目录

- Lecture 1 - Page 2
- Lecture 2 - Page 8
- Lecture 3 - Page 14
- Lecture 4 - Page 19
- Lecture 5 - Page 24
- Lecture 6 - Page 30
- Lecture 7 - Page 36
- Lecture 8 - Page 40
- Lecture 9 - Page 43
- Lecture 10 - Page 46

Section one: Getting started.

1. welcome to the course. over view

- basic programming in C# ; variables, constants, functions
- more complex: Object orientation, classes, methods, stats, enemies, error handling with exceptions, file handling...
- C# for Microsoft windows. but also mac, linux.
- Visual studio for C#
- text-based exploring game

5. What's in step 1?

- design user interface, text entry field, buttons.
- sample project using
- software install - visual studio
- course e-book and little c# book and update&Resources
- create and edit C# programs

6. install Visual Studio

① download Visual studio (community) in Visual studio.com

② ".Net desktop development" with C# (workload)

③ 100% scalling ⇒ automatic scaling in VS

④ open Calc.sln

8. Your First C# project

① File → New → Project

② Visual C#

③ ① UWP = Universal Windows Platform (win 10)

② WPF = Windows Presentation Foundation (hard but beautiful)

chosen ③ Windows Forms App (button, text box)

④ Console App (run from system prompt, non visual, no UI)

chosen Windows Forms App

⑤ Frame work: .NET 6.0 (Long-term support)

9. Adding components to a form

- View → Toolbox → button (drag to the Designing form)
- properties → Text → change text
  → Multiline → True

10. Adding program code

① properties → lightling button → click

② (Form1.cs)

```
private void button1_Click ( sender , e )
{ textBox1 . Text = "welcome to my program";
```

↑ name ↑ function

③ Ctrl + F5

11. How to use sample projects

① open Learn CSharp .sln

② right click → choose "Set as startup Project

③ "ctrl + F5" to start without debuging.

④ If error message , VS said has yet to be built, right click and choose "rebuild" source code

⑤ right click → "View code" in "Design"

12. How to customize Visual studio

- tools → options → Text editor → C#
- Edit → advanced → Formated document
- tools → environment → Fonts and colours
  → General.

13. Tour of Visual Studio

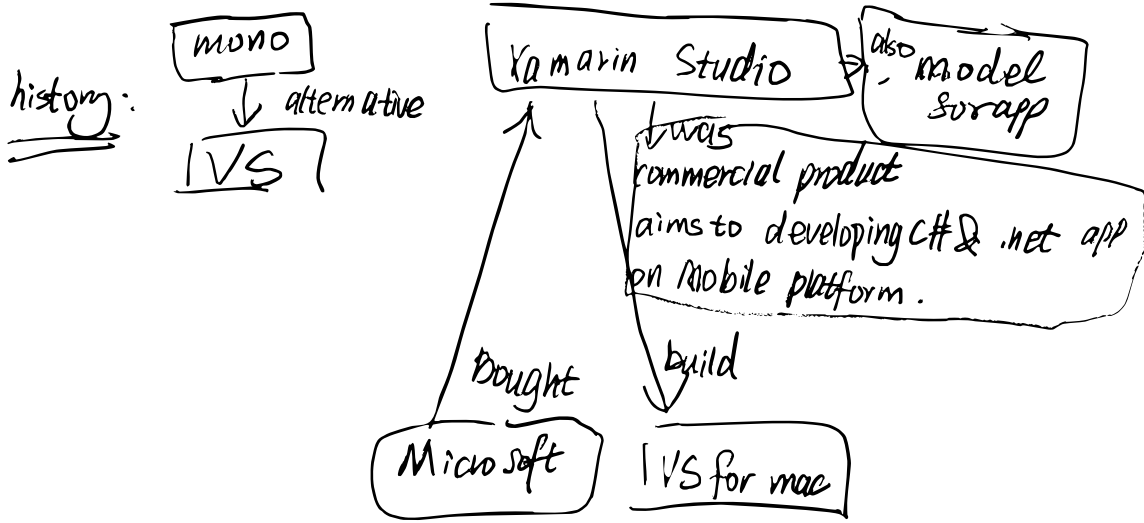
- Build → compile your code
- debug → run and debug your object,
  auto compiles any changes your made,
  so you don't need to build in seperate
  step.
- Form → design mode → can use "toolbox"
  associate with C# code file too
- Solution explorer
- Error list

14. How to follow this course on a Mac

- Mono Develop

15. Install Mono and MonoDevelop on a Mac

① "mono develop.com" ② "mono + GTK #" download.
③ download "Xamarin studio"



16. How to use the sample projects on OSX

17. How to create visual user interfaces with Mono Develop

Quiz 1: First steps in C#

① • <u>Event</u>: things that can happen (such as mouse-clicks and mouse movements) while your application is running.

- <u>Control</u>: is a visual component such as a button.
- <u>property</u>: some feature of a control (or other object) such as the Text of a button.

② textBox1.Text = "Hello world";

↑ control (name)

↑ property (feature of control)

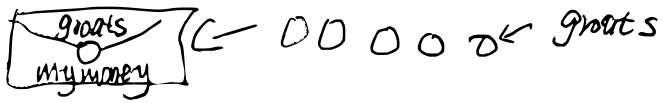
Section 2 : Step two - Programming Fundamentals

18. What's in step 2?

- essential features : data types, variables, constants, functions, keywords, name spaces, type conversion.
- chapter 2 in the course e-book. little book of C#.

19. data types

- int - integer
- double - floating point
- JavaScript & Ruby not require data type in variable, C# does.
-



value = five groats.

20. variable

* (open step2. data type) semi colon

- string my string = "Hello world!";
- char my char = '!';
- int my int = 100;
- double my double = 100.75 * TAXRATE;
- textBox1.Text = msg;
- textBox1.AppendText("You owe:" + my double + "groats!")
- my int = 200;

value can be change

▲ use "\r\n" for new line,
'\n' not work for new line

* open step2 · TaxCalc Text

step ① assign "0" for double variable "double subtotal = 0;"
step ② change value " subtotal = 12.5;"
step ③ display value in form " subtotal TB.Text = subtotal.ToString();"

21. Constants

constants

- value can not be changed: const double TAXRATE=0.2;
- value can change: double TAXRATE = 0.2;
  TAXRATE = 0.3;

22. comments

```
/*
 * This is multi-line comment
 *
 */
```

// This is line comment

23. keywords

search "Microsoft documentation site" for key word.

→ .Net ⇒ C# language reference ⇒ key word

24. Function (method)

```
private void SayHello() { ... }
private void greet( string aName ) {  }
        greet( Some Name )
private string addNumbers( int num1, int num2) { return ...; }
```

25. Namespaces and usings

- if more using statement:
  "light bulb → Preview changes → ok."
- using: gives C# code access to classes and types inside another code module or namespace.

two ways:
① using System
string.Format
② system.string.Format

two ways:
① System.IO : dealing with disk files and directories.
↑ nested inside System
② System.IO.Directory.GetCurrentDirectory();

26. Data type conversion - parse( )

- phase:
  double d = Double.Parse("15.8");
  "15.8" (string) → 15.8 (double)
  int i = Int16.parse("15");
  textBox1.Text = String.Format("{0}-{1}", d, i);
- string can't converted.
- .ToString : grandtotal.ToString(). 16.5 (double) → "16.5" (string)
- Convert.ToDouble() = Double.Parse( )

27. Type casting

- cast is more limited than convert can be.

  only between number, can not number to string.

cast • rounded Total = (int) grandtotal;

convert • rounded Total = Convert.ToInt32(grandtotal);

- this : the form.

  this.Text = Text = the form title.

28. Implicitly typed variables

- C# allow declare and initialize local variables without specifying a type.
- var - C# figure out the actual type. C# not allow the type of variable to be changed.

```
var str = "Hello world";
str = 100;
```

Can't implicitly convert type "int" to "string"!

- may useful when get data from database.
- another example

{100, 200.5, "Hello"}

```
for each (var thing in somethings) {
    textBox1.AppendText(thing + "is a:" + thing.GetType()
                                + "\n");
}
```

29. Summary of C# Languages Elements

- statements end with ";" simi colon.
- either call function or assign/declared variables.
- keywords. function,
- using, code inside namespaces.
- comment, multi-line comment.

30. Sample program = Tax Calculator

- declare variable before use it.

Quiz2 : Datatypes

Section 3: Step three - Condition Tests and Operators

- testing - make decision while the program is running.
- if, else
- operator

32. if (*step 03. if else)

```
private void ifBtn_Click( object sender, EventArgs e){
    double someMoney;
    someMoney = Double.Parse (textBox1.Text);
    if (someMoney >100.00)) {
        MessageBox.Show ("You have lots of money!")
    }
}
```

33. if... else

```
else { MessageBox.Show ("You need more money...");}
```

34. Single line statements with tests

- {} bracket can omit if{}, else{}.

35. if .. else .. if

```
else if ( (userinput.Text == "Hello") || (userinput.Text == "hi") ) {
    output.Text = "Hello to you too!";
}
```

"==": to test if value on the right and left is same.

"=" : assign value to the right

36. switch statement

```
switch (userinput.Text) {
    case "":
        output.Text = "You didn't enter anything";
        break;
    case "Hello":
    case "hi":
        output.Text = userinput.Text + " to you too!";
        break;
    default:
        output.Text = "I don't understand that!";
        break;
}
```

no break → case "Hello":

37. Operators

- = assign | == equal
  int x = 1; | if (x == 1)
- test more than one condition true: logical operator.
- comparison operator ≤ ≥
- arithmetic operator

38. Arithmetic operators

- + - / * %
- "Result=" + 155 + 3 ⇒ Result = 155+3 (just concatenated)
  "Result=" + (155+3) ⇒ Result = 158 (with parentheses have addtion operation before concatenated)
  co

39. Comparison operators

- == != > < <= >=
- if (i == j) { } else { }

40. Logical operators

- && || !
- if (! (age <= 30)) && (salary > 30000) { }

41. Compound assignment operators

- += -= *= /=
- a += b == a = a + b

42. Increment and decrement operators

- num ++ ; num -- ;
- unary : one operand
- binary : two operands

43. Prefix and Postfix operators

- a = 10
  b = ++a ; // prefix a = 11 b = 11
- b = a ++ ; // postfix a = 11 b = 10
- b = a+1 ; // just write simply.

44. Parentheses and precedence

- if (( ) && ( ) && ( )) { }
- order of valuation, precedence, search on microsoft web.

45. Complex texts

- if (( ) && ( ) || ( ) && ( ))
= if ((( ) && ( )) || (( ) && ( )))
- too difficult should clear the code.

46. Summary of operators

- && , || , :
- + , - , * , /

Section 4: Step Four : Functions and Arguments

47. What's in Step 4 ?

- function / methods and arguments ← data that send to functions
- special argument type = ref , out
- special function type: local functions, overloaded functions and recursive functions.
- terminology: function / method, argument / parameter.

48. What is a function ?

- functions : way of dividing code into named chunks.
- call function when it is needed.

① • declared function using a key word such as : private , public

define function visibility.

• follow data type of any value that returned by the

② function `int add ( ) { return num; }`

• `void greet ( ) { }`

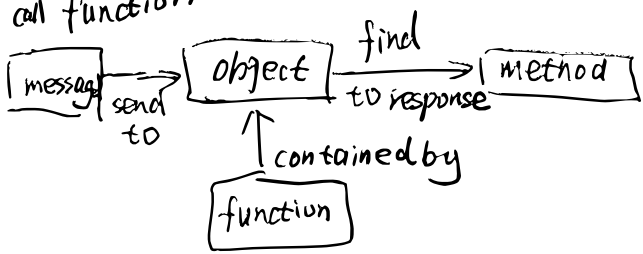
nothing return

③ • name : chosen by programmer

④ • arguments (with data type) : (int num1, int num2)

49. What is a method?

- call function



- C# is oop language.

50. Arguments and parameters

☆ more common to called

- when called function
  greet(someName)
- in method
  private void greet(a Name)

51. Return values

- textBox1.AppendText(addNumbers(550, 750));

52. Arguments by Value, and By Reference

in argument:

- by value - copies of original data.

```
first = 10
second = 20
byValue(first, second)

private void byValue( int num1, int num2) {
    num1 = 0;
    num2 = 1;
    textBox1.AppendText("In by Value num1= "+num1 +
                        ", num2 = " + num2); }
```

⇓

In by vaule, num1=0, num2=1
Now first=10, second=20.

- by reference

```
private void byReference( ref int num1, ref int num2){
    num1=0;
    num2=1;
    textBox.AppendText("In by Reference, num1 = "+ num1 "+
                       "num2= " + num2 + "\n"); }

int first = 10;
int second = 20;
    byReference(first, second);
```

⇓

In by Reference, num1=0, num2=1
Now first=0, second=1

53. Out arguments

- out argument - can just call function without assign value.
- ref argument - must assign value to argument before call function.

- calling it:

```
outParams( out first, out second);
byreference( ref first, ref second);
```

54. Method overheading

- same name but parameter is different.

55. Local Functions

```
private string ShowSalary( string aName,
                           int earnings)
{
    double addBonus( ){ }
    bonus = addBonus( );
}
```

56. Sample program : TreeView

```
TreeNode newNode = new TreeNode("[New Node]");
treeView2.Nodes.Add(newNode);
```

press F1 to get Document from TreeNode

TreeView

collection method

- Add()
- Clear()
- Insert()
- Remove()

58 Recursive functions

- breakpoint (debugging)
  in Local

Quiz4: Functions, methods and arguments

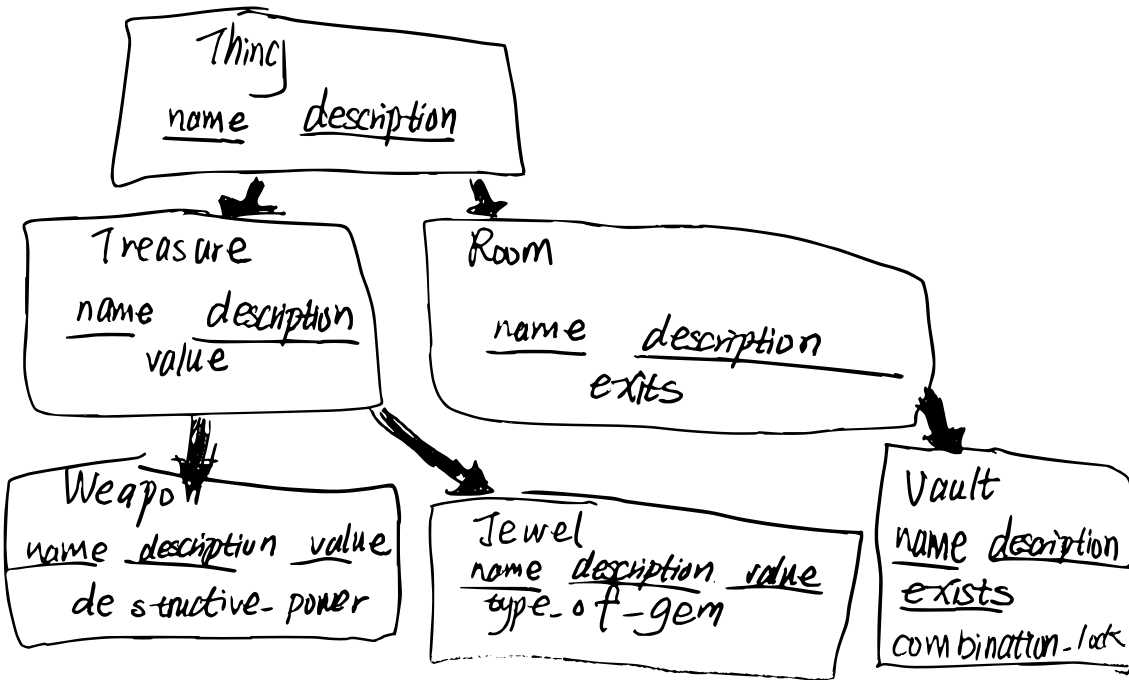
Section 5 : Step five - Object Orientation

60. What is in steps?

- encapsulation = putting the functions and variables inside an object.
- inheritance : Book object ⇒ novel object
- constructors , properties, access, modifiers.
- define and create object.

61. Object Oriented Programming explained (OOP)

- Class — blueprint of object.
- adventure game



62. How to write a class definition (blueprint) (class)

- press [F7] to go to the code.
- private string _s;
- public My class () {

  _s = "Hello World";

  }

- public string GetS() {

  return _s;

  }

- public void SetS (string aString) {

  _s = aString;

  }

☆ variable private + method public

63. How to create an object from a class

Button:

- My class ob;
- Create Ob [ob = new MyClass();]
- Set string (ob.SetS("A new string");]
- Get String [TextBox1.Text = ob.GetS();]

[☆ But wrong order click button will get error.]

64. Testing if an object exists

```
if (ob != null) { }
```

65. Visual Studio Refactoring

- Rename variable: right click → "Rename" button → Rename it.

66. Creating class hierarchies

parent class

1. define private string;
2. refactoring - geter and setter.
3. Construstor:

child class: Treasure: Thing {

1 2 same

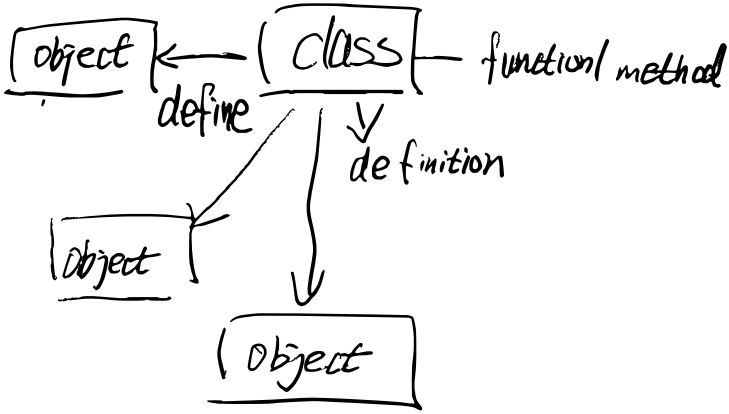
3 Constructor: public Treasure (aName, aDescription, aValue)
: base (aName, aDescription) { }

}

67. Change to one class per file

- right click → refactoring → "more" button

68. Constructors



69. Access modifers: public, private, protected

- protected: visible in ascendant classes Get z ☑
- private: only inside the class

☆ both can't directly access, but protected value can return in ascendant class.

70. Properties

- red only : get
- set : change value

71. Naming conventions

- coding style : C# is case sensitive.
- variable start lowercase , function start uppercase.

variable

property

```
exit = r.N;
```

```
private int _n;
```

underscore.

72. Summary of Object Orientation

- Class define object.
- variable private : data hidden outside of object
- public function
- get / set : return / set value in variable.
- create object from a class by calling the class constructor.

Quiz 5: Object Orientation

Section 6: Step Six - Arrays and Strings

73. What is in Step 6?

- arrays
- indexes
- start zero to (length - 1)
- for loops, for each loops, while loops.

74. Arrays

- String myArray

index	0	1	2	3
	"one"	"two"	"three"	"four"

0 base

- string [] myArray = new string [4];
- string [] myArray = new string [4] { "one", "two", "three", "four" };

15. Iterating array items with a "for" loop

for(int i = 0; i <= 2; i++) { }

16. Arrays of objects

- can create arrays in any type of object:
  - strings, integers, doubles, or even user define object.
- for int i = 0; i <= 2; i++) {
  myObjectArray [i] = new MyClass ("Object #"+i+"."); }
- (i <= myArray.length - 1)

  or

  (i < myArray.length)
- GetLowerBound(0), GetUpperBound(0).

77. foreach loops

- foreach (MyClass myob in myObjectArray) {
  textBox4.Text = textBox4.Text + myob.Name;
}

78. Strings

- textBox1.Text.Length;

79. while loops

- while ( ) { }

80. do while loops

- do while {
- do {
  } while (i<5);
  }
- do while loop at least run once time.

☆ 81. Sample Program: Text Editor

- "Tool Box → "Dialogs" → "Open File Dialog" "save File Dialog"

? 不知道怎么做 open file dialog.

82. String Operations

- String . Format

  To Upper

  To Lower

  Pad Left

  Pad Left (20, '-')

  Pad Right (30, '*')

  IndexOf ("World")

  IndexOf ("Moon")

  Replace ("World", "Moon")

  Append ("...")

  Trim Start( )

  Trim End (Trim Chars))

  Trim (Trim Chars)

83. String Builder

- StringBuilder sb = new StringBuilder("Hello World");

- s = sb.ToString();
  // convert to string.

- String Builder change (mutable)
  String (unmutable) can't change, just create a new String.

84. String Format

- String.Format("{0} of {1}", 1, 2).

85. String Interpolation

$ → use {}
interpolation

$"\n This is a \t {my string} \n" // string
@"\n This is a \t {my string} \n" // verbatim string

This is a Test

86. Summary of Arrays and Strings

- index of array = [0, length of array - 1]
- for loop, for each loop, while, do while loops
- trim white spaces of characters.

next : file handling

Section 7: Step Seven - Files, Folders and Input/Output

87. What is in Step 7?

- save your work and load data

88. Basic File Operations

- IO
- File.Exists : check file is exists.
- File.Copy (sourse file, destination file, Boolean)
- File.GetCreationTime (file)
- File.GetLastWriteTime(file)

static method

true/false

- static method: are the methods that belong to the class. don't need to create a object

- DateTime dtCreated;
- DateTime dtModified;

89. Disk and file-handling methods

useful class
- Directory Members.
- Environment Members
- Path class

90. Streaming Data

- Stream = a sequence of data that can flow from one place to another.
  - typically that's between computer's memory and a storage device such as a disk.
  - stream is made up of bites.
- const string OUTPUTFN = "C:\\Test\\Stream.txt";

class ①
```
Stream instream = File.OpenRead(SOURCEFN);
Stream outstream = File.OpenWrite(OUTPUTFN);
```

```
byte[] buffer = new Byte[BUFFSIZE];
```

- while(numbytes = instream.Read(buffer, 0, BUFFSIZE)) >0
  { outstream.Write(buffer, 0, numbytes); }

- ② class BufferedStream buffInput = new BufferedStream (instream);

- class ③ FileStream instream = new FileStream(Source, FileMode.OpenOrCreate, FileAcress.Read);

91. Reading and writing text files

- Stream Reader sread;
- Stream Writer swrite;

☆ 92. Using File-handling in your own programs

Task: • pick a project that interest you to code in C # language

☆ 93. Choosing a programming project

- example code in this course.
  - step six : Text Editor
  - step six : word count function
  - step six : back up option
- (advantage) • file management tool
  - Tree component (step 4)
- step seven code also.

94. Summary of File Handling

Quiz 7: Files and streams

Section 8: Step Eight - Classes, Methods, Structs and Enums.

9s. What is in Step 8

- ways of organizing project that contain numerous files.
- partial classes : classes split up several files.
- static methods and overloaded methods
  (no create object) (step four)
  constructor method
  overloaded method.
- static class - can use to provide method, but can't be create object
- Structs
- Enums - grouped lists of enumerated constants.

96. Partial classes.

- class, method.
- partial class MyClass

97. Static and overloaded methods

- public static string - static method.
- can have multiple construtors.

98. Static class

- Static class: don't want anybody create method from the class.

99. Using static and overloaded methods

- const string CRLF = "\r\n";

100. Structs and enums

- Structs: can not inherit from a struct. but can inherit from a class.
- Enums: collection of constants.

- Dock
- Anchor

101. Summary of Advanced classes and methods

- <u>over written method</u>: a method that can be defined many times in a family tree of class.

Section 9: Step Nine - Exception Handling and Debugging

102. What is in Step 9?

- program bug = use visual studio debugger to step through your code to find and fix bugs.

103. Exceptions

- try { st = . . . ; }

  catch (Exception exc) {
  MessageBox . Show (" ") ; }.

  - exe. message
  - exe. GetType ( )
  - MessageBox Buttons. OK
  - MessageBox Icon . Error.

104. Catching different exception types

- can catch multiple exception type.

105. try .. catch .. finally

106. Nested exception handling

- handle any error occur.

107. Exception conditions

108. Debugger

- Visual studio debugger

109. The interactive Console

- "view" → "other windows" → "C# interactive"

110. Summary of error handling

- debugger.
- collections

Section 10 - Step Ten - Futher Adventure in Programming

111. What is in Step 10?

- collections, dictionaries = list of object
- write complicated project made up of many classes and code files. - text based adventure game.

112. Generic Collections

- public List<Thing> thingList = new List<Thing>();
  - .Count - .Add - .name - .description

113. Dictionaries

- public Dictionary <string, Room>
  room Dictionary = new Dictionary <string, Rom>();
- keyValuePair <string, Room>

Dic: - .Remove(x1) - ContainsKey(xx)

List: - .RemoveAt( )

114. Overridden Methods

- override - still can run method in for each loop
- new - can't run in foreach loop.

115. Write an advanture game

116. The game user interface

- WPF

117. A brief history of adventure game

- Zalk
- Turbo Pascal
- Colossal Cave Adventure

118. Playing the game

119. The game classes

- classes -
- virtual.

120. Saving and restoring games

- BinaryFormatter.

121. Subclassing Dictionaries

122. Serializing Dictionaries

```
// Serializable in Dictionary
protected RoomList(SerializationInfo info,
StreamingContext context)
: base (info, context) {
}
```

123. Summary of Collections