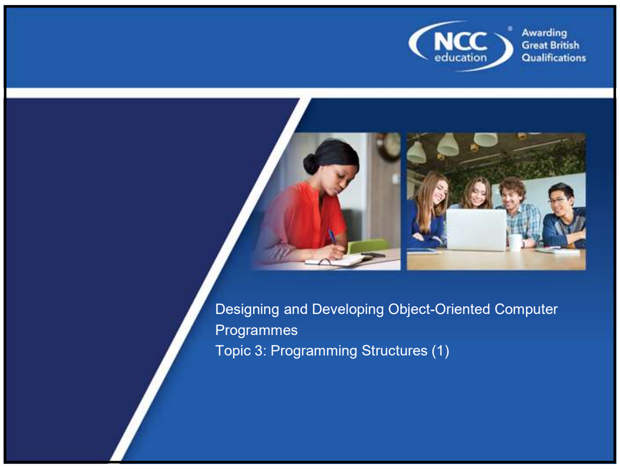
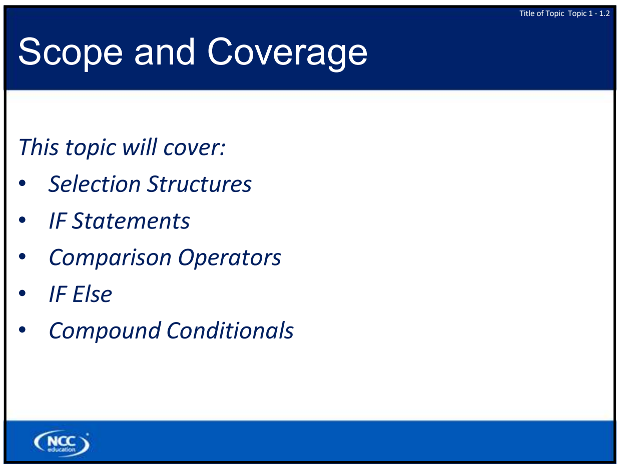


Designing and Developing Object-Oriented Computer Programmes
Topic 3: Programming Structures (1)

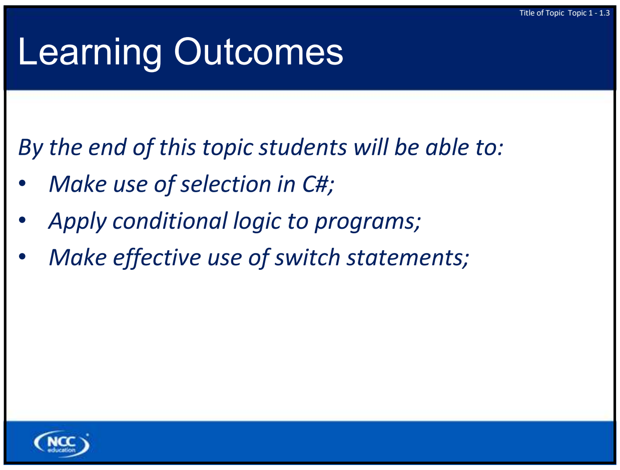
1


Title of Topic Topic 1 - 1.2

## Scope and Coverage

*This topic will cover:*
- *Selection Structures*
- *IF Statements*
- *Comparison Operators*
- *IF Else*
- *Compound Conditionals*

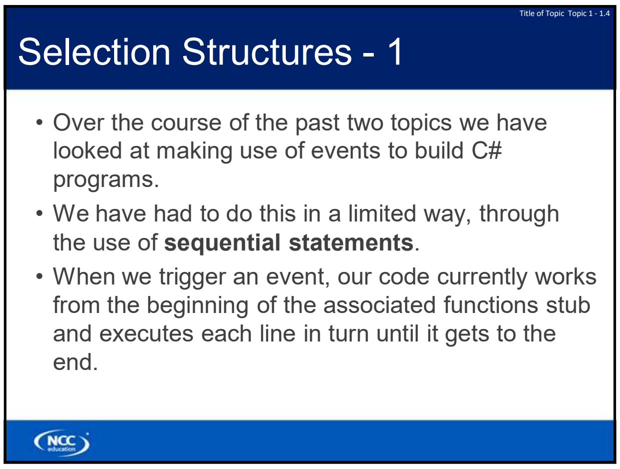
2


Title of Topic Topic 1 - 1.3

## Learning Outcomes

*By the end of this topic students will be able to:*
- *Make use of selection in C#;*
- *Apply conditional logic to programs;*
- *Make effective use of switch statements;*

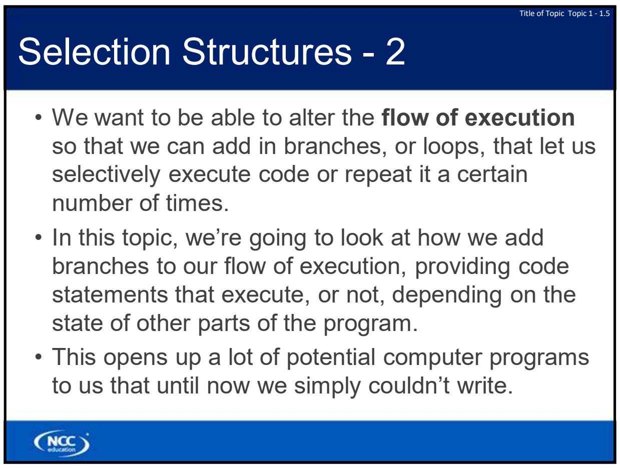
3



## Selection Structures - 1

- Over the course of the past two topics we have looked at making use of events to build C# programs.
- We have had to do this in a limited way, through the use of **sequential statements**.
- When we trigger an event, our code currently works from the beginning of the associated functions stub and executes each line in turn until it gets to the end.

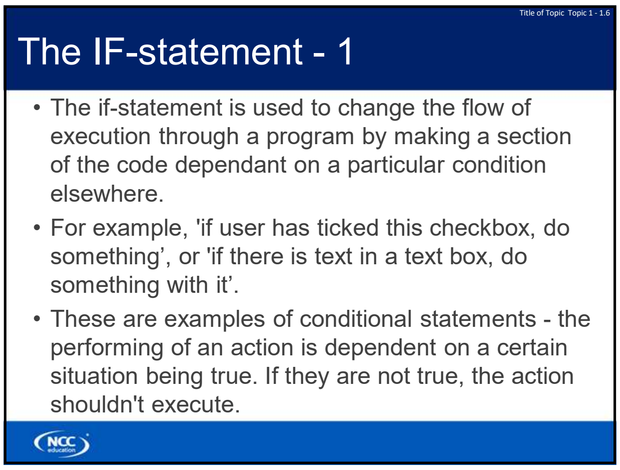
4



## Selection Structures - 2

- We want to be able to alter the **flow of execution** so that we can add in branches, or loops, that let us selectively execute code or repeat it a certain number of times.
- In this topic, we're going to look at how we add branches to our flow of execution, providing code statements that execute, or not, depending on the state of other parts of the program.
- This opens up a lot of potential computer programs to us that until now we simply couldn't write.

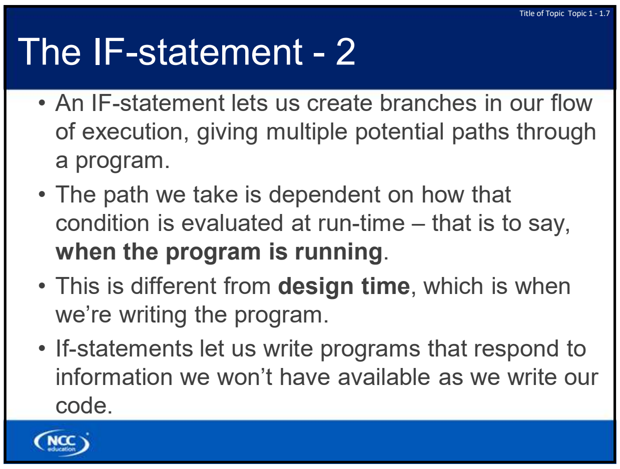
5



## The IF-statement - 1

- The if-statement is used to change the flow of execution through a program by making a section of the code dependant on a particular condition elsewhere.
- For example, 'if user has ticked this checkbox, do something', or 'if there is text in a text box, do something with it'.
- These are examples of conditional statements - the performing of an action is dependent on a certain situation being true. If they are not true, the action shouldn't execute.

6

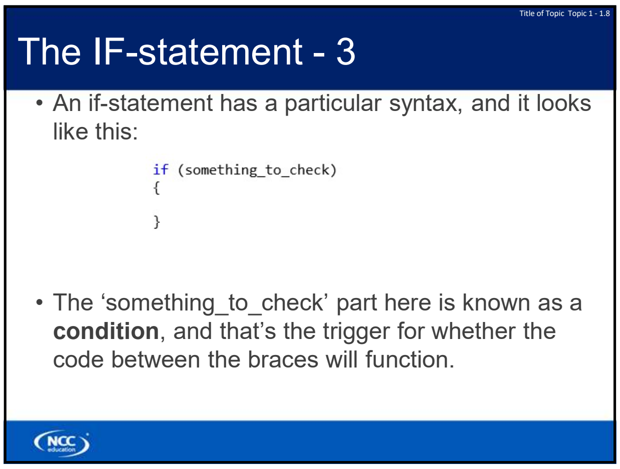


Title of Topic Topic 1 - 1.7

## The IF-statement - 2

- An IF-statement lets us create branches in our flow of execution, giving multiple potential paths through a program.
- The path we take is dependent on how that condition is evaluated at run-time – that is to say, **when the program is running**.
- This is different from **design time**, which is when we're writing the program.
- If-statements let us write programs that respond to information we won't have available as we write our code.

7



Title of Topic Topic 1 - 1.8

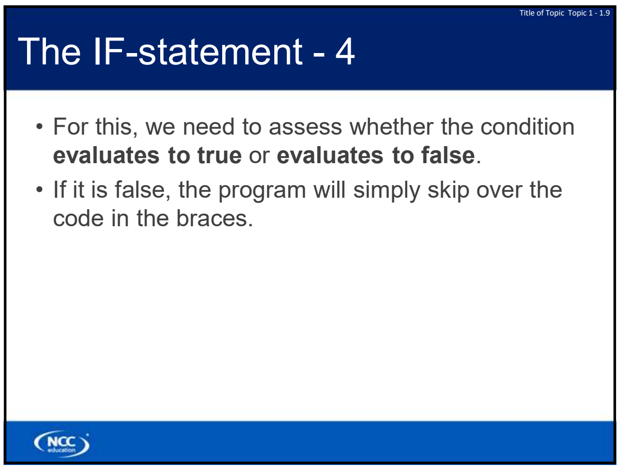
## The IF-statement - 3

- An if-statement has a particular syntax, and it looks like this:

```
if (something_to_check)
{

}
```

- The 'something_to_check' part here is known as a **condition**, and that's the trigger for whether the code between the braces will function.

8

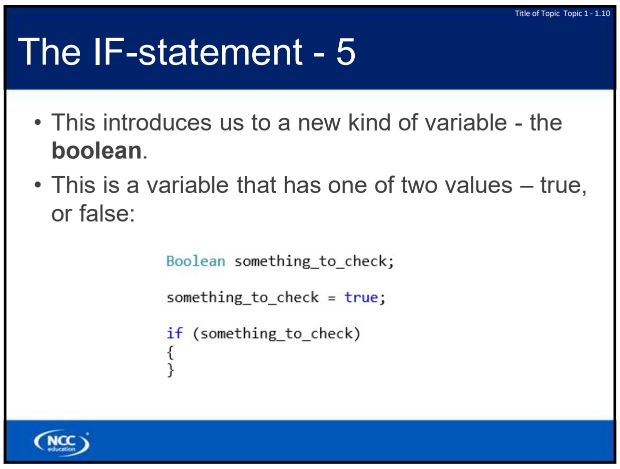


Title of Topic Topic 1 - 1.9

## The IF-statement - 4

- For this, we need to assess whether the condition **evaluates to true** or **evaluates to false**.
- If it is false, the program will simply skip over the code in the braces.

9



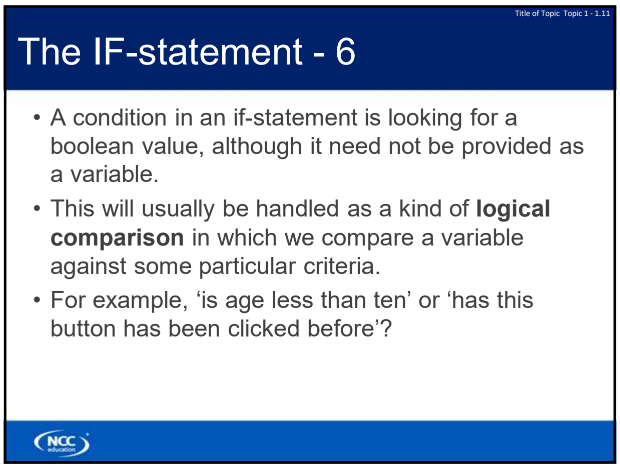
Title of Topic Topic 1 - 1.10

## The IF-statement - 5

- This introduces us to a new kind of variable - the **boolean**.
- This is a variable that has one of two values – true, or false:

```
Boolean something_to_check;

something_to_check = true;

if (something_to_check)
{
}
```

10

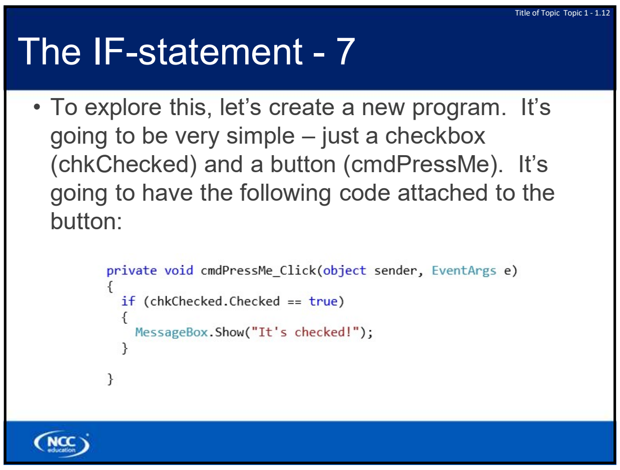


Title of Topic Topic 1 - 1.11

## The IF-statement - 6

- A condition in an if-statement is looking for a boolean value, although it need not be provided as a variable.
- This will usually be handled as a kind of **logical comparison** in which we compare a variable against some particular criteria.
- For example, 'is age less than ten' or 'has this button has been clicked before'?

11



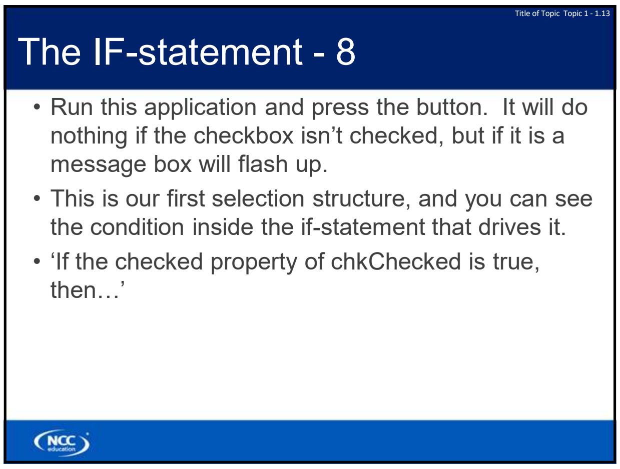
Title of Topic Topic 1 - 1.12

## The IF-statement - 7

- To explore this, let's create a new program. It's going to be very simple – just a checkbox (chkChecked) and a button (cmdPressMe). It's going to have the following code attached to the button:

```
private void cmdPressMe_Click(object sender, EventArgs e)
{
  if (chkChecked.Checked == true)
  {
    MessageBox.Show("It's checked!");
  }

}
```

12

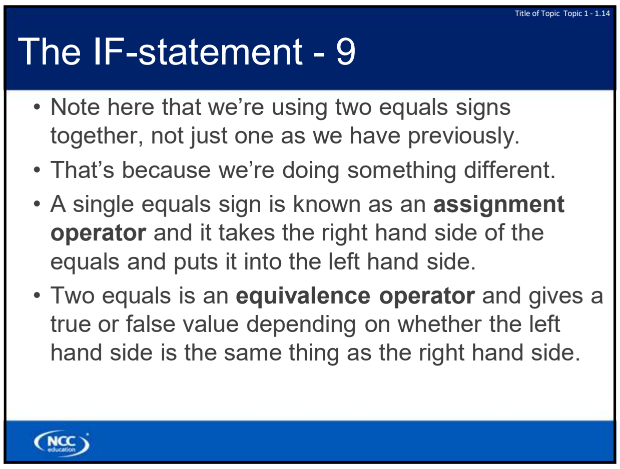


Title of Topic Topic 1 - 1.13

## The IF-statement - 8

- Run this application and press the button. It will do nothing if the checkbox isn't checked, but if it is a message box will flash up.
- This is our first selection structure, and you can see the condition inside the if-statement that drives it.
- 'If the checked property of chkChecked is true, then…'

13

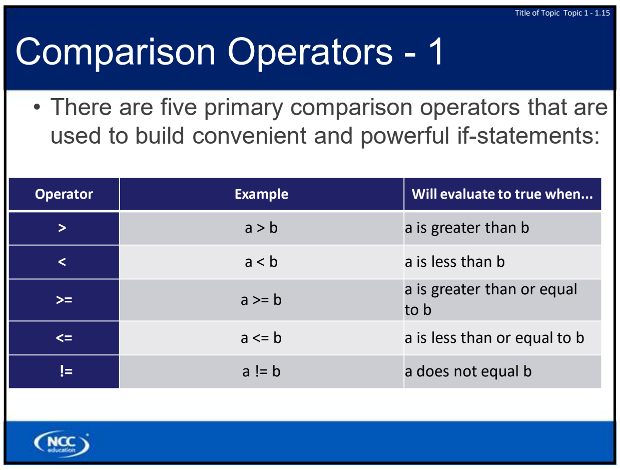


Title of Topic Topic 1 - 1.14

## The IF-statement - 9

- Note here that we're using two equals signs together, not just one as we have previously.
- That's because we're doing something different.
- A single equals sign is known as an **assignment operator** and it takes the right hand side of the equals and puts it into the left hand side.
- Two equals is an **equivalence operator** and gives a true or false value depending on whether the left hand side is the same thing as the right hand side.

14



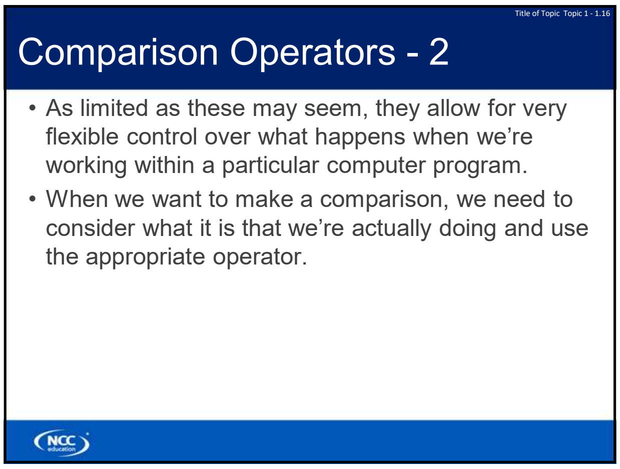
Title of Topic Topic 1 - 1.15

## Comparison Operators - 1

- There are five primary comparison operators that are used to build convenient and powerful if-statements:

| Operator | Example | Will evaluate to true when… |
|---|---|---|
| > | a > b | a is greater than b |
| < | a < b | a is less than b |
| >= | a >= b | a is greater than or equal to b |
| <= | a <= b | a is less than or equal to b |
| != | a != b | a does not equal b |

15

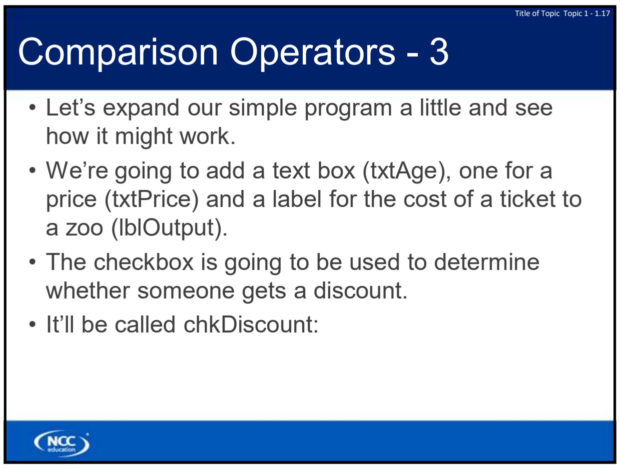


Title of Topic Topic 1 - 1.16

## Comparison Operators - 2

- As limited as these may seem, they allow for very flexible control over what happens when we're working within a particular computer program.
- When we want to make a comparison, we need to consider what it is that we're actually doing and use the appropriate operator.

16

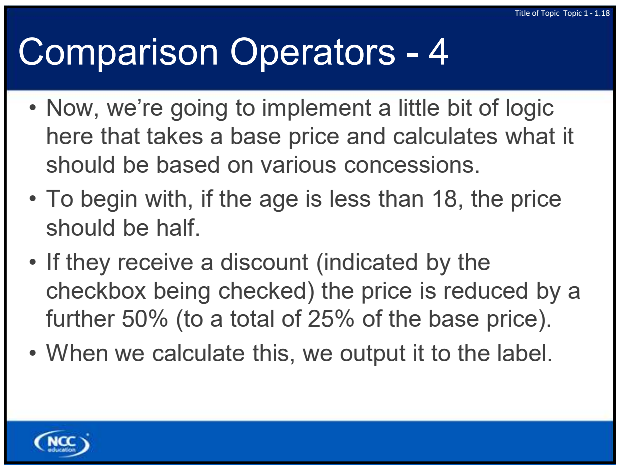


Title of Topic Topic 1 - 1.17

## Comparison Operators - 3

- Let's expand our simple program a little and see how it might work.
- We're going to add a text box (txtAge), one for a price (txtPrice) and a label for the cost of a ticket to a zoo (lblOutput).
- The checkbox is going to be used to determine whether someone gets a discount.
- It'll be called chkDiscount:

17

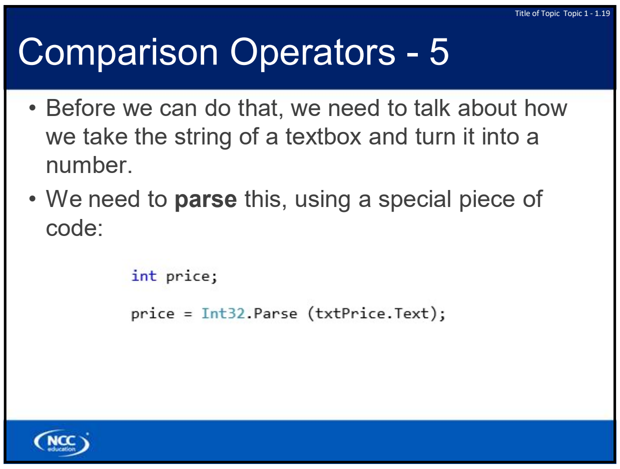


Title of Topic Topic 1 - 1.18

## Comparison Operators - 4

- Now, we're going to implement a little bit of logic here that takes a base price and calculates what it should be based on various concessions.
- To begin with, if the age is less than 18, the price should be half.
- If they receive a discount (indicated by the checkbox being checked) the price is reduced by a further 50% (to a total of 25% of the base price).
- When we calculate this, we output it to the label.

18



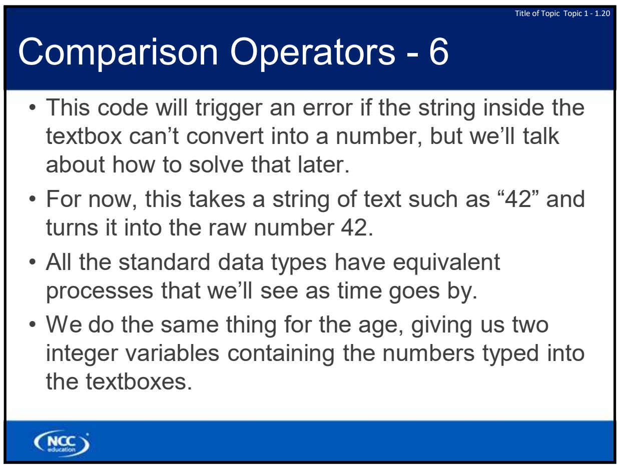
Title of Topic Topic 1 - 1.19

## Comparison Operators - 5

- Before we can do that, we need to talk about how we take the string of a textbox and turn it into a number.
- We need to **parse** this, using a special piece of code:

```
int price;

price = Int32.Parse (txtPrice.Text);
```

19

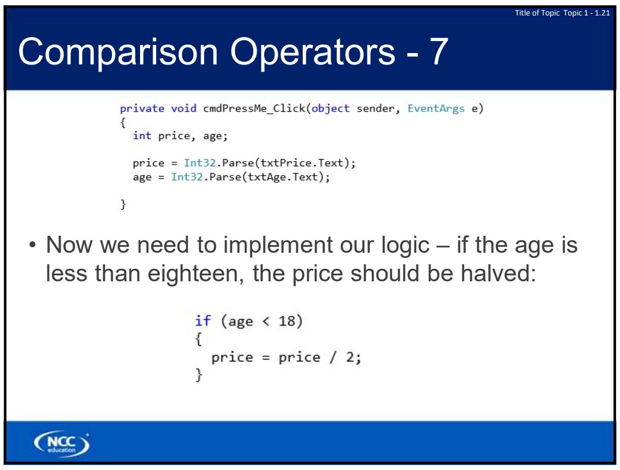


Title of Topic Topic 1 - 1.20

## Comparison Operators - 6

- This code will trigger an error if the string inside the textbox can't convert into a number, but we'll talk about how to solve that later.
- For now, this takes a string of text such as "42" and turns it into the raw number 42.
- All the standard data types have equivalent processes that we'll see as time goes by.
- We do the same thing for the age, giving us two integer variables containing the numbers typed into the textboxes.

20



Title of Topic Topic 1 - 1.21

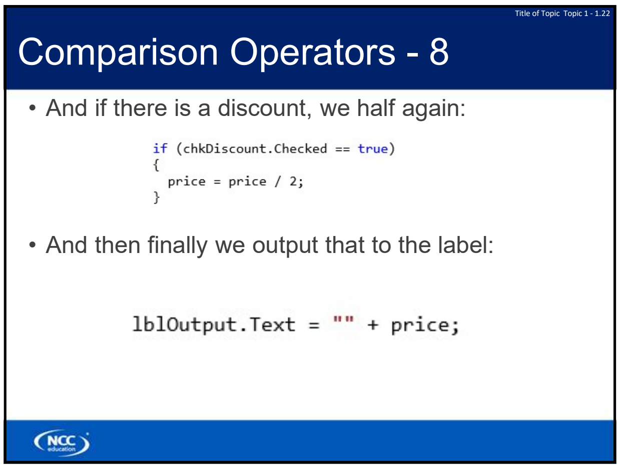
## Comparison Operators - 7

```
private void cmdPressMe_Click(object sender, EventArgs e)
{
  int price, age;

  price = Int32.Parse(txtPrice.Text);
  age = Int32.Parse(txtAge.Text);

}
```

- Now we need to implement our logic – if the age is less than eighteen, the price should be halved:

```
if (age < 18)
{
  price = price / 2;
}
```

21



## Comparison Operators - 8

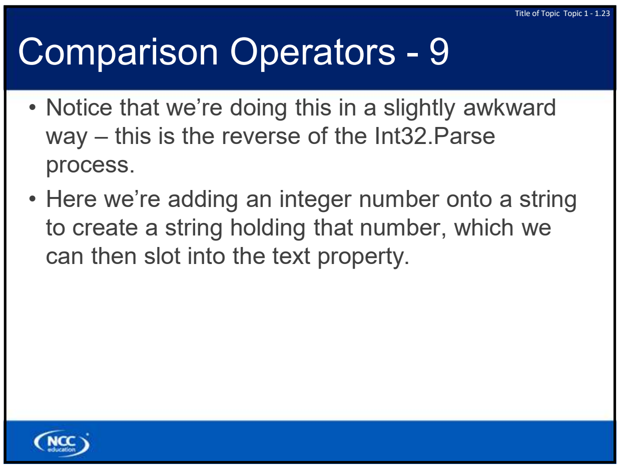
- And if there is a discount, we half again:

```
if (chkDiscount.Checked == true)
{
  price = price / 2;
}
```

- And then finally we output that to the label:

```
lblOutput.Text = "" + price;
```

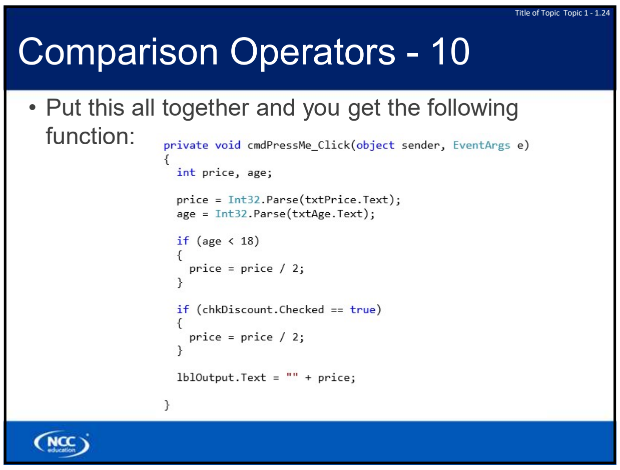
22



## Comparison Operators - 9

- Notice that we're doing this in a slightly awkward way – this is the reverse of the Int32.Parse process.
- Here we're adding an integer number onto a string to create a string holding that number, which we can then slot into the text property.

23

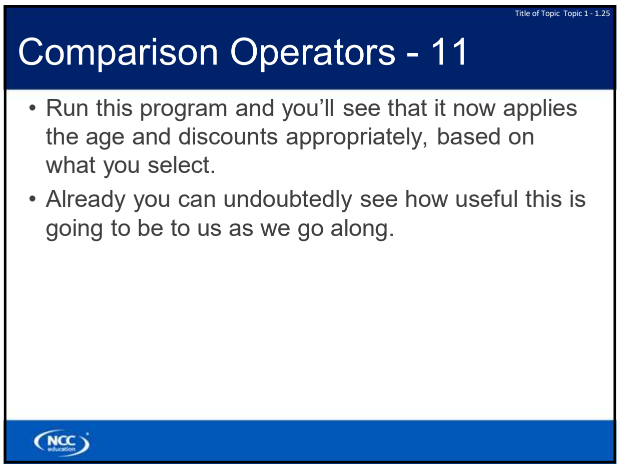


## Comparison Operators - 10

- Put this all together and you get the following function:

```
private void cmdPressMe_Click(object sender, EventArgs e)
{
  int price, age;

  price = Int32.Parse(txtPrice.Text);
  age = Int32.Parse(txtAge.Text);

  if (age < 18)
  {
    price = price / 2;
  }

  if (chkDiscount.Checked == true)
  {
    price = price / 2;
  }

  lblOutput.Text = "" + price;

}
```

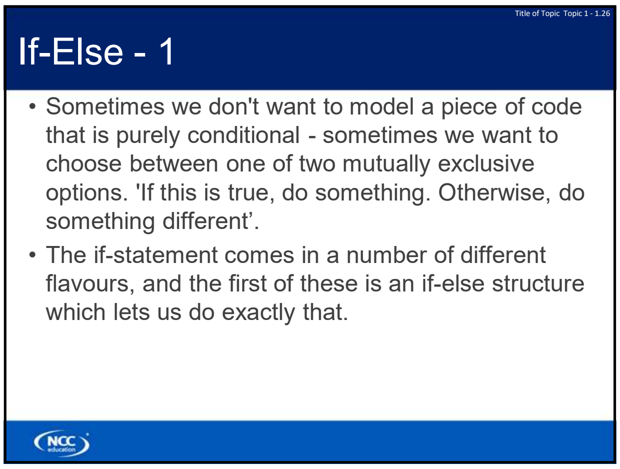
24



## Comparison Operators - 11

- Run this program and you'll see that it now applies the age and discounts appropriately, based on what you select.
- Already you can undoubtedly see how useful this is going to be to us as we go along.

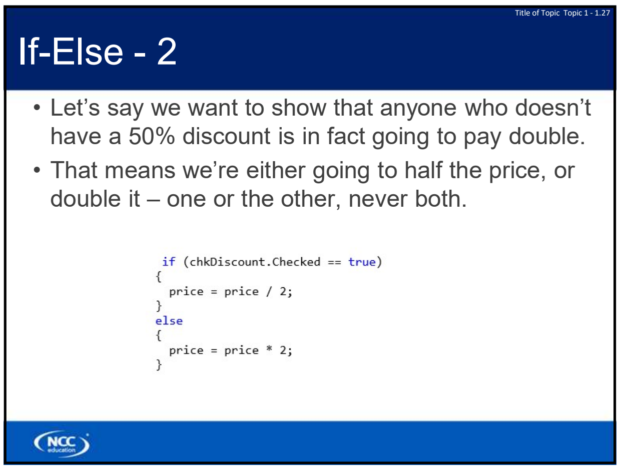
25



## If-Else - 1

- Sometimes we don't want to model a piece of code that is purely conditional - sometimes we want to choose between one of two mutually exclusive options. 'If this is true, do something. Otherwise, do something different'.
- The if-statement comes in a number of different flavours, and the first of these is an if-else structure which lets us do exactly that.

26

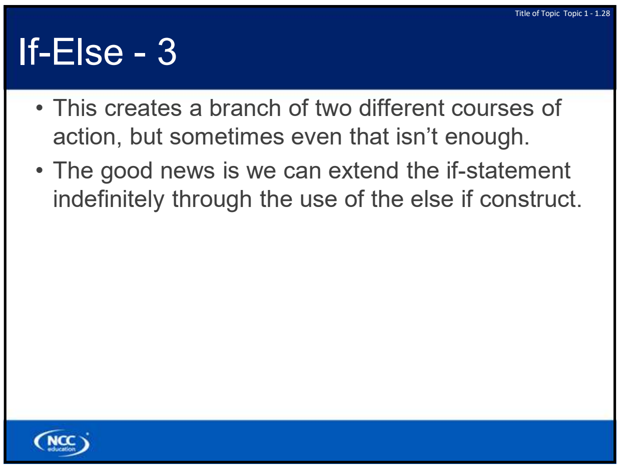


## If-Else - 2

- Let's say we want to show that anyone who doesn't have a 50% discount is in fact going to pay double.
- That means we're either going to half the price, or double it – one or the other, never both.

```
 if (chkDiscount.Checked == true)
{
  price = price / 2;
}
else
{
  price = price * 2;
}
```

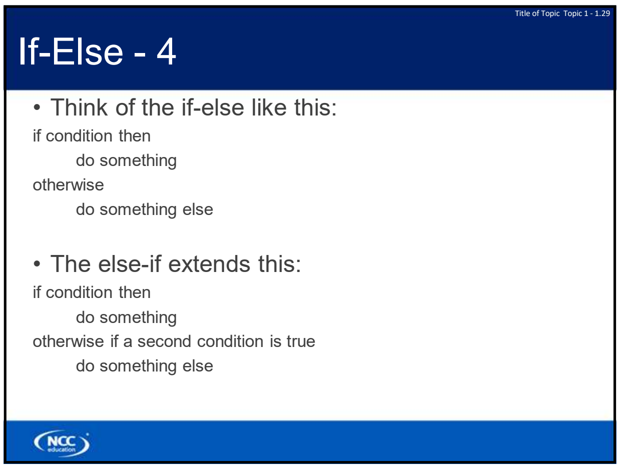
27


Title of Topic Topic 1 - 1.28

## If-Else - 3

- This creates a branch of two different courses of action, but sometimes even that isn't enough.
- The good news is we can extend the if-statement indefinitely through the use of the else if construct.

28


Title of Topic Topic 1 - 1.29

## If-Else - 4

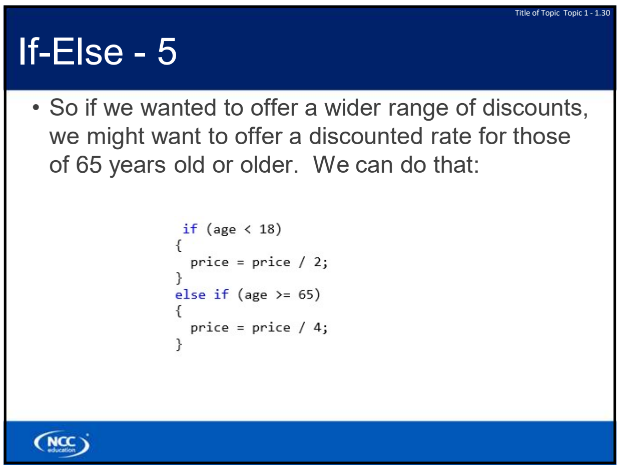
- Think of the if-else like this:

```
if condition then
        do something
otherwise
        do something else
```

- The else-if extends this:

```
if condition then
        do something
otherwise if a second condition is true
        do something else
```

29

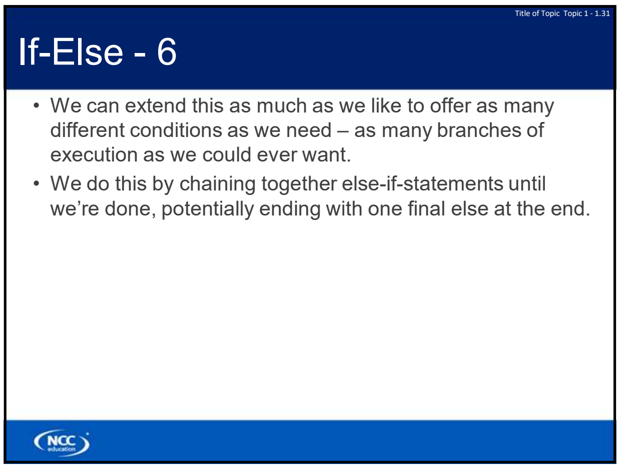

Title of Topic Topic 1 - 1.30

## If-Else - 5

- So if we wanted to offer a wider range of discounts, we might want to offer a discounted rate for those of 65 years old or older. We can do that:

```
 if (age < 18)
{
  price = price / 2;
}
else if (age >= 65)
{
  price = price / 4;
}
```

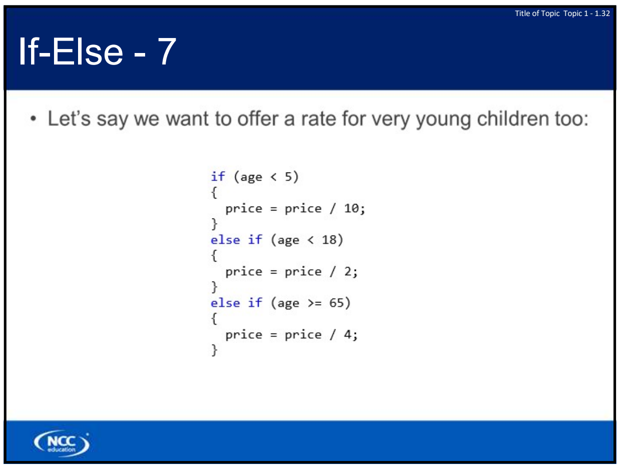
30


Title of Topic Topic 1 - 1.31

## If-Else - 6

- We can extend this as much as we like to offer as many different conditions as we need – as many branches of execution as we could ever want.
- We do this by chaining together else-if-statements until we're done, potentially ending with one final else at the end.

31

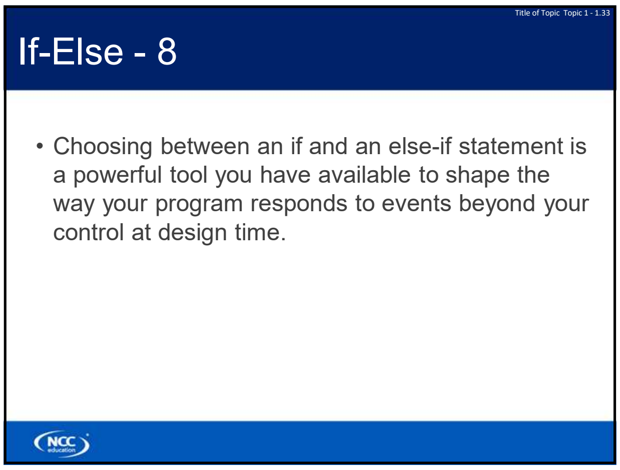

Title of Topic Topic 1 - 1.32

## If-Else - 7

- Let's say we want to offer a rate for very young children too:

```
if (age < 5)
{
  price = price / 10;
}
else if (age < 18)
{
  price = price / 2;
}
else if (age >= 65)
{
  price = price / 4;
}
```

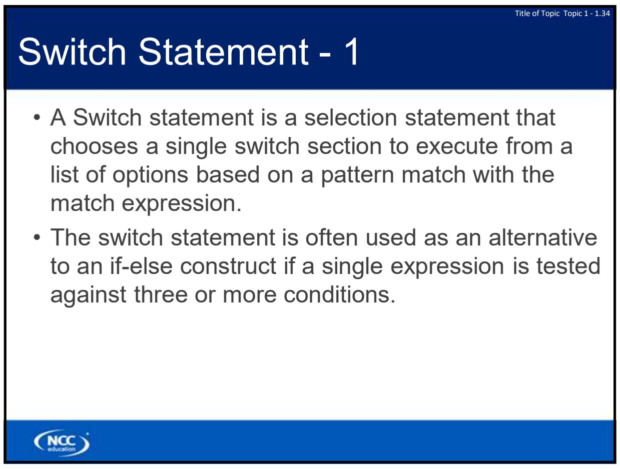
32


Title of Topic Topic 1 - 1.33

## If-Else - 8

- Choosing between an if and an else-if statement is a powerful tool you have available to shape the way your program responds to events beyond your control at design time.

33

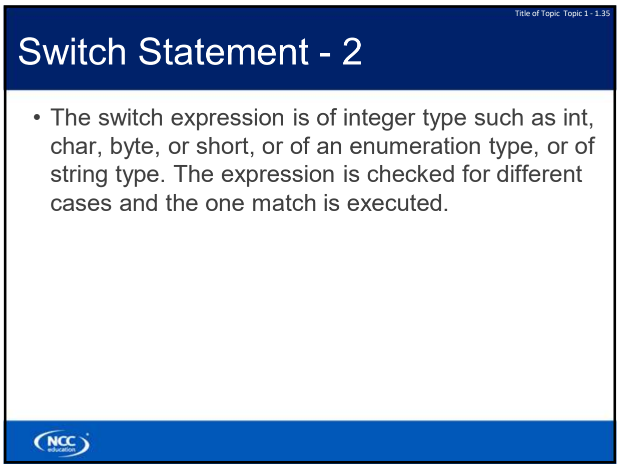


Title of Topic Topic 1 - 1.34

## Switch Statement - 1

- A Switch statement is a selection statement that chooses a single switch section to execute from a list of options based on a pattern match with the match expression.
- The switch statement is often used as an alternative to an if-else construct if a single expression is tested against three or more conditions.

34

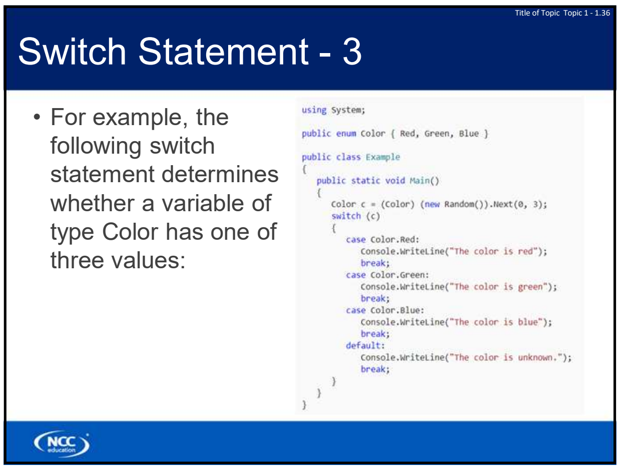


Title of Topic Topic 1 - 1.35

## Switch Statement - 2

- The switch expression is of integer type such as int, char, byte, or short, or of an enumeration type, or of string type. The expression is checked for different cases and the one match is executed.

35



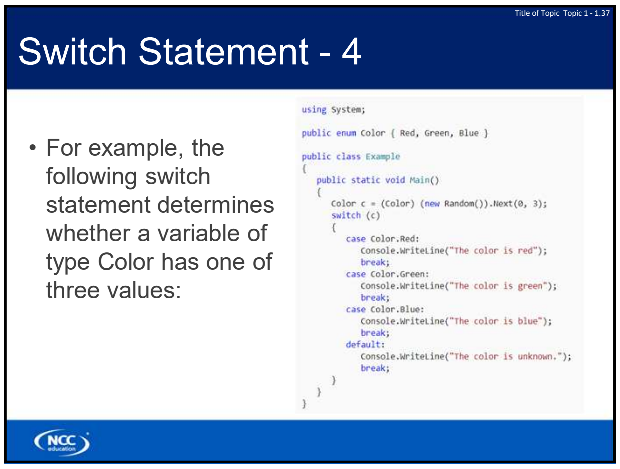
Title of Topic Topic 1 - 1.36

## Switch Statement - 3

- For example, the following switch statement determines whether a variable of type Color has one of three values:

```
using System;

public enum Color { Red, Green, Blue }

public class Example
{
   public static void Main()
   {
      Color c = (Color) (new Random()).Next(0, 3);
      switch (c)
      {
         case Color.Red:
            Console.WriteLine("The color is red");
            break;
         case Color.Green:
            Console.WriteLine("The color is green");
            break;
         case Color.Blue:
            Console.WriteLine("The color is blue");
            break;
         default:
            Console.WriteLine("The color is unknown.");
            break;
      }
   }
}
```

36

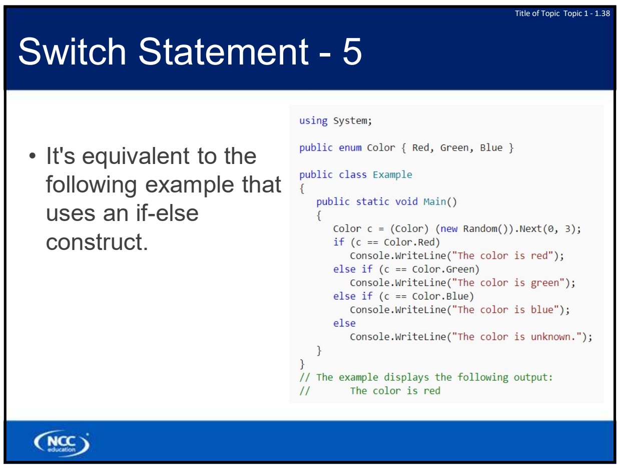


## Switch Statement - 4

- For example, the following switch statement determines whether a variable of type Color has one of three values:

```
using System;

public enum Color { Red, Green, Blue }

public class Example
{
   public static void Main()
   {
      Color c = (Color) (new Random()).Next(0, 3);
      switch (c)
      {
         case Color.Red:
            Console.WriteLine("The color is red");
            break;
         case Color.Green:
            Console.WriteLine("The color is green");
            break;
         case Color.Blue:
            Console.WriteLine("The color is blue");
            break;
         default:
            Console.WriteLine("The color is unknown.");
            break;
      }
   }
}
```

37

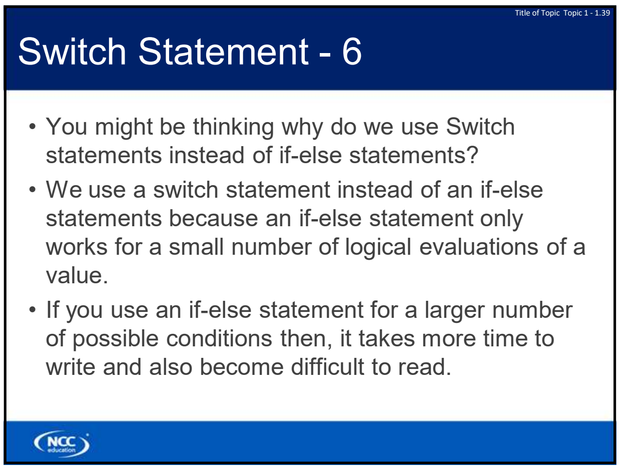


## Switch Statement - 5

- It's equivalent to the following example that uses an if-else construct.

```
using System;

public enum Color { Red, Green, Blue }

public class Example
{
   public static void Main()
   {
      Color c = (Color) (new Random()).Next(0, 3);
      if (c == Color.Red)
         Console.WriteLine("The color is red");
      else if (c == Color.Green)
         Console.WriteLine("The color is green");
      else if (c == Color.Blue)
         Console.WriteLine("The color is blue");
      else
         Console.WriteLine("The color is unknown.");
   }
}
// The example displays the following output:
//       The color is red
```

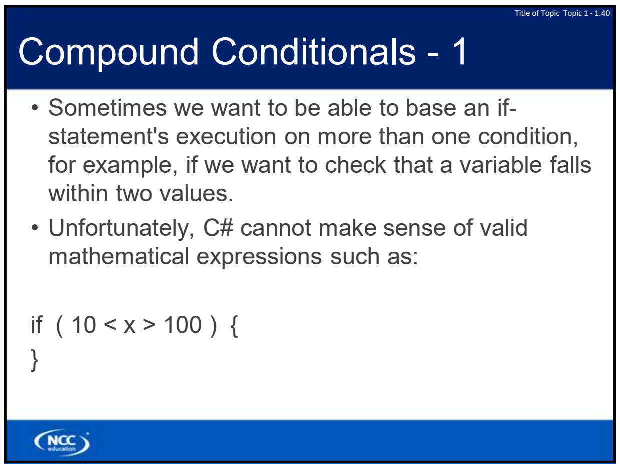
38



## Switch Statement - 6

- You might be thinking why do we use Switch statements instead of if-else statements?
- We use a switch statement instead of an if-else statements because an if-else statement only works for a small number of logical evaluations of a value.
- If you use an if-else statement for a larger number of possible conditions then, it takes more time to write and also become difficult to read.

39

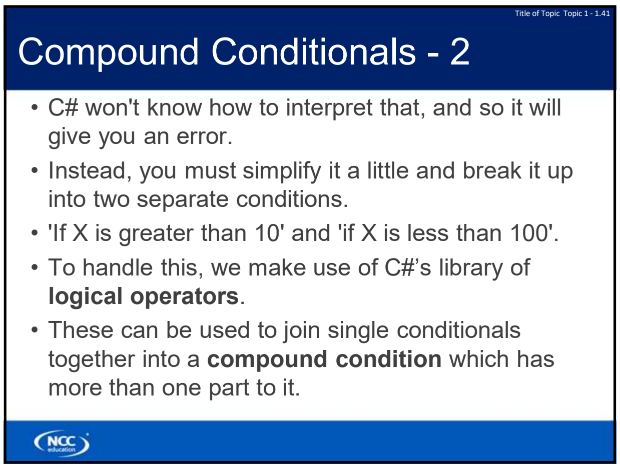


## Compound Conditionals - 1

- Sometimes we want to be able to base an if-statement's execution on more than one condition, for example, if we want to check that a variable falls within two values.
- Unfortunately, C# cannot make sense of valid mathematical expressions such as:

```
if ( 10 < x > 100 ) {
}
```

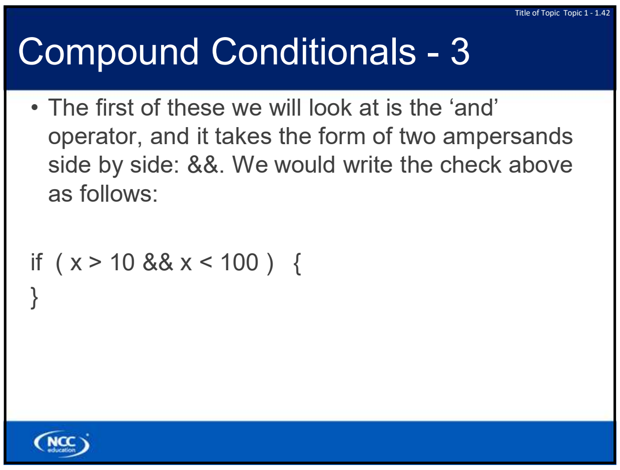
40



## Compound Conditionals - 2

- C# won't know how to interpret that, and so it will give you an error.
- Instead, you must simplify it a little and break it up into two separate conditions.
- 'If X is greater than 10' and 'if X is less than 100'.
- To handle this, we make use of C#'s library of **logical operators**.
- These can be used to join single conditionals together into a **compound condition** which has more than one part to it.

41

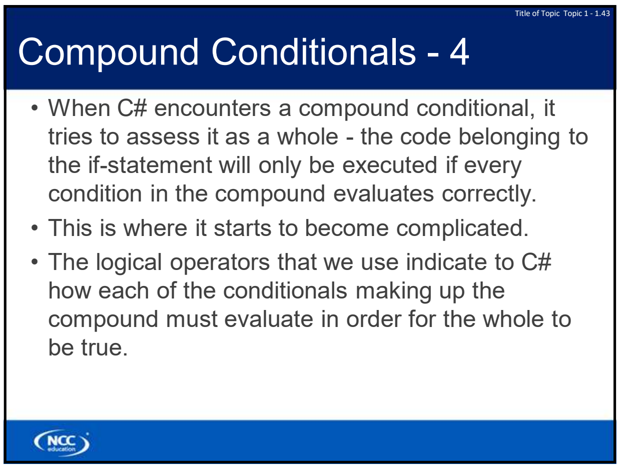


## Compound Conditionals - 3

- The first of these we will look at is the ‘and’ operator, and it takes the form of two ampersands side by side: &&. We would write the check above as follows:

```
if ( x > 10 && x < 100 )  {
}
```

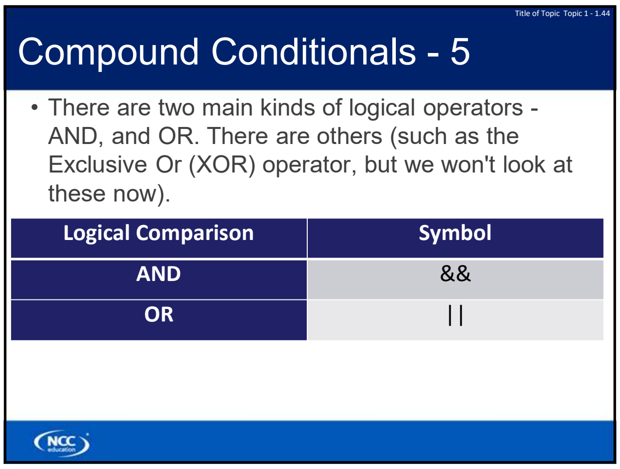
42



## Compound Conditionals - 4

- When C# encounters a compound conditional, it tries to assess it as a whole - the code belonging to the if-statement will only be executed if every condition in the compound evaluates correctly.
- This is where it starts to become complicated.
- The logical operators that we use indicate to C# how each of the conditionals making up the compound must evaluate in order for the whole to be true.

43

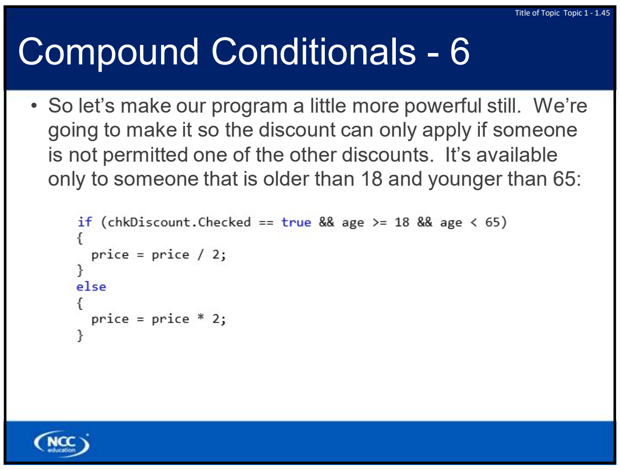


## Compound Conditionals - 5

- There are two main kinds of logical operators - AND, and OR. There are others (such as the Exclusive Or (XOR) operator, but we won't look at these now).

| Logical Comparison | Symbol |
|---|---|
| AND | && |
| OR | \|\| |

44

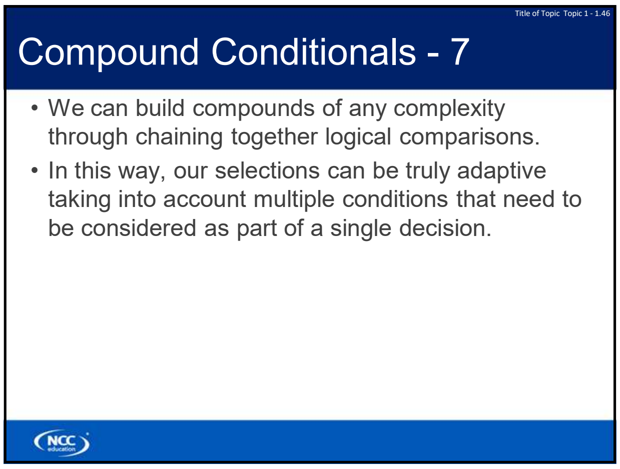


## Compound Conditionals - 6

- So let's make our program a little more powerful still. We're going to make it so the discount can only apply if someone is not permitted one of the other discounts. It's available only to someone that is older than 18 and younger than 65:

```
if (chkDiscount.Checked == true && age >= 18 && age < 65)
{
  price = price / 2;
}
else
{
  price = price * 2;
}
```

45

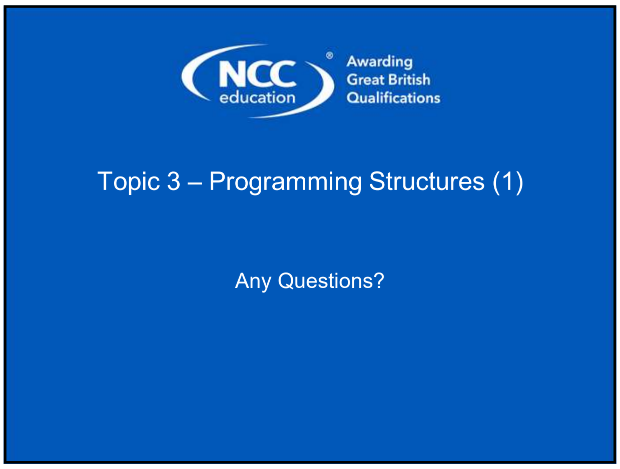


Title of Topic Topic 1 - 1.46

## Compound Conditionals - 7

- We can build compounds of any complexity through chaining together logical comparisons.
- In this way, our selections can be truly adaptive taking into account multiple conditions that need to be considered as part of a single decision.

46



Awarding Great British Qualifications

Topic 3 – Programming Structures (1)

Any Questions?

47