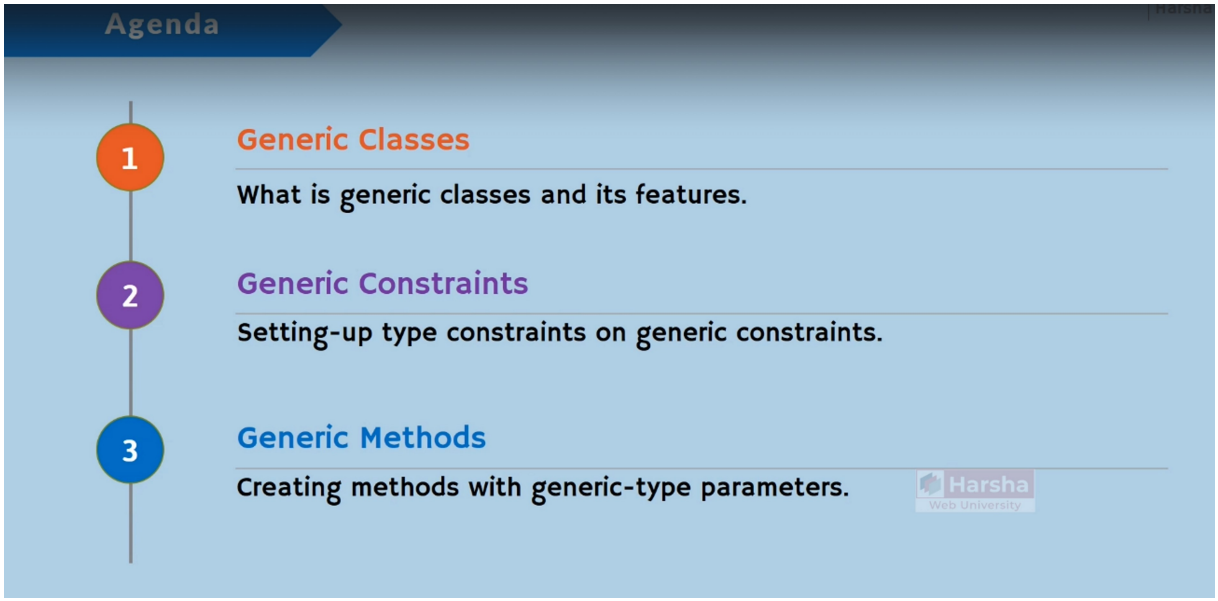
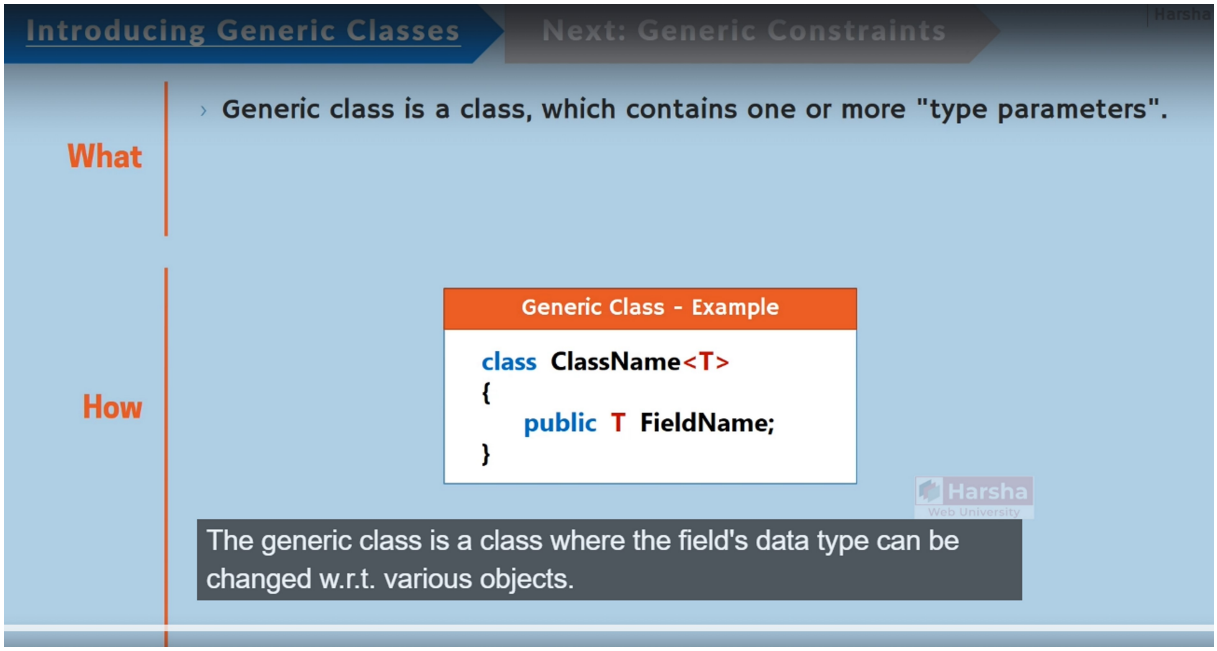
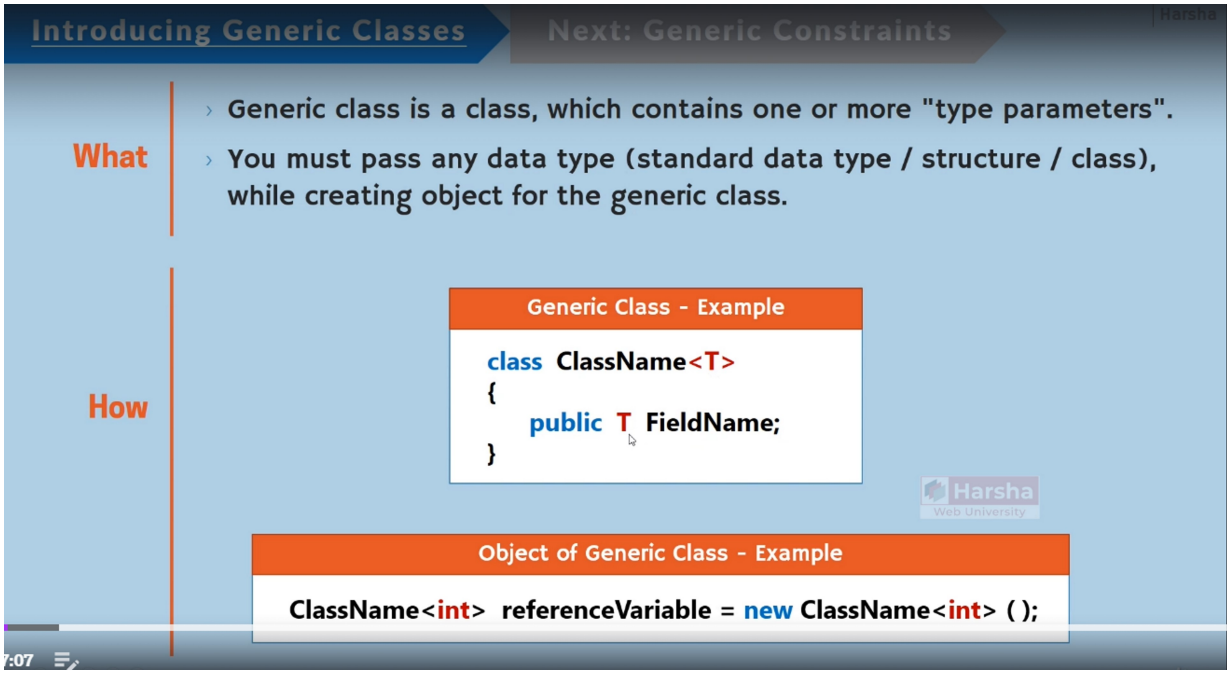
Section 16: Generics

14 July 2024 17:30







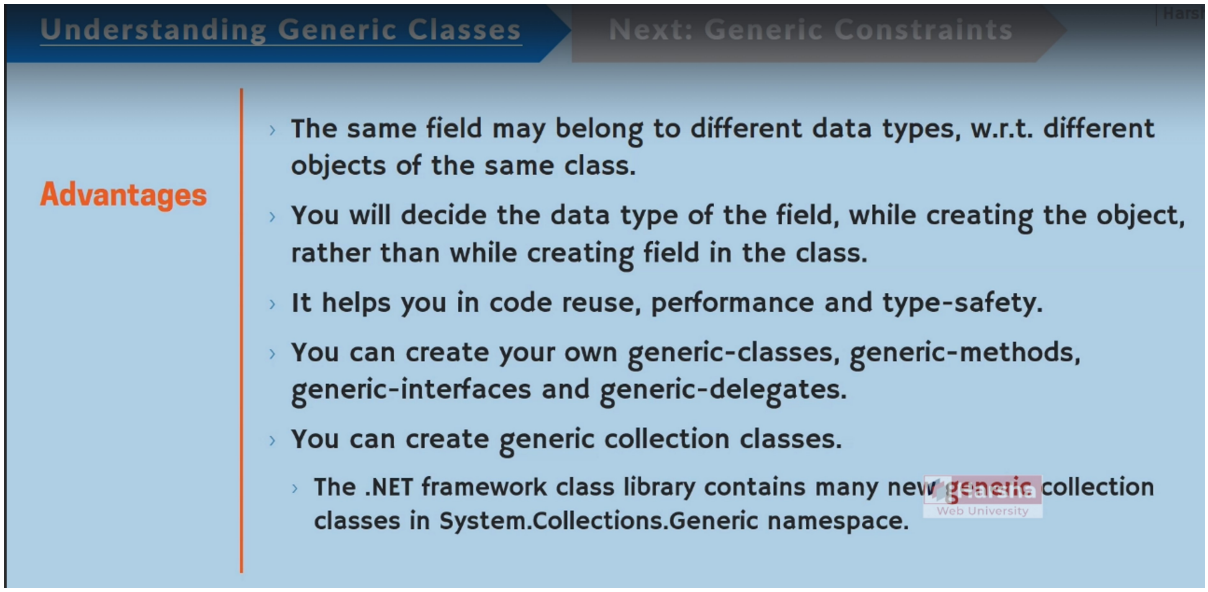
T => Generic Type Parameter.

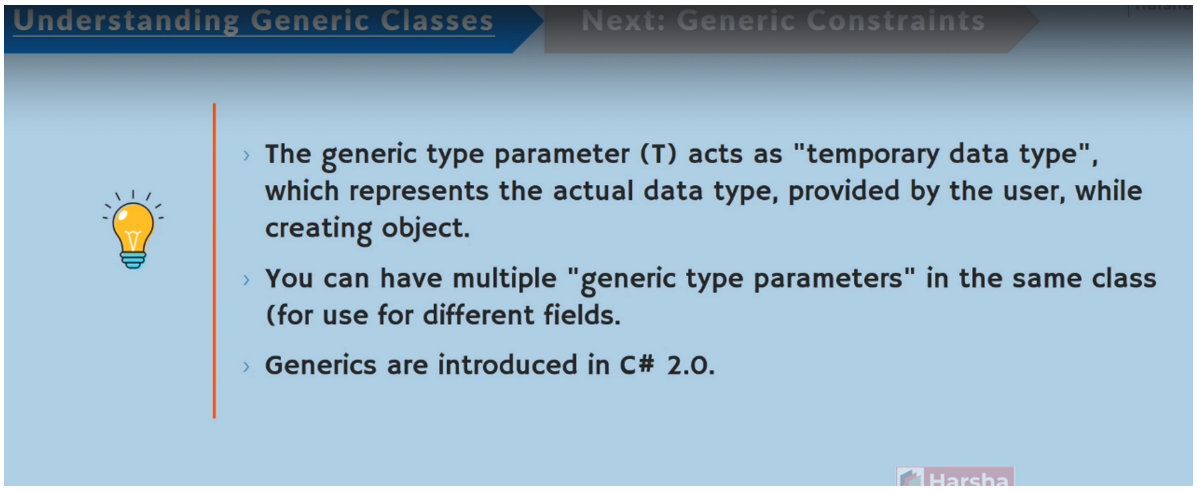
you can use any other letter as generic type parameter.

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.





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 restririctions 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;
}
```

In case if you pass wrong type for the generic parameter, it will be compile time error.

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

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

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

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

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 Methods

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 );
```