# Практически проект: Падащи камъни

Днес ще разработим конзолната игра "**Падащи камъни**":



"**Падащи камъни**" е игра, чието цел е да **избягвате падащи камъни** и да **оцелеете възможно най-дълго**, за да изкарате **най-висок резултат**.

Graphical user interface, text

Description automatically generated

Background pattern

Description automatically generated Background pattern

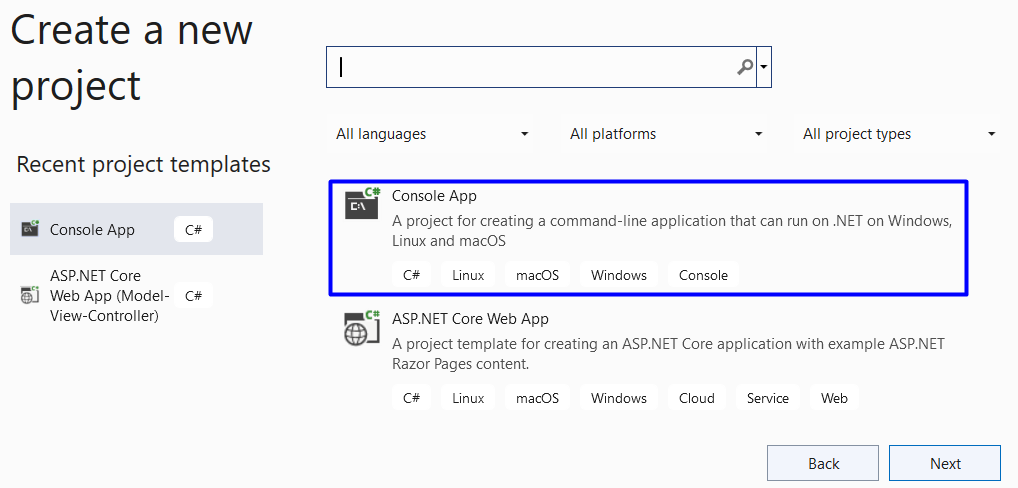
Description automatically generated

## Писане на кода на играта

Нека да започнем да създаваме нашата игра.

### Създаване на Visual Studio проект

Първо трябва да **стартираме Visual Studio** и да създадем **ново C# конзолно приложение**:



Създайте **.NET 6 Console App** и преди да продължим, променете името на **главния клас** от Program.cs на нещо **по-смислено**, например **FallingRocks.cs**.

## Създаване на GitHub репо

Изберете **един от начините**, които вече сте учили, и **създайте GitHub репо** за проекта – от сайта на **GitHub** или директно през **Visual Studio**.

|  |  |
| --- | --- |
| Icon  Description automatically generated | Важно е да изберете **ваше оригинално име** за проекта!  Вашият GitHub профил трябва да бъде **уникален** и да не бъде същият като този на вашите съученици.  Може да следвате тези инструкции за разработка на проекта, но можете и да **правите промени** и да **имплементирате проекта по различен начин** от вашите съученици. |

Graphical user interface, text, application, email

Description automatically generated

### Стъпки за имплементация на играта

Ще имплементираме **два класа**.

Първият, **class Player**, ще представлява **играча**, който се **движи** чрез **лявата** и **дясната стрелка** на клавиатурата, за да **избягва камъните**. Класът **Player** ще има:

* Конструктор, който приема:
  + X координата на играча
  + Y координата на играча
* **Четири** свойства:
  + int X – **хоризонталната координата** на играча
  + int Y – **вертикалната координата** на играча
  + string Representation – означението на играча чрез символи
  + bool HasBeenHit – връща дали играчът и някой камък са на една и съща позиция (тогава има сблъсък)
* Четириметода:
  + void Draw() – рисува играча на конзолата с текущите **X** и **Y** **координати**
  + void Move() – движи играча в зависимост от стрелката на клавиатурата, която е натисната
  + bool IsOutOfConsoleBoundaries() – проверява дали **текущата позиция** на **играча** е **извън границите** на конзолата
  + void ClearKeyAvailableBuffer() – изчиства буфера с метода Console.KeyAvailable()

Вторият клас, Rock, представлява **камъните**, които **падат** от горната част на конзолата. Класът **Rock** ще има:

* **Конструктор**, който приема:
  + **X** **координатата** на камъка
* Три **свойства**:
  + int X – **хоризонталната координата** на камъка
  + int Y – **вертикалната координата** на камъка
  + char Symbol – означението на всеки камък чрез символ

След като сме **готови с класовете**, ще имплементираме **главната логика** на играта. За нея ще ни трябват **четиринадесет метода**, които ще си взаимодействат, за да **работи играта правилно**.

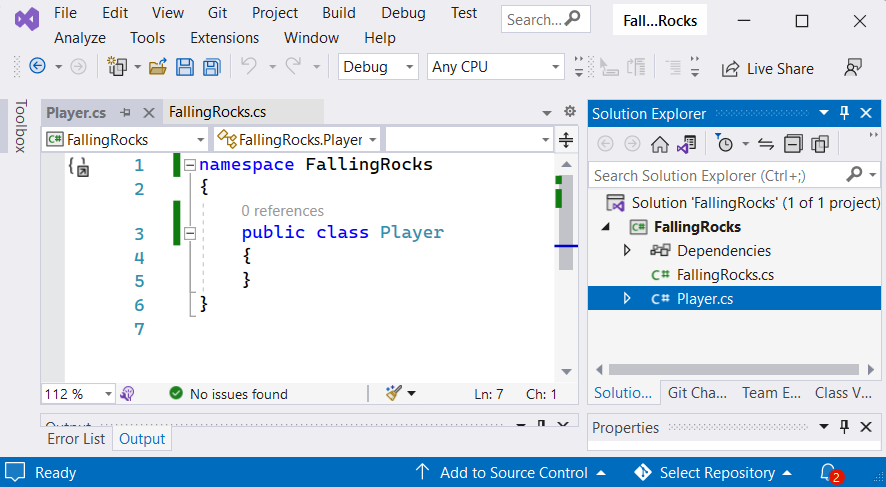
* int ProcessPlayerRocksSpawnRateChoice(string inputSpawnRate) – обработва **входа** на играча за **скоростта на генериране на камъни**
* int ProcessPlayerRockFallSpeedChoice(string inputFallSpeed) – обработва **входа** на потребителя за **скоростта на падане** на камъните
* void SetWindowProperties() – задава **ширината** и **височината** на **конзолата**
* void RedrawConsole() – рисува **наново конзолата**, за да избегнем **премигване на конзолата**
* void DrawScorePanel() – рисува **панел с резултата**, където ще се визуализират **точките**
* void CreateRocks() – **създава камъни**, за да можем да ги **нарисуваме на конзолата**
* bool ShouldGenerateRock() – дава знак на програмата кога да **генерира нови камъни**
* void DrawRocks() – **рисува** генерираните **камъни**
* void MoveRocks() – **премества** **камъните**
* bool ThereIsCollision(Rock rock, Player player) – проверява дали има **сблъсък** между **играча** и някой **камък**
* bool RockAndPlayerAreOnSameWidth(Rock rock, Player player) – проверява дали **играчът** и **камъкът** са на **една и съща ширина**
* bool RockAndPlayerAreOnSameHeight(Rock rock, Player player) – проверява дали **играчът** и **камъкът** са на **една и съща височина**
* void RemoveRocks(List<Rock> rocksToRemove, List<Rock> rocks) – **премахва камъните**, които са **задминали играча**
* void EndGame() – изписва "**Game Over!**", след като **играчът** **загуби**

Това е **всичко**, от което се нуждаем. Нека да продължим с **имплементацията**.

## Дефиниране на класовете

### Class Player

Нека да **създадем клас** и да го наименуваме **Player**. Вече знаете как да го направите:



Първото нещо, което ще направим в класа, е да **добавим свойствата**, които са ни нужни. **Първите две** свойства ще бъдат **X** и **Y** (**координатите на играча**). Те ще бъдат от **тип** int и ще ни помогнат да знаем **къде** **на конзолата** трябва да бъде **играчът**, за да го **нарисуваме**.

Text

Description automatically generated with medium confidence

Следващото свойство е Representation. То ще бъде от **тип string** и ще съдържа **означението** на играча чрез **символи**.



**Последното** свойство, което ще ни трябва в този клас, еHasBeenHit. То ще бъде от **тип bool** и ще ни показва дали **играчът** е бил **ударен от камък**. Първоначално това свойство ще има **стойност false**, в противен случай играта **няма да започне**, защото **главната логика** на играта ще установи, че **играчът е ударен**, и ще **прекрати цикъла**.



Готови сме със **свойствата**. Сега трябва да добавим **конструктора**, който **приема x** и **y координатите**:



Следващата стъпка е да зададем подадените през **конструктора x** и **y координати** като стойностина **X** и **Y свойствата**:

Text

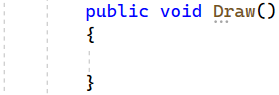
Description automatically generated with low confidence

Накрая задаваме свойството **Representation**, като избираме **символ**, с който да **означим играча** на конзолата. В този пример ще използваме "(0)".

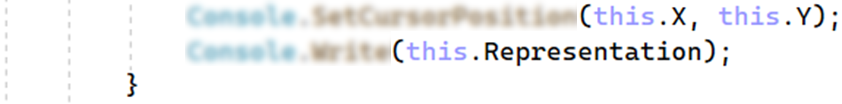


Това е всичко, от което се нуждаем за нашия **конструктор**. Следващата стъпка е да **създадем методите**.

Създайте метода Draw(). Този метод ще бъде от тип **void** и ще **рисува на конзолата** символите на **играча** в зависимост от **X** и **Y координатите**.



В този метод трябва да зададем **позицията на курсора** на текущите **X** и **Y** **координати** и да **напишем на конзолата** избраните **символи** (**репрезентация**):



Това е **всичко за този метод**. Нека да продължим със **следващия**.

Създайте метода Move(). Той ще бъде от тип **void** и ще **движи** играча **наляво** или **надясно** в зависимост от това **коя стрелка** на клавиатурата е **натисната**.

A picture containing logo

Description automatically generated

В този метод първо **проверяваме** дали **има натиснат клавиш**. Можем да го направим с Console.KeyAvailable:

A picture containing text

Description automatically generated

Ако има **натиснат клавиш**, трябва да го **запазим** в **променлива**, използвайки метода Console.ReadKey(). Тази променлива ще бъде от **тип** ConsoleKeyInfo. Трябва да изглежда така:



Следващото нещо, което ще използваме, е метода ClearKeyAvailableBuffer()– той ще **пречи** на **ConsoleKey** да запазва **натиснатите клавиши**. Тъй като този метод **още не е имплементиран**, само го **извикайте** и след това го **закоментирайте**.

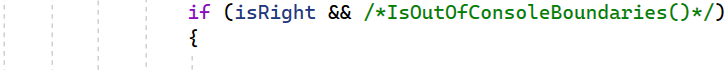


Сега трябва да създадем **две булеви променливи**. Първата ще проверява дали **натиснатият клавиш** е **дясната стрелка** (→), а втората ще проверява дали **натиснатият клавиш** е **дясната стрелка** (→). Тези проверки за **важни**, защото ако ги нямаме, играта ще се **счупи**, ако потребителят натисне **някой друг клавиш**.

Може да го направим, като достъпим **свойството** Keyна **променливата** pressedKey и проверим дали е **равно** на ConsoleKey.RightArrow и респективно LeftArrow. Кодът трябва да изглежда така:



Сега трябва да проверим дали **натиснатият клавиш** е **дясна стрелка** и дали не е **извън границите**. Можем да проверим дали е **извън границите** с метода IsOutOfConsoleBoundaries(). Тъй като методът **още не е имплементиран**, го **извикайте** и го **закоментирайте**.



Ако проверката върне стойност **true**, **добавяме** **1** към свойството **X**.

A screenshot of a computer

Description automatically generated with medium confidence

Направете същото за **лявата стрелка**, само че **извадете 1** от свойството **X**. Също помислете как да проверите дали играчът е **извън лявата граница**.

A picture containing graphical user interface

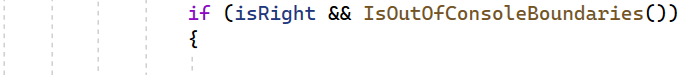
Description automatically generated

Това е всичко за този метод. Сега нека да имплементираме метода IsOutOfConsoleBoundaries(), за да можем да **откоментираме** извикването му от метода **Move()**. Методът ще ни покаже, ако достигнем **максималната ширина** на конзолата. Трябва да направим тази проверка, иначе играчът ще излезе извън конзолата и играта няма да работи. Методът ще върне стойност **true**, ако текущата **X координата** **плюс** дължината на **Representation** е **по-малка** от **ширината на конзолата минус 1**. Ще използваме ламбда израз. Методът трябва да бъде **частен**, защото не искаме да се достъпва от друг клас освен от **Player**.

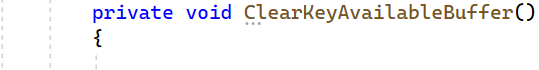
Трябва да изглежда така:



След като **приключим с метода**, можем да се върнем в метода Move() и да **откоментираме** тази част**.**



Сега само трябва да имплементираме метода ClearKeyAvailableBuffer(). Този метод е необходим, защото ще попречи на **ConsoleKey** да запази **натиснатите клавиши** (ако някой клавиш е задържан натиснат, ConsoleKey ще запомни, че този клавиш е **натиснат** и **няма да можете да преместите играча**). Този метод трябва да е **частен**, защото не искаме да се достъпва от друг клас освен от **Player**.



В метода трябва да създадем while-цикъл, който ще итерира докато има **наличен клавиш**.

A picture containing text

Description automatically generated

В цикъла ще **четем клавишите**.

A picture containing text

Description automatically generated

В момента, в който се **натисне клавиш**, цикълът ще **спре** и играчът ще се **премести** в **избраната посока**. Това е **всичко за този метод**. Върнете се в метода Move() и **откоментирайте** новосъздадения метод.

Готови сме с класа **Player**. Следващата стъпка е да имплементираме класа **Rock**.

### Клас Rock

Създайте **нов клас** и го наименувайте **Rock**. Вече знаете как да направите това:

A picture containing text

Description automatically generated

Първо, нека да добавим **три свойства**. **Първите две** ще бъдат от тип **int** и ще представляват **X** и **Y координатите** на **камъка**.

Text

Description automatically generated with low confidence

**Третото свойство** ще бъде от тип **char** и ще включва **символа за камъка**.



Преди да **създадем конструктора**, се нуждаем от **две частни полета**. Първото ще бъде от тип char[] и ще съдържа **всички различни символи за камъка**.



**\* Бележка:** Може да **копирате** **тези символи** или да **добавите ваши**, така че проектът ви да е **по-различен** от този на вашите съученици.

|  |
| --- |
| '^', '@', '\*', '&', '+', '%', '$', '#', '!', '.', ';' |

Следващото **частно поле** ще бъде от тип **Random** и ще ни помогне да вземем **нов символ** всеки път, когато **създаваме нов камък** (за да може символите на падащите камъни да бъдат **различни**).



Вече имаме **всички свойства** и **полета**. Остава да добавим **конструктор** на класа.

Създайте **конструктор** и го направете така, че да **приема X координата** на камъка. Тук **нямаме нужда от Y координата**, защото камъните винаги **започват да падат** от **Y = 0**.

A picture containing text

Description automatically generated

Сега в конструктора трябва да зададем **стойност на свойствата**, както в предишния клас. Този път **Y координатата** ще е **равна на 0**.

Icon

Description automatically generated

Свойството Symbol ще бъде равно на **елемент** на **случаен индекс** от нашия **Representation масив**. Вече знаете как да го вземете:



Това е **всичко**, от което се нуждаем в класа **Rock**. Нека да продължим с **инициализацията** на **играта**.

## Инициализация на играта

### Подготовка на настройките

Първо, кликнете на файла "FallingRocks.cs" в [SolutionExplorer]. В този файл ще **инициализираме играта**. Първо ще имплементираме **главната логика** на играта, като ще **закоментираме** някои части. След като **имплементираме методите**, ще ги **откоментираме** и ще **тестваме** играта.

Нуждаем се от **три** int **константи**, които ще ни помогнат да **зададем ширина** и **височина** на конзолата, както и **размера** на **панела с резултати**. Ще използваме тези константи **по-късно**.

Text, chat or text message

Description automatically generated

**\* Бележка:** Може да използвате **същите числа** или да зададете **различни**.

Следващата променлива е от тип int и ще пази **резултата** на потребителя (променливата ще се **увеличава с 1**, когато играчът **успешно заобиколи камък**).



Сега трябва да **прочетем входа** на потребителя за **скоростта** на **генериране** на **камъни** и **скоростта** на **падане** на всеки камък (трудността на играта). Вместо да пишем всичко тук, нека да създадем **два метода**, които да отговарят за това. По този начин правим кода **по-четим**.

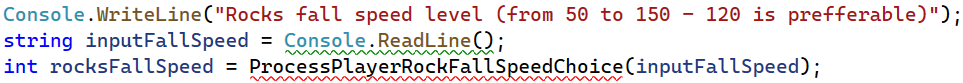
Преди да използваме **първия метод**, трябва да **напишем съобщение** на **конзолата**, за да обясним на потребителя **какво да въведе**, след което да **прочетем входа**.



Сега създайте int **променлива** за **скоростта** на **генериране** на камъни и я направете **равна** на **резултата от** ProcessPlayerRocksSpawnRateChoice(string inputSpawnRate).



Направете същото нещо за **скоростта** на **падане**. За целта използвайте метода ProcessPlayerRocksFallSpeedChoice(string inputFallSpeed).



**\* Бележка** – Закоментирайте **методите**, за да не получите **грешки**.

Следващата стъпка е да зададем свойствата на **конзолния прозорец**. Ще използваме метода SetWindowProperties().



Сега се нуждаем от **още три** int **променливи**, които ще ни помогнат да зададем X и Y координатите на **играча**. За да вземем **X координатата**, се нуждаем от **ширината** на конзолния прозорец, **разделена на 2**.



За да вземем Y **координатата**, се нуждаем от **височината** на конзолния прозорец минус **размера** на **панела с резултати** (имаме константа за това).



За да вземем **точната Y координата**, трябва да **извадим 1** от резултата (иначе играчът ще се **позиционира** на **реда** на **панела** с **резултати)**.



След като имаме X и Y координатите, сме готови да **създадем инстанция** на **играча**. Този път, когато създаваме инстанцията, ще имаме **два параметъра**. Те идват от **конструктора** на **класа Player**, който създадохме **по-рано**.

A picture containing application

Description automatically generated,

Това, което трябва да направим, е да **подадем предишните две променливи**, които създадохме.



Това е всичко, от което се нуждаем, за да **създадем инстанция** на нашия **играч**. Последното нещо, преди да разработим **цикъла на играта**, е да създадем List<Rock>, в който ще **добавяме новите камъни**, преди да **паднат** и да се опитат да **убият играча**.



Сега сме **готови** да продължим с **цикъла на играта**.

### Инициализация на цикъла на играта

Сега имаме **всичко подготвено** за **цикъла** на **играта**. Играта започва с цикъла и **свършва**, когато **потребителят загуби**.

Първо, отворете един **while-цикъл**.

Graphical user interface, application

Description automatically generated

Първият **метод**, от който ще се нуждаем в цикъла, е RedrawConsole(). Този метод само ще **пренарисува конзолата**. Засега **закоментирайте метода**, докато дойде време да го **имплементираме**.



Следващата ни задача е да **създадем камъните**. Ще използваме **метода** CreateRocks(), но засега ще го **закоментираме**.



Сега трябва да **нарисуваме играча**. За тази цел ще използваме метода Draw() **метода**, който създадохме по-рано в **класа** **Player**. Ще достъпим метода от **инстанцията** на **класа** **Player**, която създадохме преди.



След това трябва да **нарисуваме камъните**. За да го направим, ще използваме **метода** DrawRocks(). Нека засега да го **закоментираме**:



Сега **играчът** и **камъните** са подготвени. Следващата задача е да **започнем** да ги **движим**. Ще **местим** **играча**, като извикаме неговия Move() **метод**, а **камъните** ще **местим** с **метода** MoveRocks().

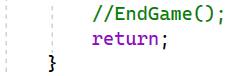


Последното нещо, което трябва да направим, е да **проверим** дали играчът е **ударен от камък**. Ще го направим, като използваме **свойството** HasBeenHit.

A picture containing logo

Description automatically generated

Ако играчът е бил **ударен**, ще използваме метода EndGame(), който ще изпише "**Game Over**" на конзолата и ще **прекрати цикъла**.



Накрая ще използваме **метода** Thread.Sleep(), който **спира цикъла** за даден **брой милисекунди**. Милисекундите ще бъдат **определени** от **потребителя** и ще се съхраняват в **променливата** rockFallSpeed.



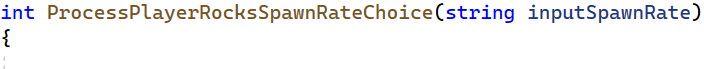
Това е всичко, което ни трябва за **цикъла** на **играта**. Нека сега да започнем **имплементацията** на **методите**.

### Имплементация на методите

#### Имплементация на метода ProcessPlayerRocksSpawnRateChoice(string inputSpawnRate)

Този **метод** ще обработвапотребителския **вход** за **скоростта** на **генериране** на **камъни**. Трудността ще бъде от **едно** (1) до **десет** (10), ще имаме и **предпочитана опция**, която ще бъде **четири** (4). Когато потребителският вход е **обработен**, методът ще върне **стойността**.

Създайте метод, който **приема скоростта**, избрана от потребителя.



Следващата стъпка е да **валидираме** потребителския вход. Обмислете начин да го направите **сами**.

**Насоки за валидация:**

* Входът трябва да бъде **цяло число**
* Числото трябва да бъде по скалата от **1** до **10**
  + Не по-малко от **едно**
  + Не повече от **десет**
  + Ако числото е **по-голямо от 10** или **по-малко от 1**, направете го **равно на предпочитаната** опция, т.е. **4**

**Върнете резултата**:

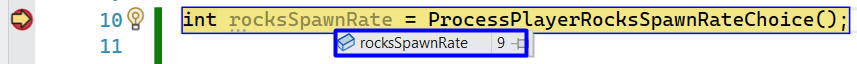
Graphical user interface, application, Teams

Description automatically generated

Когато сте **готови** с **метода**, се върнете в **цикъла** на **играта**, **откоментирайте** го и го **тествайте**, за да проверите дали **работи правилно**.

Graphical user interface, text

Description automatically generated

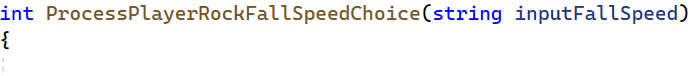


Нека да **продължим със следващия**.

#### Имплементация на метода ProcessPlayerRockFallSpeedChoice()

Този метод ще обработва потребителския **вход** за **скоростта** на **падане** на **камъка**. **Трудността** ще бъде от **петдесет** (50) до **сто и петдесет** (150) и ще имаме **предпочитана опция** от **сто и двадесет** (120). Когато

Създайте **метод**, който приема **входа** за **скоростта** на **падане**.



Сега обмислете как да я **валидирате**.

**Насоки за валидация:**

* Входът трябва да бъде **цяло число**
* Числото трябва да бъде по скалата от **50** до **150**
  + Не по-малко от **50**
  + Не повече от **150**
* Ако числото е **по-голямо от 150** или **по-малко от 50**, направете го **равно на предпочитаната** опция, т.е. **120**

Graphical user interface, application, Teams

Description automatically generated

Преди да **върнем скоростта** на **падане**, трябва да направим **още едно изчисление**. Тъй като работим с **милисекунди**, ще възникне **грешка**. Ние искаме **50** да е **най-бавната** скорост, а **150** да е **най-бързата**. Само че ако въведем **50**, играта ще върви **по-бързо** (защото 50 милисекунди са по-малко от 150). За да **разрешим проблема**, можем да **обърнем логиката**.

**Насока** – **Общата скорост** на падане на камъка е **двеста** (200).

Опитайте се да разрешите проблема сами и да **върнете стойността**.

Graphical user interface

Description automatically generated

Когато сте **готови с метода**, се върнете в **цикъла** наиграта, **откоментирайте** го и го **тествайте**.

Text

Description automatically generated



Нека да продължим със **следващия метод**.

#### Имплементация на метода SetWindowProperties()

Този метод ще **задава свойствата**, които ни трябват за **конзолата**. **Ширината** и **височината** на конзолата, както и **видимостта** на **курсора**, ще бъдат зададени на **false**. Методът ще прави настройки **преди играта** да **започне**.

Създайте **метода**, **изчистете конзолата** и задайте **видимостта** на **курсора** на **false**.

Text

Description automatically generated

Сега ще използваме **два вградени метода**, които ще ни помогнатда зададем **ширината** и **височината** на конзолния прозорец. Първо ще използваме метода Console.SetWindowSize(). Като стойности ще **подадем** константите WindowsHeight у WindowsWidth, които създадохме по-рано.



**Повече** за метода: <https://learn.microsoft.com/en-us/dotnet/api/system.console.setwindowsize?view=net-6.0>.

Вторият метод е Console.SetBufferSize(). Той ще задава **височината** и **ширината** на екранния **буфер** на конкретни стойности. **Екранният буфер** е **двумерен масив** с данни за **символи** и **цветове** за **изхода** на конзолата. Ще подадем **същите** стойности като в **предишния метод**.



**Повече** за метода: <https://learn.microsoft.com/en-us/dotnet/api/system.console.setbuffersize?view=net-6.0>.

**Повече** за екранния буфер: <https://learn.microsoft.com/en-us/windows/console/console-screen-buffers>.

Това е всичко за този метод. Върнете се в **цикъла** на **играта**, **откоментирайте** го и го **тествайте**.

Ако всичко работи правилно, **конзолният прозорец** трябва да се **събере** до **подадените стойности** веднага след като ги **въведете**.

Text

Description automatically generated 🡪Text

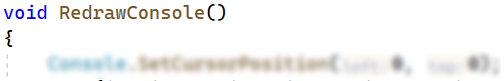
Description automatically generated

Нека да продължим със **следващия метод**.

#### Имплементация на метода RedrawConsole()

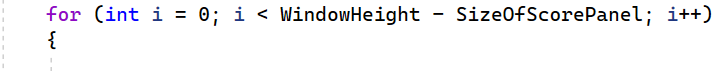
Този метод ще **прерисува** конзолата **без да я чисти**. Това означава, че ще **започнем** от **горния ляв ъгъл** на конзолния прозорец и ще рисува **празен стринг** до **края** на **конзолата**. Това ще ни помогне да **избегнем мигането** на **конзолата**, което е причинено от **метода** Console.Clear().

Създайте метод, който **позиционира курсора** на координати **(0, 0)**, т.е. **горния ляв ъгъл** на конзолата.



Следващата стъпка е да създадем **for-цикъл**, който ще итерира, докато **i е по-малко** от **височината на прозореца минус размера** на **панела** с **резултати**.

Ще извадим **размера** на **панела** с **резултати** от **височината** на **прозореца**, защото панелът **винаги** ще бъде на **едно и също място** и ще си остане там **до края на играта**.



В цикъла ще напишем **интервал петдесет пъти**, което е **ширината** на **прозореца** (имаме константа за това):



Последното нещо е да използваме **метода** DrawScorePanel() – методът ще чертае **панела с резултати** на конзолата. Тъй като **още не е имплементиран**, само го **извикайте** и го **закоментирайте**.

A picture containing logo

Description automatically generated

Това е всичко за този метод – можем да го **тестваме**,след като имплементираме **метода** DrawScorePanel(), защото **двата метода работят заедно**. Нека да продължим с него.

#### Имплементация на метода DrawScorePanel()

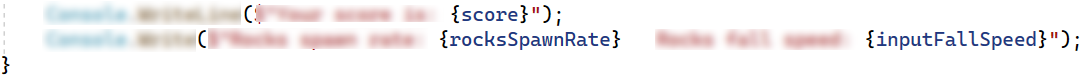
Методът ще **разделя бойното поле** (горната част, откъдето падат камъните) със символа "**=**" и ще визуализира на конзолата **резултата** на потребителя, **скоростта** на **падане** на камъните, както и **скоростта** на **генериране** на **камъни**, избрана от потребителя.

Създайте **метода** и измислете начин да нарисувате символа "**=**" през **цялата ширина на прозореца**.

Graphical user interface, text, application, chat or text message

Description automatically generated

Остава да **визуализираме резултата** на потребителя, **скоростта** на **падане** на камъните и **скоростта** на **генериране** на **камъни**. За визуализация на **скоростта** на **падане** ще използваме стойността, избрана от **потребителя**, иначе ще покажем **променената стойност** на метода ProcessPlayerRocksFallSpeedChoice(string inputFallSpeed). Можете да направите това сами – имате **необходимите променливи**, остава да измислите **съобщение**, което да визуализирате.



Това е **всичко за този метод**. Върнете се в **метода** RedrawConsole() и го **откоментирайте**, след това се **върнете** в **цикъла** на **играта** и **откоментирайте метода** RedrawConsole(), за да може да го **тествате**.

A screenshot of a computer

Description automatically generated with medium confidence

Ако всичко **работи правилно**, трябва конзолата да е разделена на **две части**, като долната част съдържа **данните** за **играта**. Нека да продължим със следващия метод.

#### Имплементация на метода CreateRocks()

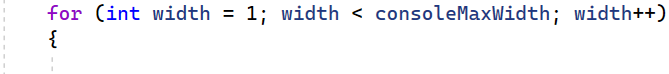
Този метод **създава камъни** в зависимост от **скоростта** на **генериране**, която потребителят е избрал, и ги добавя към **списъка с камъни**, който създадохме по-рано.

Създайте **метод** и **променлива**, които ще съдържат **максималната ширина** на конзолата **минус 1**. Ако не извадим 1, ще излезем **извън границите** на прозореца и програмата ще **спре**.

A picture containing logo

Description automatically generated

След това **създайте for-цикъл**, който ще итерира, докато **i** е по-малко от **максималната ширина** на конзолата. Това ще ни помогне да създадем камък и да **зададем** неговата **X координата** на **текущата ширина** (**i**).



Преди да **създадем камък**, трябва да проверим дали **е време да го създадем**. За тази цел ще използваме метода ShouldGenerateRock(). Тъй като **още не е имплементиран**, само го извикайте и го **закоментирайте**.

A picture containing logo

Description automatically generated

Ако е време, просто **създайте нов камък** с **текущата ширина** като параметър и го добавете към **списъка с камъни**.

Graphical user interface, text, application

Description automatically generated

Това е **всичко за този метод**. Можете да го тествате, след като имплементирате метода ShouldGenerateRock(), защото **двата метода** работят **заедно**. Нека да продължим с него.

#### Имплементация на метода ShouldGenerateRock()

Методът ще **върне true** или **false** в зависимост от случайно число между **0** и **101** и от **скоростта** на **генериране** на камъни, избрана от играча.

Създайте **метода** и **нова случайна променлива**.

Graphical user interface, application

Description automatically generated

Сега се нуждаем от **променлива** за **максималната скорост** на **генериране**, равна на 100.



След това трябва да проверим дали **случайното число** е **по-голямо** или **равно** на променливата spownRateMaxValue **минус скоростта на генериране**.



Това е **всичко за тоз метод**. Върнете се в **метода** CreateRocks() и го **откоментирайте**, след това се върнете в **цикъла** на **играта** и **откоментирайте** **метода** CreateRocks(). При **тестване** приложението трябва да работи така:

A screenshot of a computer

Description automatically generated with medium confidence

Можем да видим, че конзолата е **същата** като преди, но ако добавим **breakpoint** и **дебъгнем** приложението, ще видим, че всяка итерация добавя **повече** и **повече камъни** към списъка.

Table

Description automatically generated with low confidence 🡪 Table

Description automatically generated

Нека да продължим със **следващия метод**.

#### Имплементация на метода DrawRocks()

Този метод ще **визуализира камъните** на конзолата. Ще използваме нашия **списък** с **камъни**, в който добавяме камъни в предишния метод. Ще зададем **позицията** на **курсора** на текущите **X** и **Y координати** и ще визуализираме **символа** на **камъка** на конзолата.

Създайте **метода** и отворете foreach**-цикъл**, който итерира през **списъка** с **камъни**.

A picture containing graphical user interface

Description automatically generated

Накрая задайте **позицията** на **курсора** на текущите **X** и **Y координати** (достъпвайки свойствата на камъка) и **нарисувайте символа** на **камъка** на конзолата.

Graphical user interface

Description automatically generated with low confidence

Това е **всичко за този метод**. Сега се върнете в **цикъла** на **играта** и го **откоментирайте**. Също **откоментирайте предишния** и **следващия ред**, където рисуваме и движим играча, защото вече имаме **всички необходими методи**.

Text

Description automatically generated

Нека да **стартираме приложението** и да го **тестваме**:

A picture containing graphical user interface

Description automatically generated A picture containing graphical user interface

Description automatically generated

Можем да видим, че **камъните** започват да се **появяват на конзолата**, **играчът** също е означен и можем да го **движим**.

Следващият **метод**, който ще имплементираме**,** ще **движи камъните**. Нека да продължим с него.

#### Имплементация на метода MoveRocks()

Това е **най-комплексният метод** в **цялото приложение**. Той ще **движи камъни**, ще **премахва някои**, когато тяхната **Y координата** достигне **максималната височина** на конзолата, и ще проверява дали **има сблъсък** между **играча** и някой **камък**.

Създайте **метода** и **списък от камъни**, който ще съдържа **камъни** за **премахване**.

A picture containing logo

Description automatically generated

След това отворете foreach-цикъл, който ще итерира през списъка с камъни (списъка, в който добавяме новосъздадените камъни).

A picture containing logo

Description automatically generated

Първото нещо, което трябва да направим в цикъла, е да **добавим едно** (1) към **Y координатата** на текущия камък. Това ще **премести камъка** един ред **по-надолу** на конзолата.



Следващата стъпка е да **проверим** дали **Y координатата** на **текущия камък** е достигнала **максималната конзолна височина**. Ако я е достигнала, това означава, че **сме избегнали камъка**, така че можем да го **добавим** към **списъка** с **камъни** за **премахване** и можем да **добавим едно** (1) към **резултата** на потребителя.

Graphical user interface, text, application, chat or text message

Description automatically generated

Сега трябва да **проверим** дали има **сблъсък** между **камъка** и **играча**. Тук ще използваме **метода** ThereIsCollision(rock, player). Тъй като той **още не е имплементиран**, само го **извикайте** и го **закоментирайте**. Ако **има сблъсък**, трябва да направим **свойството** HasBeenHit равно на **true** (по този начин знаем, че трябва да **прекратим цикъла** на **играта**).

Graphical user interface, text

Description automatically generated

Накрая трябва само да **премахнем камъните** от **списъка**. Тук ще използваме **метода** RemoveRocks(rocksToRemove, rocks). Тъй като **още не е имплементиран**, само го **извикайте** и го **закоментирайте**.

Logo

Description automatically generated with medium confidence

Това е **всичко за този метод**. Можем да го тестваме **след имплементацията** на **методите** ThereIsCollision(rock, player) и RemoveRocks(rocksToRemove, rocks), защото те **работят заедно**. Нека да **продължим**.

#### Имплементация на метода ThereIsCollision(Rock rock, Player player)

Методът ще **получи играч** и **текущия камък** и ще провери дали **има** **сблъсък** между тях (т.е. да са на **една и съща точка** от конзолата). Методът връща **true** или **false** в зависимост от проверката.

**Създайте метода**:

A picture containing logo

Description automatically generated

Сега трябва да **проверим** дали **камъкът** и **играчът** са с **една и съща ширина** и **височина**. Ще използваме **методите** RockAndPlayerAreOnSameWidth(rock, player) и RockAndPlayerAreOnSameHeight(rock, player). Тъй като **още не са имплементирани**, само ги **извикайте** и ги **закоментирайте**. Ако **камъкът** и **играчът** имат **еднаква ширина** и **височина**, върнете **true**. В противен случай върнете **false**.

Graphical user interface, text, application

Description automatically generated

Това е **всичко за този метод**. Нека да имплементираме **останалите методи**, за да може този да **работи правилно**.

#### Имплементация на метода RockAndPlayerAreOnSameWidth(Rock rock, Player player)

Този **метод** ще **приема играча** и **текущия камък** и ще **проверява** дали **някой** от **символите** на **играча** се **сблъсква** с **камъка** (в случая **играчът** ни е отбелязан с **три символа**  "(0)"). Методът ще **върне true или false** в зависимост от проверката. Ще **създадем метода**, като използваме **ламбда изрази**.

Създайте **метода** и **проверете** дали **X координатата** на **играча** е **равна** на **X координатата** на **текущия камък**. Искаме да направим това за **всеки символ** от **аватара** на играча. Опитайте се да го **напишете сами**.



Това е **всичко за този метод**. Нека да **продължим**.

#### Имплементация на метода RockAndPlayerAreOnSameHeight(Rock rock, Player player)

Този **метод** ще **приема играча** и **текущия камък** и ще **проверява** дали **Y координатата** на **камъка** е **равна** на **Y координатата плюс едно** (1) на **играча**. Ще връща **true** или **false** в зависимост от **проверката**. Отново ще използваме **ламбда изрази**. Опитайте се да **създадете метода сами**.



**Методите** са **готови**. Върнете се в **метода** ThereIsCollision() и ги **откоментирайте**. След това се върнете в **метода** MoveRocks()и **откоментирайте метода** ThereIsCollision().

 🡪



Все още **не можем** да **тестваме** методите, защото трябва да **създадем** още един - RemoveRocks(rocksToRemove, rocks). Нека да го направим.

#### Имплементация на метода RemoveRocks(List<Rock> rocksToRemove, List<Rock> rocks)

Този **метод** ще **приема списък** с **камъни** и **списък** с **камъни**, които трябва да се **премахват**. Ще итерира през **списъка** с **камъни** за **премахване** и ще ги маха от **списъка** с **всички камъни**.

Създайте **метода** и отворете foreach**-цикъл**, който **итерира** през **списъка** с **камъни** за **премахване**.

Graphical user interface, text

Description automatically generated with medium confidence

В цикъла **премахнете камъка** от списъка с **всички камъни**.

A picture containing text

Description automatically generated

Това е **всичко за този метод**. Сега се върнете в **метода** MoveRocks() и **откоментирайте** този.



**Преди да тестваме методите**, се върнете в **цикъла** на **играта** и **откоментирайте** метода MoveRocks(), както и **последния ред** от **цикъла**, където **спираме метода** за даден **интервал** от **време**.

A picture containing logo

Description automatically generated

**Изпълнете приложението** и проверете дали **работи правилно**.

Background pattern

Description automatically generated with medium confidence

Можем да видим, че имаме **напълно функционираща игра**. Можем да **движим играча**, **камъните падат** и **резултатът** се **актуализира** след **всеки паднал камък**. Има **още едно нещо**, което трябва да направим. Трябва да **спрем играта**, когато играчът е **ударен** от **камък**. Нека да имплементираме **финалния** **метод** - EndGame().

#### Имплементация на метода EndGame()

Този **метод** ще задава **позицията** на **курсора** на **определени координати** и ще **изписва** "**GAME OVER!!**".

**Създайте метода** и **променлива** от тип **int**, която ще съдържа **конкретна координата**. Тя ще бъде **тридесет и три** (33). Това е **последният ред** в **панела** за **резултати**.

A picture containing text

Description automatically generated

Сега задайте **позицията** на **курсора** на **нула** (0) и като **втори параметър** подайте **новосъздадената променлива**. След това **напишете** на конзолата"**GAME OVER!!**".

Text

Description automatically generated with medium confidence

Това е **всичко за последния метод**. Върнете се в **цикъла** на **играта** и **откоментирайте** последната част.



Нека отново да **тестваме** **играта**.

Background pattern

Description automatically generated with medium confidence Background pattern

Description automatically generated

Можем да видим, че когато **играчът** е **ударен** от **камък**, играта **спира** и на конзолата се изписва текстът"**GAME OVER!!!**". Това е **всичко**, което ще има в **нашата игра**.

## Качване на проекта в GitHub

Качете промените в проекта в **GitHub**, както вие прецените – чрез **TortoiseGit**, чрез **Git Bash** или от **Visual Studio.**

Graphical user interface, text, application, email

Description automatically generated

## \* Модификации на кода

Сега можете да добавите и **други функционалности** към кода.

|  |  |
| --- | --- |
| Icon  Description automatically generated | Това е ваш собствен проект. **Бъдете уникални**.   * Имплементирайте **свои собствени функционалности**. * Имплементирайте кода **сами**, добавете ваш стил, форматиране, коментари и т.н. |

Ето **няколко идеи** за това какви други функционалности можете да добавите.

### Добавяне на Unicode символи

Потърсете за **по-интересни символи** за **играча** и за **камъните**.

**Таблица с Unicode символи** – <https://www.rapidtables.com/code/text/unicode-characters.html>.

### Добавете multiplayer

Обмислете начин за **добавяне** на **още един играч**, за да може **играта** да се играе от **двама потребители**.

### Обръщане на играта

Още една идея е да **обърнете играта** – играчът да е на **първия ред** (в горната част на конзолата), а **камъните** да идват **от долната част** на конзолата.

### Създаване на GUI приложение

Може да създадете **GUI приложение** за играта.

### Допълнителни идеи

* Създайте **импровизирана база данни**, която да съхранява **името на потребителя** и разултата, и ги **визуализирайте на конзолата**.
  + Използвайки JSON – <https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/how-to?pivots=dotnet-6-0>.
  + Използвайки XML – <https://learn.microsoft.com/en-us/dotnet/standard/serialization/how-to-serialize-an-object>.
* Може да добавите и **свои идеи**

### Commit-ване в GitHub

Сега е време да **качите промените в GitHub**!

|  |  |
| --- | --- |
| Icon  Description automatically generated | Много е важно да **commit-вате често** вашия код в GitHub. По този начин създавате **богата commit история** за вашия **проект** и **профил**:  A picture containing chart  Description automatically generated |

## Създаване на README.md файл

Силно препоръчително е да напишете **документация на вашия проект** в GitHub, за да опишете какво представлява. Нека да направим **README.md** файл за нашия проект. Можем да го редактираме от GitHub репото.

Graphical user interface, text, application, email

Description automatically generated

### Секции на документацията

Добавете информация за проекта, използвайки **Markdown**: цели на проекта, използвани технологии, скрийншоти, живо демо и т.н. Обикновено файлът съдържа следните **секции**:

* **Име** на проекта (трябва да отговаря на въпроса "Какво има в проекта?")
* **Цели** на проекта (какъв проблем решаваме, напр. определена игра)
* **Решение** (описва как решаваме проблема 🡪 алгоритми, технологии, библиотеки, технологични рамки и т.н.)
* Линк към **source code**
* **Скрийншоти** (скрийншоти от проекта в различни ситуации)
* **Живо демо** (демо, което може да се достъпи и тества с един клик)

### Цели на проекта

Започнете документацията, като опишете **целите на проекта**. Какъв проблем решава?

### Примерна документация

Това е пример как може да изглежда **документацията** на проекта. Не ви съветваме да копирате директно:

Graphical user interface, text, application, chat or text message

Description automatically generated

|  |  |
| --- | --- |
| Icon  Description automatically generated | **Напишете документацията сами!** Не копирайте примерната такава!  Това е **вашият уникален GitHub профил** и вашият проект. **Бъдете различни** от останалите. |

Намерете подходящо **изображение** и го добавете. Можете да го направите по следния начин:



### Вход

Добавете информация за **входа**, който **потребителят** трябва да **въведе**:

Graphical user interface, text, application

Description automatically generated

### Контроли

Добавете информация за **контролите на играта**:

Graphical user interface, text, application

Description automatically generated

### Вашето решение

Опишете как сте **решили проблема**: алгоритми, технологии, библиотеки, технологични рамки и др.

### Линк към source code

Добавете **линк** към вашия **source код**.



### Скрийншоти

Добавете **скрийншоти** на проекта:

1. **Направете скрийншот** с предназначен за това инструмент (например [Snipping Tool](https://support.microsoft.com/en-us/windows/open-snipping-tool-and-take-a-screenshot-a35ac9ff-4a58-24c9-3253-f12bac9f9d44) за Windows).
2. **Поставете** скрийншота във файла за документация, използвайки [Ctrl+V]:
3. Примерни скрийншоти за играта "Падащи камъни":

A picture containing timeline

Description automatically generated

## Качване на приложението си в Replit

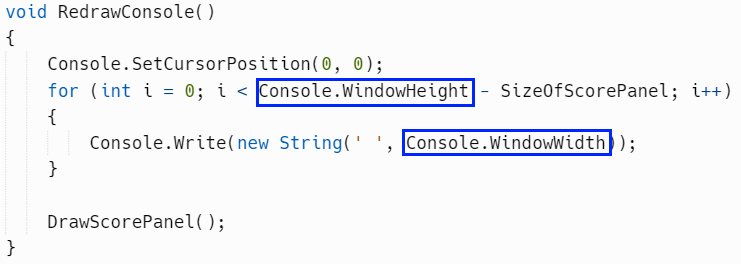
Нека да добавим **проекта** в **Replit**, за да можем да го **споделяме с приятели** и да го добавим в нашия **GitHub** профил. Вече знаете как да направите това.

Имаме **две опции** за **избор** – първата е да използваме **Mono C#**. Той още **не е актуализиран** и работи с **.NET 5.0**. Това означава, че ще трябва да **преправяме кода** на .NET 5.0, за да работи с **Mono C#**. Втората опция е да създадем **C# replit project**. Той работи с **.NET 6.0** и можем **директно да копираме** файловете с **кода**. Единствената разлика е, че **Mono C#** работи малко **по-бързо** от C# replit project.

**Replit не може** да работи с **Console.SetWindowSize()** и **Console.SetBufferSize()**, така че ще трябва да ги **закоментираме** (просто няма да ги използваме. Също навсякъде, където използваме константите **WindowHeight** и **WindowWidth**, трябва да ги **заместим** с **Console.WindowWidth** и **Console.WindowHeight**. Това е защото **Replit конзолата** е зададена с конкретна **ширина** и **височина**, които **не могат да се променят**.

Graphical user interface, text

Description automatically generated

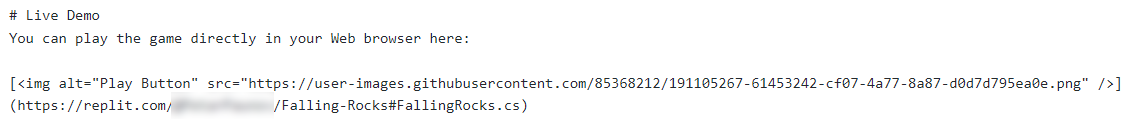


Graphical user interface, application, Word

Description automatically generated

## Добавяне на Replit линк към README.md

Сега може да добавите **демо**, което да се тества с **един клик**, към вашето **GitHub README**:



Може да направите **скрийншот** от **Replit.com** и да го добавите в README файла с [**Ctrl+V**]:

Graphical user interface, text, application

Description automatically generated

Поздравления! Успешно завършихте **конзолната игра "Падащи камъни"** и имате още един проект в **GitHub портфолиото ви**.