

## Section 16: Generics

14 July 2024 17:30



Harsha Vardhan

- › UI Expert
- › .NET Expert
- › Module Lead
- › Corporate Instructor



## Generics

### Agenda

- 1 **Generic Classes**  
What is generic classes and its features.
- 2 **Generic Constraints**  
Setting-up type constraints on generic constraints.
- 3 **Generic Methods**  
Creating methods with generic-type parameters.



**What**

- > Generic class is a class, which contains one or more "type parameters".

**How****Generic Class - Example**

```
class ClassName<T>
{
    public T FieldName;
```



The generic class is a class where the field's data type can be changed w.r.t. various objects.

**What**

- > Generic class is a class, which contains one or more "type parameters".
- > You must pass any data type (standard data type / structure / class), while creating object for the generic class.

**How****Generic Class - Example**

```
class ClassName<T>
{
    public T FieldName;
```

**Object of Generic Class - Example**

```
ClassName<int> referenceVariable = new ClassName<int>();
```

T => Generic Type Parameter.

you can use any other letter as generic type parameter.

So, in the real projects, use generic classes wherever we are not sure about the data type of the specific field.

Let me give another example of generic classes.

Assume, there is a class called Student; that has a field called "Rank". So if the student appears for the examination, you wanted to store the rank; that means an "int" value. Assume there are 10 students that appear for examination; so for those 10 objects, you wanted to store the "int" value in the Rank field. But for the absent students, you wanted to store the letter "A"; "A" for "Absent".

So what should be the data type of Rank? If it is "int", how can you store the char value w.r.t. absent students?

In case if you mention "char" data type for the Rank field, how can you store the rank number for the students who appear for examination?

So sometimes it should be "char"; sometimes it should be number, right! So for this case also, you will use generic classes.

## Understanding Generic Classes

## Next: Generic Constraints

### Advantages

- › The same field may belong to different data types, w.r.t. different objects of the same class.
- › You will decide the data type of the field, while creating the object, rather than while creating field in the class.
- › It helps you in code reuse, performance and type-safety.
- › You can create your own generic-classes, generic-methods, generic-interfaces and generic-delegates.
- › You can create generic collection classes.
  - › The .NET framework class library contains many new  collection classes in System.Collections.Generic namespace.

| Harsha



## Understanding Generic Classes

## Next: Generic Constraints



- › The generic type parameter (T) acts as "temporary data type", which represents the actual data type, provided by the user, while creating object.
- › You can have multiple "generic type parameters" in the same class (for use for different fields).
- › Generics are introduced in C# 2.0.



## Agenda

1

### Generic Classes

What is generic classes and its features.

2

### Generic Constraints

Setting-up type constraints on generic classes.

3

### Generic Methods

Creating methods with generic-type parameters.



## Generic Constraints

## Next: Generic Methods

### What

- > where T : class
- > where T : struct
- > where T : ClassName
- > where T : InterfaceName
- > where T : new( )

### Generic Constraints - Example

```
class ClassName<T> where T : class
{
    public T FieldName;
}
```



Generic Constraints are the expectations or restrictions on the Generic Parameters.

## Generic Constraints

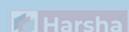
## Next: Generic Methods

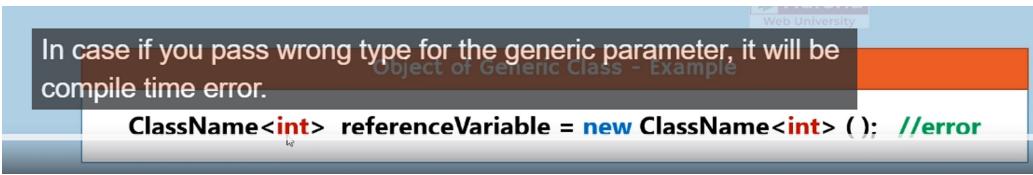
### What

- > Generic Constraints are used to specify the types allowed to be accepted in the "generic type parameter".
- > where T : class
- > where T : struct
- > where T : ClassName
- > where T : InterfaceName
- > where T : new( )

### Generic Constraints - Example

```
class ClassName<T> where T : class
{
    public T FieldName;
}
```

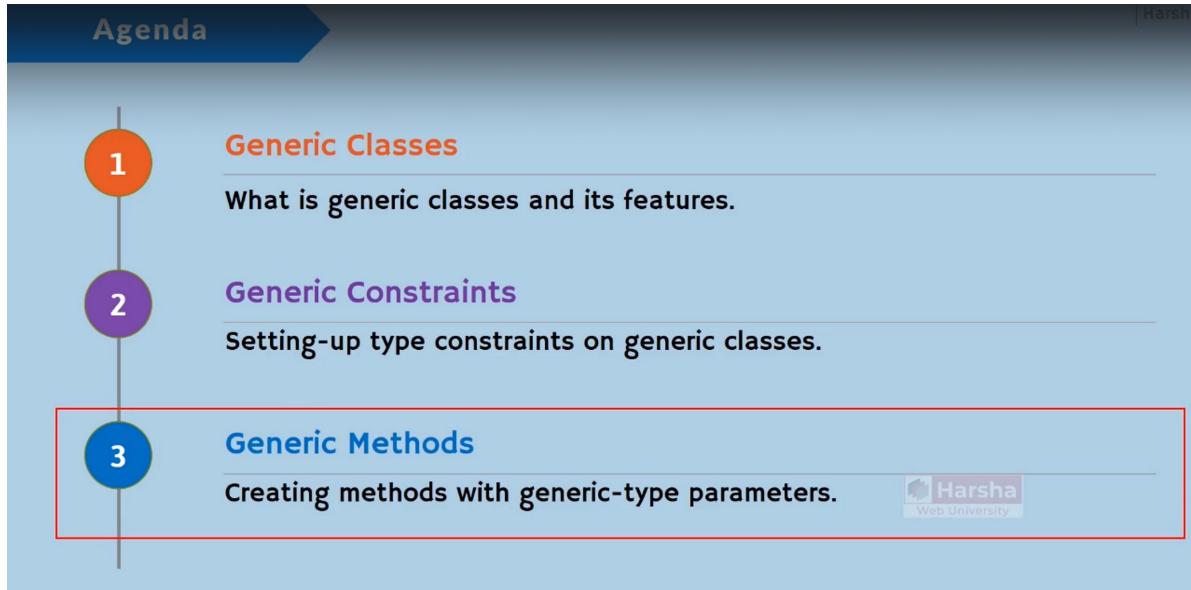




```
public class PostGraduateStudent : Student
{
    public override int Marks { get; set; }
}

//generic class with constraints (want to accept Student or its child classes only)
public class MarksPrinter<T1, T2> where T1 : Student where T2 : Employee
{
    public T stu;

    public void PrintMarks()
    {
        Student temp = (Student)stu;
        System.Console.WriteLine(temp.Marks);
    }
}
```



### What

- › Generic Method is a method that has one or more generic parameter(s).

### Advantage

- › You can restrict what type of data types (class names) allowed to be passed while creating object.

#### Generic Method - Example

```
public void MethodName<T>
{
}
```

#### Calling Generic Method - Example



```
MethodName<datatype>( valueHere );
```