Contents

Actions

Motivation

Higher-order programming allows to manipulate for example methods themselves. This can be useful for many purposes, and is called "delegates" in C#. We explain its basics below, and refer to the sorting lecture for an example of how it can be used.

Definition

An *action* is

> a method that has no parameters and does not return a value.

An "Action" is a method that has a single parameter (of type T) and does not return a value.

Here are for example three actions:

```
using System;

public static class ExampleActions
{
    public static void Test()
    {
        Console.WriteLine("Test");
    }
    public static void Display(int i)
    {
        Console.WriteLine(i);
    }
}

public static class ExampleActions<T>
{
    public static void DisplayArray(T[] arrayP)
    {
```

```
        for (int i = 0; i < arrayP.Length; i++)
        {
            Console.WriteLine(arrayP[i]);
        }
    }
}
```

- `Test`, `Display` and `DisplayArray` all have `void` as their return type,
- `Test` does not take any argument,
- `Display` takes an `int` as an argument,
- `DisplayArray` takes "an array of T" (that is, a generic type as an argument.

We can call those easily:

```
        ExampleActions.Test();
        ExampleActions.Display(3);
        ExampleActions<int>.DisplayArray(new int[] { 20,
↪   30, 40 });
```

Manipulating Actions as Variables

We can also store them into variables and then call them:

```
        Action test_variable = ExampleActions.Test;
        test_variable();

        Action<int> display_variable =
↪   ExampleActions.Display;
        display_variable(10);

        Action<int[]> display_array_variable =
↪   ExampleActions<int>.DisplayArray;
        display_array_variable(new int[] { 10, 20, 30 });

        // Passing an action as an argument:
        CallingAction.Demo(ExampleActions.Test);
        CallingAction.DemoT(ExampleActions.Display);

        // Done passing an action.
    }
}
```

As we can see, `ExampleActions.Display` is of type `Action<int>` since the `Display` method takes an `int` as argument.

Action as Parameters

Method can take actions as parameter:

```
using System;

public static class CallingAction
{
    public static void Demo(Action actionP)
    {
        Console.WriteLine("Now calling action: ");
        actionP();
        Console.WriteLine("Done calling action.");
    }

    public static void DemoT(Action<int> actionP)
    {
        Console.WriteLine("Now calling action with
↪   arguments ranging from 0 to 9:");
        for (int i = 0; i < 10; i++)
        {
            actionP(i);
        }
        Console.WriteLine("Done calling action.");
    }
}
```

and then can be passed an action as an argument:

```
        CallingAction.Demo(ExampleActions.Test);
        CallingAction.DemoT(ExampleActions.Display);
```