## PITANJA ZA USMENI

**1. Šta je cilj definisanja klase?**

Класс – определяемый пользователем тип, сосотоящий из полей данных (переменные) и члены, которые работают с этими данными (методы, свойства).

Цель определения класса заключается в том, что за счёт группирования данных и связанной с ними функциональности в определённый класс мы получаем возможность моделировать свою программу в соответствии с объектами как в реальном мире.

**2. Šta znate o konstruktoru?**

Конструктор – это специальный метод класса, который неявно вызывается при создании объекта с применением ключевого слова new. В отличии от остальных методов класса, конструктор не имеет возвращаемое значение (даже void) и носит имя идентичное имени класса.

public string Name { get; set; }  
public Student(string name)  
{  
 Name = name;  
}

При определении класса без своего конструктора, то определяется дефолтный, который не носит никаких параметров. При желании мы можем его переопределить. Также мы можем воспользоваться перегрузкой методов (preopterećenje metoda), чтобы создать несколько вариантов определения класса.

public class MyNumber  
{  
 public int Number { get; set; }  
  
 public MyNumber(string number)  
 {  
 Number = int.Parse(number);  
 }  
 public MyNumber(int number)  
 {  
 Number = number;  
 }  
}

**3. Kakvo nasleđivanje podržava C#?**

C# и .NET поддерживает только Jednostruko nasleđivanje. Это означает, что каждый класс может наследовать члены только от одного класса. Это помогает нам выполнять иерархию наследования.

public class A { }  
public class B : A { }  
public class C : B { }  
public class D : C { }

**4. Koje nivoe pristupa možemo definisati atributu klase?**

Мы имеем несколько модификаторов доступа для членов класса:

* public – элементы не имеют никаких ограничений. Открытый член может быть доступен из объекта и из любого другого произвольного класса. Также доступен для внешних сборок
* private – Элементы доступны только классу, где они определены
* protected – элементы доступны классу, в котором они определены и для любого другого класса, который его наследует.
* internal – элементы доступны только для текущей сборки .NET.
* protected internal – так же, как и protected, но только для текущей сборки .NET

**5. Koji elementi klase se ne nasleđuju?**

* Статические конструкторы, которые инициализируют статические данные в классе
* Конструкторы, которые вызываются для создания нового экземпляра класса
* Члены класса с модификатором доступа private

**6. Navedite dva razloga zbog kojih bi trebalo da se predefiniše metod koji je već implementiran u nadklasi!**

* Чтобы не копировать тот же код метода, что и в родительском классе
* Мы можем дополнить функциональность метода в родительском классе и это изменение должно быть затронуто во всех дочерних объектах

**7. Koje metode se ne mogu predefinisati u podklasi?**

static, private, конструкторы, методы, которые не имеют virtual, astract или overrride модификатор.

**8. Koje metode se moraju predefinisati u podklasi?**

Только абстрактные методы у родительских для дочерних классов. (не интерфейс)

**9. Koja je ključna reč za kreiranje klase koja se ne može koristiti u procesu nasleđivanja?**

sealed

**10. Objasnite pojam polimorfizma!**

Полиморфизм преоставляет подклассу способ определения собственной версии метода, определённого в родительском классе, с применением процесса, называемым переопределение метода. В C# языке для этого мы используем ключевые слова override (для подклассов) и virtual (для базового класса)

**11. Objasnite pojam apstraktne klase!**

Абстрактный класс похож на обычный класс. Он также может иметь переменные, методы, конструкторы, свойства. Единственное, что при определении абстрактных классов используется ключевое слово abstract.

Но главное отличие абстрактных классов от обычных состоит в том, что мы НЕ можем использовать конструктор абстрактного класса для создания экземпляра класса.

Тем не менее абстрактные классы полезны для описания некоторого общего функционала, который могут наследовать и использовать производные классы

12. Koje prednosti ima korišćenje interface-a?

Интерфейс представляет собой лишь именованный набор абстрактных членов и не имеют своей реализации как делает абстрактный класс. Класс может реализовывать (не наследовать) множества интерфейсов и посредством этого поддерживать множество линий поведения.

После реализации интерфейса в классе, мы можем взаимодествовать с объектом этого класса посредством интерфейса.

13. Šta je izuzetak?

Исключения – это дефекты, некорректный пользовательский ввод и ошибки в процессе выполнения программы.

В языке C# все исключения наследуются от класса System.Exception, где мы можем получить сообщение об ошибки, источник, номер линии в коде и т.д.

Также C# позволяет перехватывать ошибки при помощи try { ... } catch (ExceptionClass ex) { ... }.

14. Objasnite prednost korišćenja stilova u XAML fajlu!

XAML (Extensible Application Markup Language) описывает пользовательский интерфейс в программе для последующей работы с ним в коде (наподобия HTML). XAML не ограничивается приложениями .NET и может быть воспользована в других языках.

Помимо этого, xaml делает настройку визуализации элементов управления легче, используя стили и анимации.

15. Šta znate o Grid kontejneru?

Распологает содержимое внутри серии ячеек, поддерживаемых внутри табличной сетки.

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

Размер ячеек мы можем определять с помощью RowDefinition и ColumnDefinition свойствами height, width. Мы можем устанавливать собственные размеры (100, 250) или относительные (1\*, 3\*).

Чтобы присоединить элемент к ячейки, мы используем свойства Grid.Row и Grid.Column, где пишем число ячейки по порядку.

16. Šta znate o StackPanel kontejneru?

Укладываем содержимое вертикально или горизонтально, в зависимости от свойства Orientation. При изменении размера StackPanel, содержимое не переносится, а растягивается внутри StackPanel’а