Доклад на

Тема:

MVC архитектура

От: Симеон Марков

Клас: 11А

1. **Какво e MVC(Model-View-Controller) ?**

* Представлява архитектурен шаблон, който най-често се използва при създаването на потребителски интерфейс. Той „разделя“ приложението на **три взаимосвързани части**. Това е направено с цел да се раздели вътрешното представяне на информация от начините по които информацията се представя и приема от потребителя. MVC шаблонът разделя тези главни компоненти, което позволява на разработчиците да използват отново вече написан код по-ефективно, а също така позволява и паралелна разработка.
* Традиционно MVC шаблонът се използва за графични потребителски интерфейси (Graphical User Interface – GUI), но вече той се използва както и за уеб приложения, така и за мобилни такива, също и за други типове клиенти.

1. **Компоненти на MVC шаблона**

* **Модел(Model)** - централен компонент в шаблона. Това динамичната структура от данни на приложението, независима от потребителския интерфейс. Моделът **управлява данните, логиката и правилата** на приложението.
* **Изглед(View)** - е изходящият поток от информация (това, което приложението **изпраща като отговор до дисплея**, респективно – до потребителя, в следствие на неговата заявка). Възможни са няколко различни изгледа на една и съща информация.
* **Контролер(Controller)** - **приема потребителския вход** (т.е. данните, които потребителя въвежда, неговите заявки и т.н.) и ги преобразува в команди към модела или изгледа.

1. **Историята на MVC модела**

* Тригве Реенскауг създава MVC, докато работи върху Smalltalk-79 като гостуващ учен в изследователския център Xerox Palo Alto (PARC) в края на 70-те години. Той иска модел, който може да се използва за структуриране на всяка програма, в която потребителите взаимодействат с голям, сложен набор от данни. Дизайнът му първоначално има четири части: Модел, Изглед, Нещо и Редактор(Model-View-Thing-Editor). След като го обсъдиха с другите разработчици на Smalltalk, той и останалата част от групата се спряха на Model, View и Controller.
* В окончателния си дизайн Моделът представлява някаква част от програмата чисто и интуитивно. **Изгледът е визуално представяне** на **Модел, извличащ данни** от Модела, за да се покаже на потребителя и предаващ заявки напред-назад между потребителя и Модела. **Контролерът е организационна част от потребителския интерфейс**, която оформя и координира множество изгледи на екрана и която получава потребителски вход и изпраща съответните съобщения до основните си изгледи. Този дизайн включва и редактор като специализиран вид контролер, използван за модифициране на конкретен изглед и който се създава чрез този изглед.
* MVC моделът впоследствие се развива, пораждайки варианти като йерархичен модел-изглед-контролер (**HMVC**), модел-изглед-адаптер (**MVA**), модел-изглед-презентатор (**MVP**), модел-изглед-изглед (**MVVM**) и други, които адаптират MVC към различни контексти.

1. **Кои езици и фреймуърци използват MVC ?**

* Програмни езици, които използват модела -  C, C++, C#, Java, Ruby, Smalltalk и много други.
* Фреймуърци, които използват модела -  Angular, Express, Django, Flask, Laravel, Ruby on rails и други.

1. **Предимства на MVC**

* Тъй като MVC разделя основните компоненти на приложението, това позволява на разработчиците да работят паралелно по различни компоненти, без да оказват влияние или да си пречат един на друг.
* Създавайки компоненти, които са независими едни от други, разработчиците могат да използват многократно един компонент в различни приложения.
* Организиране на мащабни web приложения
* Поддържа **AMI(Asynchronous Method Invocation)**
* Лено променяем
* По-бърз процес на разработване
* Лено планиране и поддържане
* MVC модела връща данните без да ги форматира
* Поддържа **TTD(test-driven development)**
* Възможност за съдаване на няколко изгледа

1. **Пример за използване на MVC модел**

* Структура:

A screenshot of a computer program

Description automatically generated

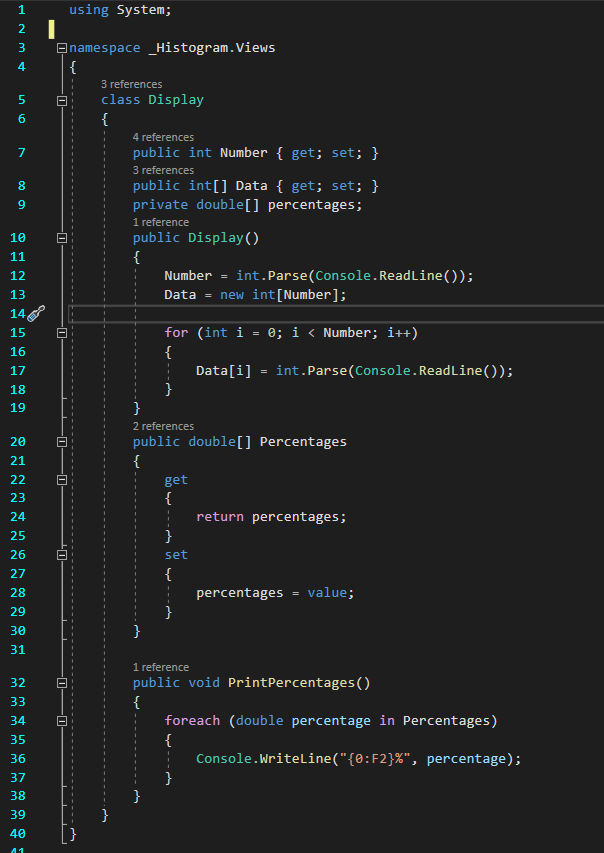
* A screen shot of a computer program

  Description automatically generatedController:
* Model:

A computer screen shot of a program

Description automatically generated

* Views:



* A screen shot of a computer

  Description automatically generatedProgram.cs