Introduction to Engineering Using Robotics Laboratories

Lecture 14
From Object-Oriented Computing To Service-Oriented Computing

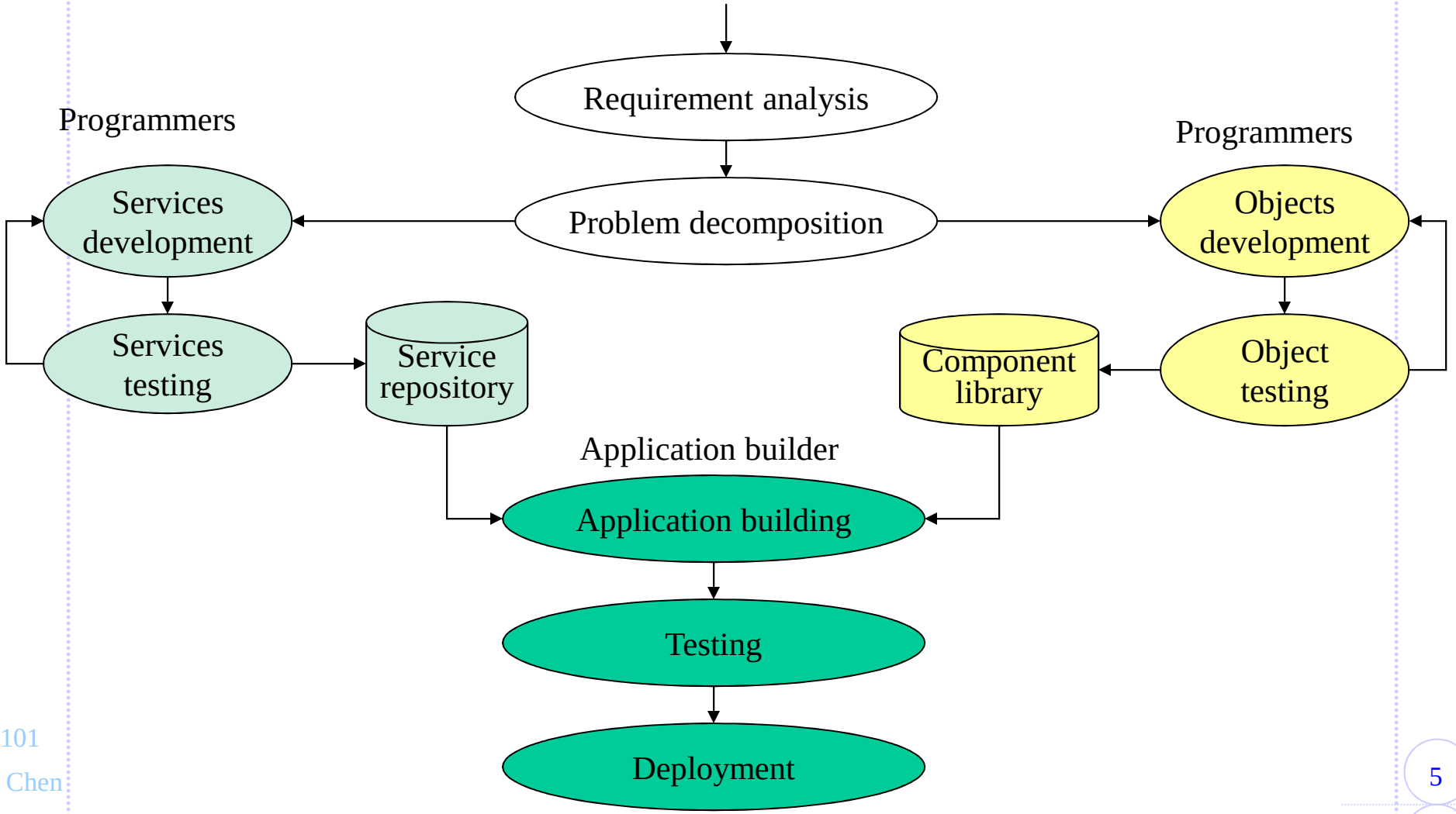
Yinong Chen

Table of Contents

Features of Object-Oriented Computing

- A program consists data and functions (operations) that manipulate data;
- A class consists of data members (variables) and function members (methods);
- *A class is an Abstract data type:* Encapsulation of state in an object that can only be accessed through operations defined on them. Clean interface -- public and private components.
- Inheritance: extending a class by keeping the unchanged parts. Supports code reuse.
- Classes can be organized in a hierarchy through inheritance.
- Dynamic memory allocation and de-allocation
- Dynamic binding
- Polymorphism

Object-Oriented and Service-Oriented Software Development



Travel Preparation

Problem Definition (Requirements)

***Input*:** **The number of days to travel;**

The country name;

The local temperature in Celsius;

***Output*:** **The amount of local currency needed**

The local temperature in Fahrenheit

Problem Decomposition

Main class

- Enter numbers of days to travel;
- Enter the country name
- Make use of other classes to perform computation;
- Print output;

days

amount in USD

Amount in local currency

Celsius

Fahrenheit

- Hotel cost in USD;
- Rental car cost in USD;
- Meal cost in USD;
- Total cost

myCost class

amount in USD

- Convert USD amount into local currency amount;

CurrencyConversion class

- Convert Celsius to Fahrenheit;
- Convert Fahrenheit to Celsius;

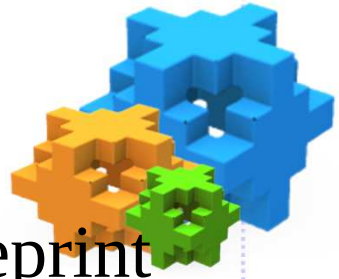
TemperatureConversion class

Class Members

A **class** consists of a list of members

- Each member can be
  - a data member, or called a variable
  - a function member, or called a method
- Function members
  - Constructor: is a function member that has the same name as the class name. It is used to
    - Initialize the data members in the class
    - Pass values into the class, if the values will be used by multiple function members
  - Other function members are used
    - Manipulate data members;
    - Provide reusable functions for other classes to call;

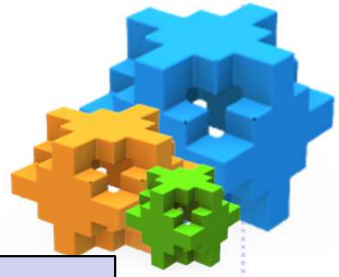
Class versus Objects



1. A **class** is a structural design, or a called blueprint
   - A member can be **private**, **protected**, or **public**
     - private members can be accessed in the class only
     - Protected members can also be accessed by child classes
     - Public members can be accessed all classes in application.
   - **Static** member: If static keyword is used, the member can be accessed without instantiation:
     - *className.memberName*
2. A class can be used to instantiate one or more **objects**
   - *ClassName refName = new ClassName();*
   - *refName.memberName* to access the member
3. A set of functions (methods) are grouped in one class;
4. A group of classes are organized as a **namespace**

Main Class is the class with the Main method

```
using System;
class TravelPreparation {
   static void Main(string[] args) {                    // The main method
      Console.WriteLine("Please enter the number of days you will travel");
      String str = Console.ReadLine();                  // read a string of characters
      Int32 daysToStay= Convert.ToInt32(str);           // Convert string to integer
      myCost usdObject = new myCost(daysToStay);   // Create an object
      int usdCash = usdObject.total();                  // Call a method in the object

      Console.WriteLine("Please enter the country name you will travel to");
      String country = Console.ReadLine();
      CurrencyConversion exchange = new CurrencyConversion();
      Double AmountLocal = exchange.usdToLocalCurrency(country, usdCash);
      Console.WriteLine("The amount of local currency is: " + AmountLocal);

      Console.WriteLine("Please enter the temperature in Celsius");
      str = Console.ReadLine();
      Int32 celsius = Convert.ToInt32(str);
      TemperatureConversion c2f = new TemperatureConversion();
      Int32 fahrenheit = c2f.getFahrenheit(celsius);
      Console.WriteLine("The local temperature in Fahrenheit is: " + fahrenheit);
   }
}
```



Class name

Return type

Method name

No constructor

```
class TemperatureConversion
{
    public Int32 getFahrenheit(Int32 c)
    {
        Double f = c * 9 / 5 + 32;
        return Convert.ToInt32(f);
     }
    public Int32 getCelsius(Int32 f)
    {
        Double c = (f - 32) * 5 / 9;
        return Convert.ToInt32(c);
    }
}
```

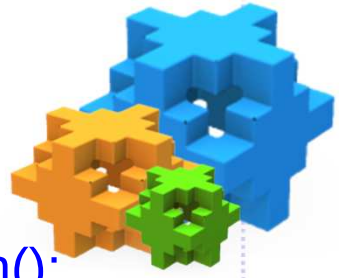
Method 1

Methods2

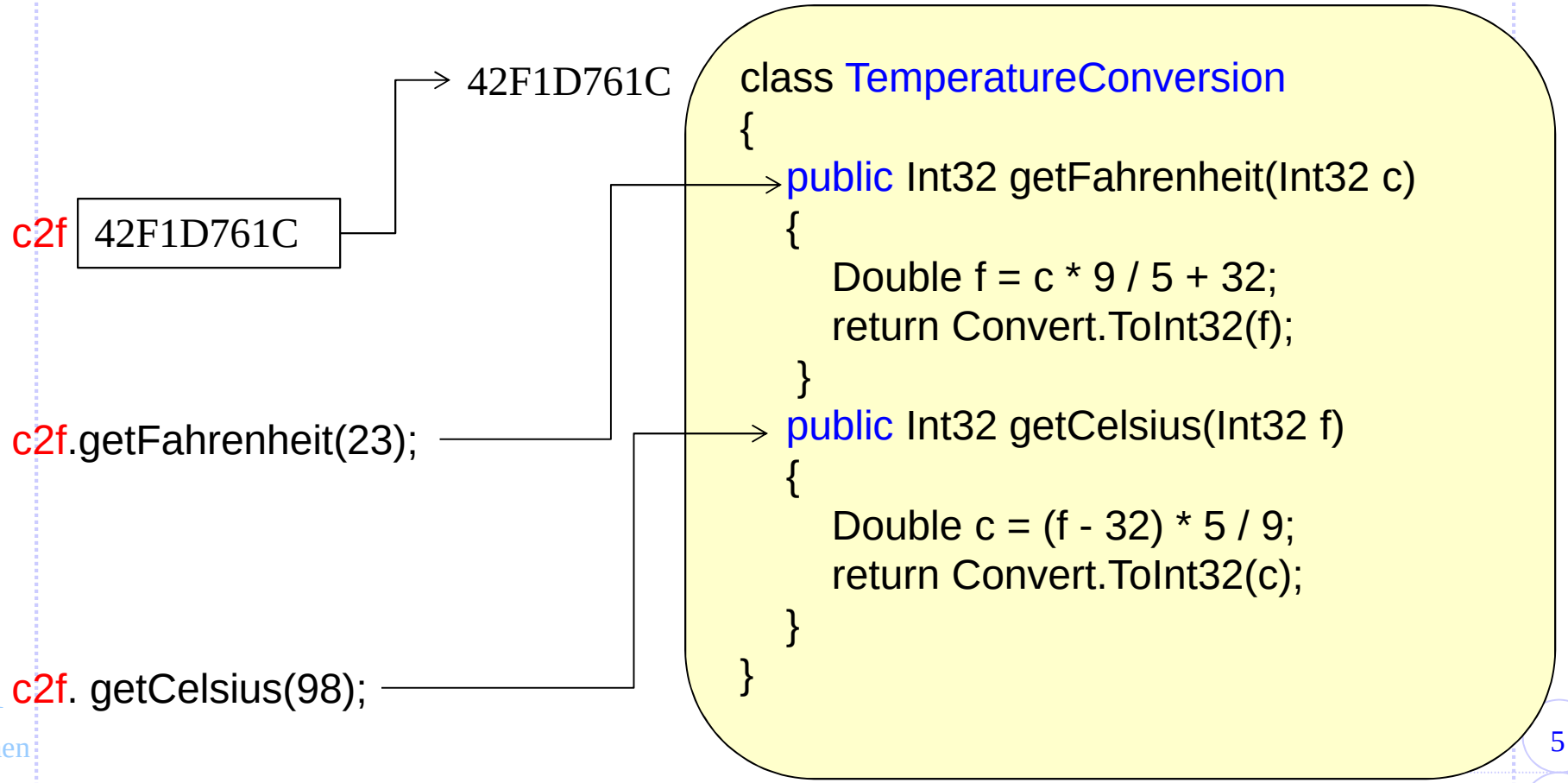
Parameter passing into function directly

```
TemperatureConversion c2f = new TemperatureConversion();
 Int32  fahrenheit = c2f.getFahrenheit(celsius);
```

Reference to an Object



```
TemperatureConversion c2f = new TemperatureConversion();
 Int32  fahrenheit = c2f.getFahrenheit(celsius);
```



```
class myCost {
    private Int32 days;

    public myCost(Int32 daysToStay)  {
        days = daysToStay;
    }

    private Int32 hotel() {
        return 100 * days;
    }

    private Int32 rentalCar() {
        return 30 * days;
    }

    private Int32 meals() {
        return 20 * days;
    }

    public Int32 total() {
        return hotel() + rentalCar() + meals();
    }
}
```

Data member

Constructor: is a method that has the same name as the class. Allow to pass a value into the class and used by all methods

The value is provided when an object is created.

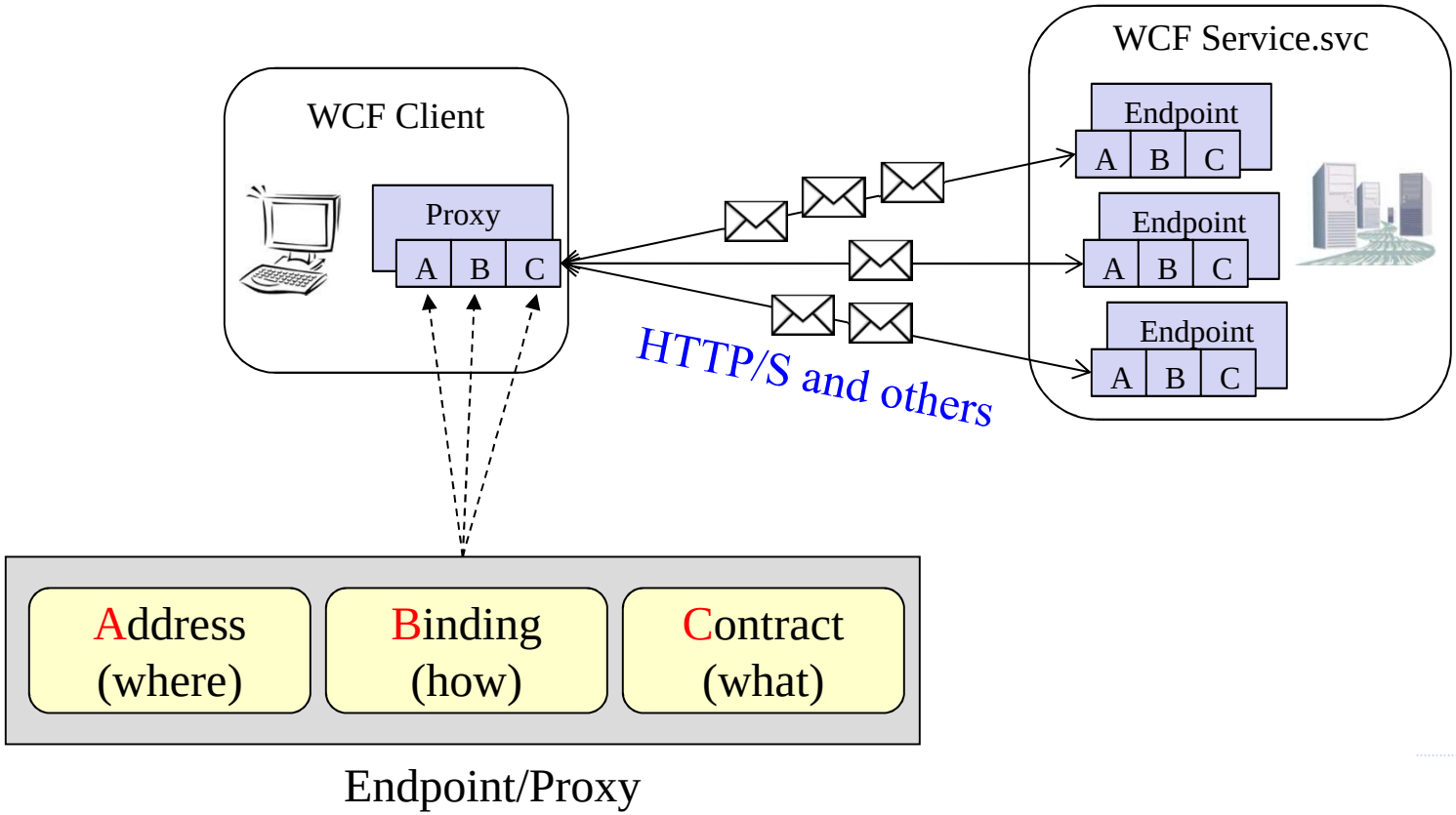
```
myCost usdObject = new myCost(7);
int usdCash = usdObject.total();
```

```
class CurrencyConversion  {
    public Double usdToLocalCurrency(String country, Int32 usdAmount)  {
        switch(country)  {
            case "Japan":
                return usdAmount * 117;

            case "EU":
                return usdAmount * 0.71;

            case "Hong Kong":
                return usdAmount * 7.7;

            case "UK":
                return usdAmount * 0.49;

            case "South Africa":
                return usdAmount * 6.8;

            default:
                return -1;
        }
    }
}
```

This class does not need to have a constructor: It has no data member;
The parameter is passed to the function member, instead of into the class for all member functions to use.

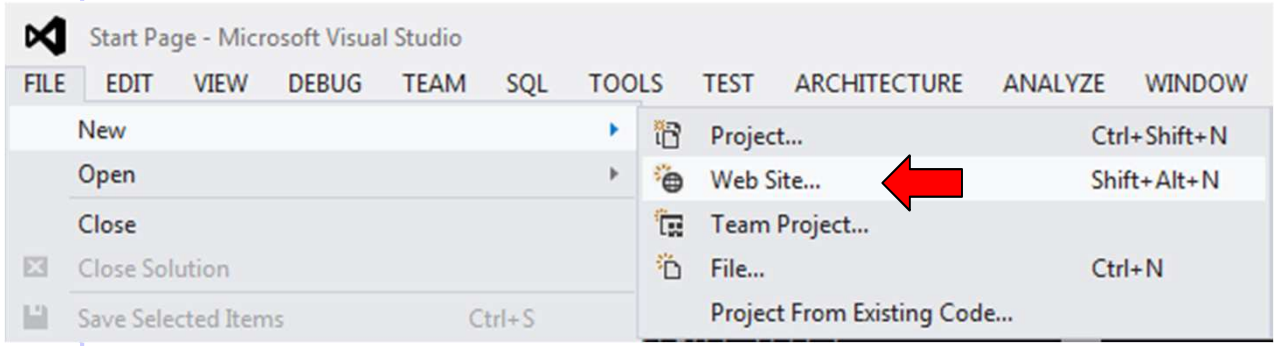
There is one function member only in this class. If there would be more methods and all methods use the same parameters, we would want to pass the parameters into the class, instead of into individual methods.

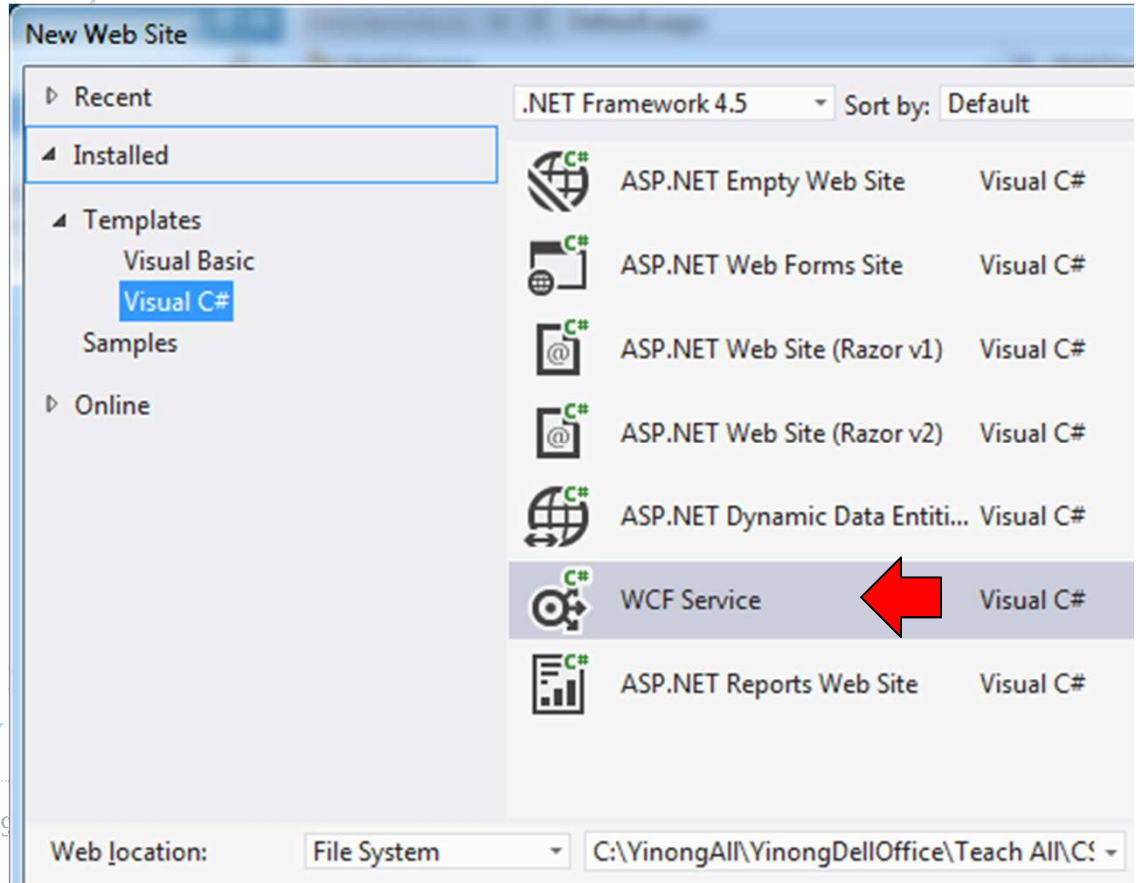
Developing Web Services in Windows Communication Foundation

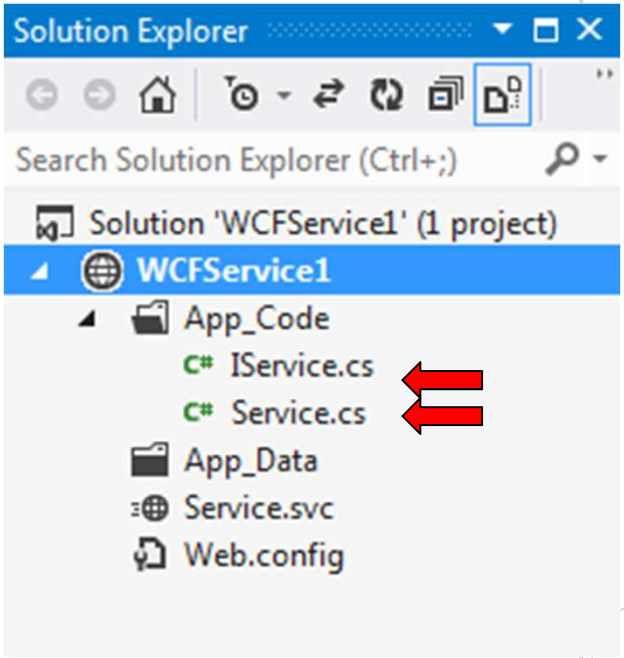


Develop Web Services Using WCF

Text Chapter 3, section 3.2.1
You did this exercise in assignment 1







IService.cs and Service.cs Files

App_Code/IService.cs | App_Code/Service.cs

IService | HelloWorld()

```
using System;
using System.ServiceModel;
// NOTE: If you change the interface

[ServiceContract]
public interface IService
{
    [OperationContract]
    string HelloWorld();

    [OperationContract]
    double PiValue();

    [OperationContract]
    int absValue(int intVal);
}
```

App_Code/IService.cs | App_Code/Service.cs*

Service | absValue(int x)

```
using System;
using System.ServiceModel;

// NOTE: If you change the class name
public class Service : IService
{
    public string HelloWorld()
    {
        return "Hello World";
    }
    public double PiValue()
    {
        double pi = System.Math.PI;
        return (pi);
    }
    public int absValue(int x)
    {
```

Address (where)

Binding

Contract

ASU Repository | Service Service

venus.eas.asu.edu/WSRepository/Services/BasicThreeSvc/Service.svc

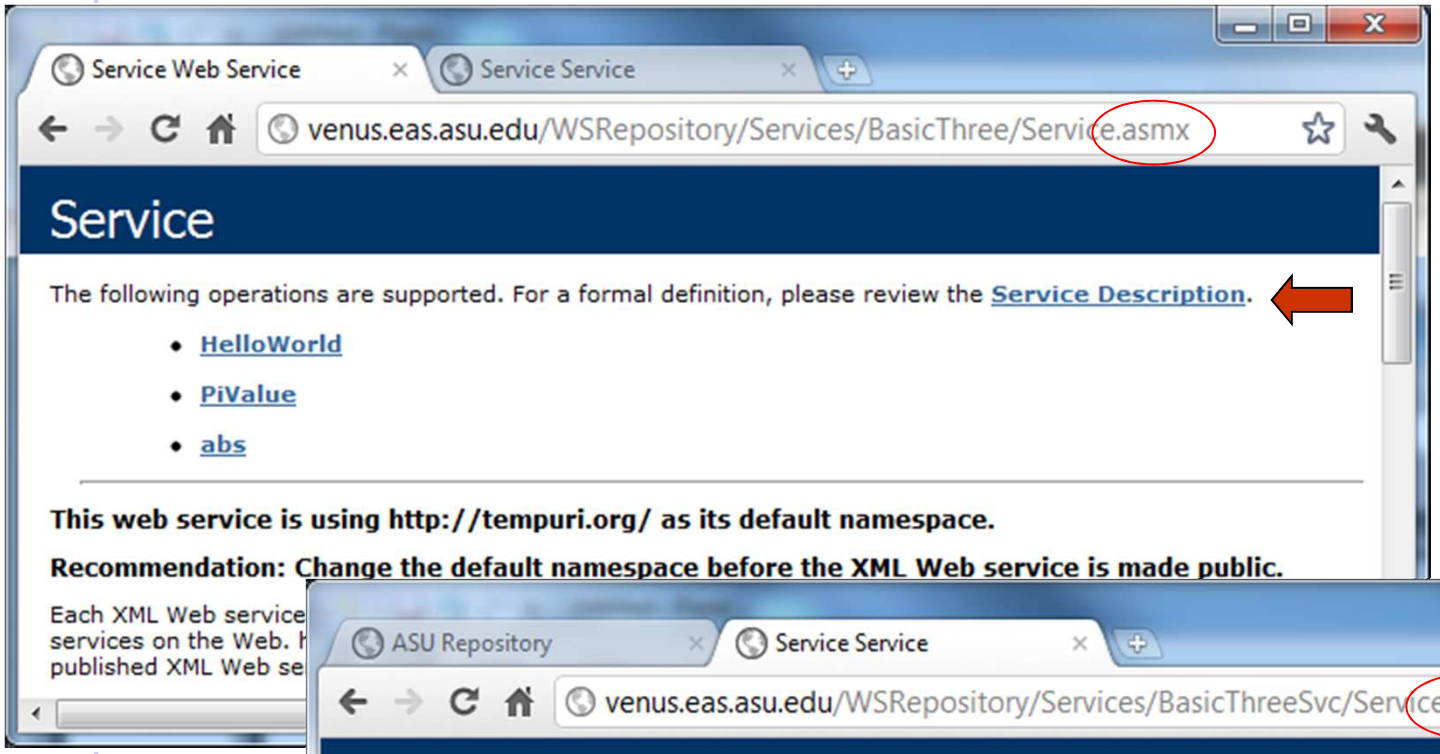
Service Service

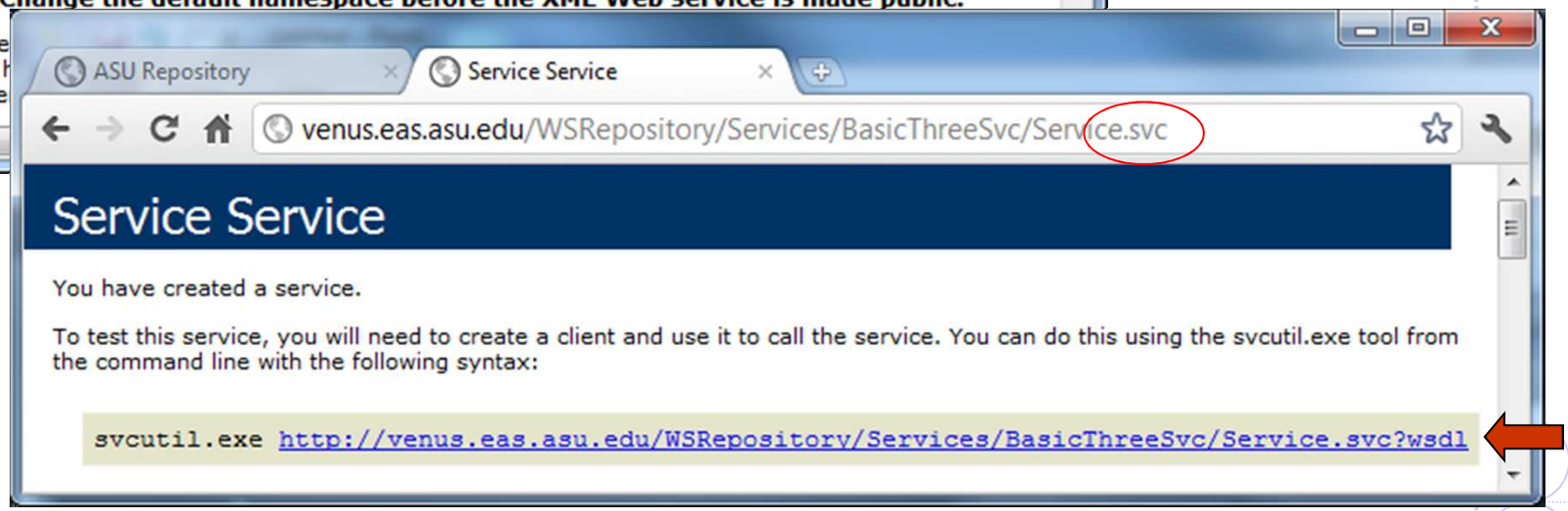
You have created a service.

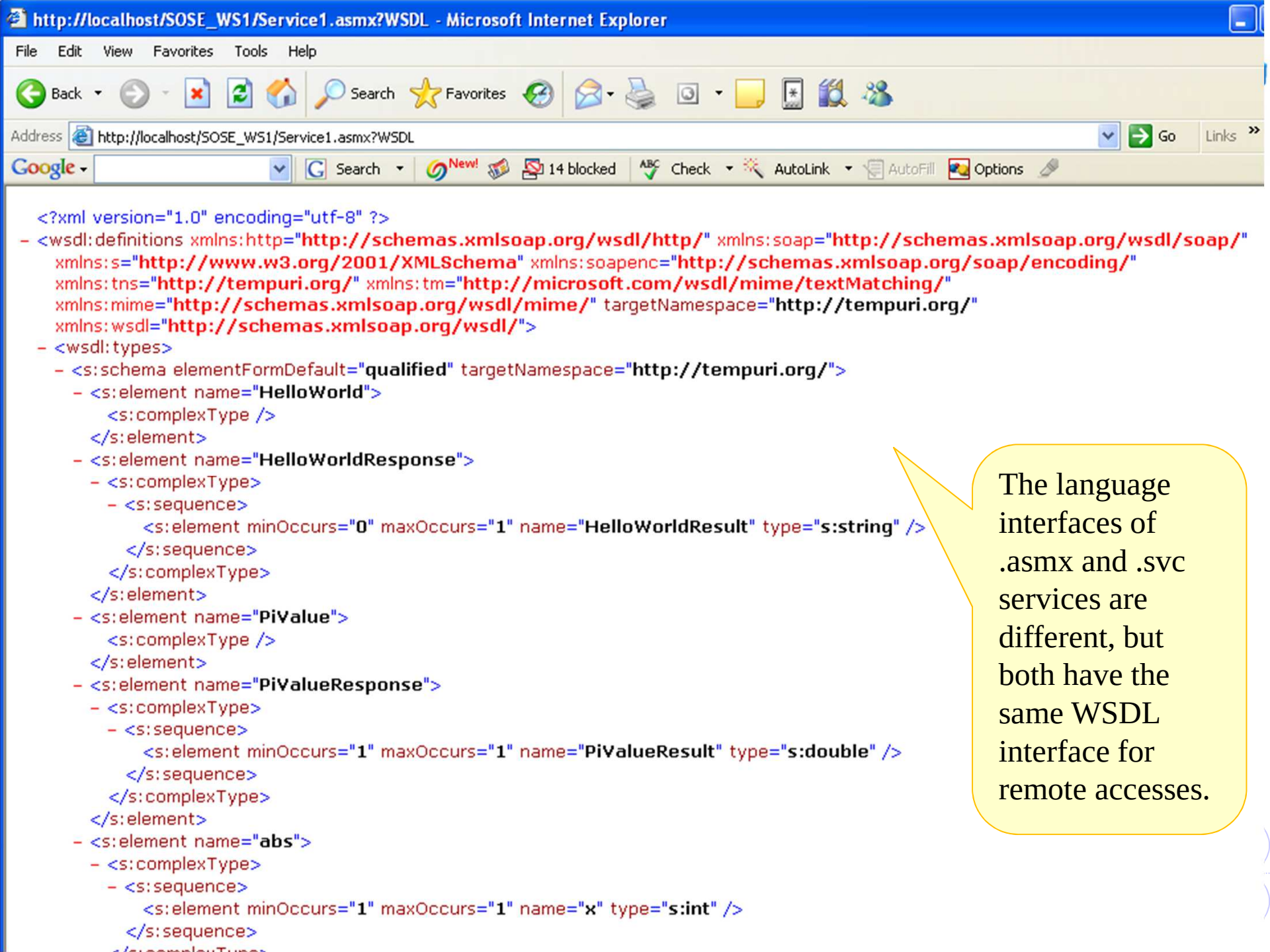
To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://venus.eas.asu.edu/WSRepository/Services/BasicThreeSvc/Service.svc?wsdl
```

Difference between .asmx and .svc services



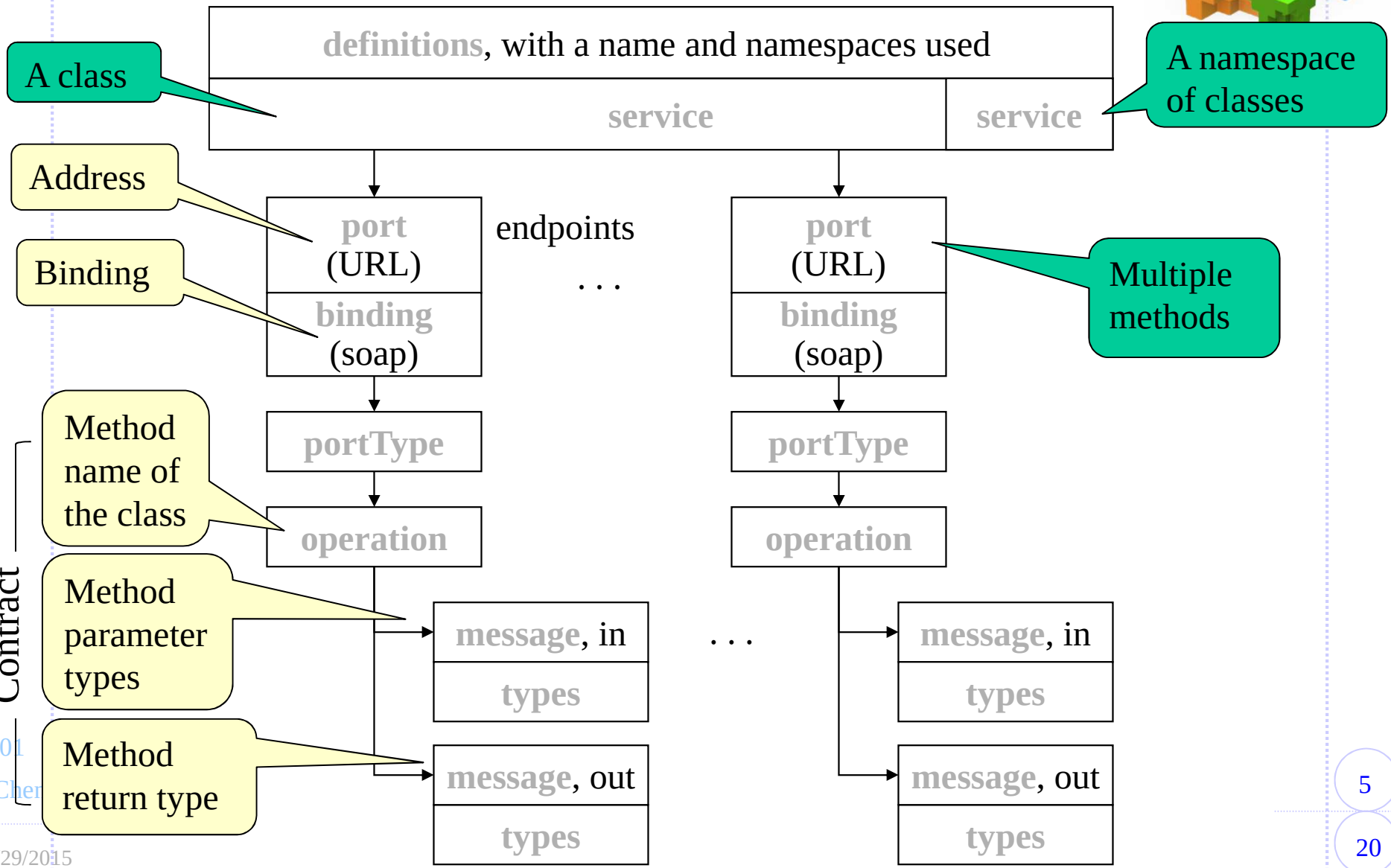



http://localhost/SOSE_WS1/Service1.asmx?WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/SOSE_WS1/Service1.asmx?WSDL
Go Links
Google Search New! 14 blocked Check AutoLink AutoFill Options
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://tempuri.org/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="HelloWorld">
<s:complexType />
</s:element>
<s:element name="HelloWorldResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
<s:element name="PiValue">
<s:complexType />
</s:element>
<s:element name="PiValueResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="PiValueResult" type="s:double" />
</s:sequence>
</s:complexType>
</s:element>
<s:element name="abs">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />
</s:sequence>
The language interfaces of .asmx and .svc services are different, but both have the same WSDL interface for remote accesses.

WSDL: Web Service Description Language

- WSDL is used to describe Web services, including four critical aspects of Web services:
  - Functionality description of the services in standard taxonomy;
  - Contract of parameter types and return type of the function (service) calls;
  - Binding information about the transport protocol to be used, usually, SOAP;
  - Address information for locating the specified service.
- The last three aspects can be automatically generated.
- Web services described in WSDL can be searched, matched with the requirement.
- Web services described in WSDL provides the remote call detail.

Logical Structure of WSDL Document's Elements



Summary

- An object-oriented application consists of multiple classes;
- Each class consists of data members (variables) and function members (methods);
- Each member can be public, protected, or private;
- A service corresponds to a class;
- A service typically does not have data members;
- Not all public methods of a service can be accessed remotely;
- Only public methods further marked with [OperationContract] can be accessed remotely;
- A service can have multiple methods marked with [OperationContract]