocar objetos que sejam afiados. Por vezes, quando a tento abrir fica presa, e assim não consigo aceder ao conteúdo.

* 1. Cria uma interface que represente algo seja afiado, capaz de cortar **[6 pontos]**
  2. Cria uma classe que identifique uma faca, através de propriedades e métodos que aches necessários **[10 pontos]**
  3. Cria um atributo chamado Stuck, que apenas pode ser aplicado a classes, não podendo existir, explicitamente, mais que um destes atributos associados a uma classe. Aplica-o à classe Drawer. **[7 pontos]**
  4. Cria sete classes distintas, onde três das quais devem implementar a interface produzida. Não é necessário elaborar muito! **[7 pontos]**
  5. Altera a classe Drawer, para que, através de um indexador seja possível organizar os objetos em divisórias. Não deverá permitir que sejam introduzidos objetos afiados **[15 pontos]**
  6. Cria uma exceção que indique que a gaveta está presa. Implementa uma forma de lançar a exceção quando é necessário aceder a uma divisória da gaveta, com base num valor aleatório. **[7pontos]**

# **Grupo IV – Extras (50 pontos)**

1. Às 13h00, coloca a tua resolução num repositório do GitHub, enviando o link juntamente com a resolução. **[5 pontos]**
2. Documenta todas as funções com as quais interagiste durante o teste **[5 pontos]**
3. Altera os namespaces dos projetos de forma a que estes sejam representativos da solução e dos projetos. **[10 pontos]**
4. Na biblioteca Extras completa o código da classe ContentSerializer. Altera apenas o código na zona indicada pelos comentários. **[15 pontos]**
5. Os exercícios deverão ser colocados na pasta ThirdExercise:
   1. Cria uma classe que te permita realizar as operações básicas (soma, subtração, multiplicação e divisão) a uma calculadora. Esta deverá manter apenas o valor em memória, que é atualizado após realizar uma operação. Deverás aplicar os conceitos que deste de forma a organizar o teu código da melhor forma. **[10 pontos]**
   2. Atualiza a tua classe para adicionares os mesmos métodos, sem que seja necessário um objeto. **[5 pontos]**