Inheritance

Inheritance extends a class and supports a baseclass-subclass relationship between classes. This is also referred to as a parent-child class relationship or a super-derived class relationship or an ancestor-descendant class relationship. Every subclass extends its base class.

Inheritance:

- Can be described by “is a” relationship
- Provides a structural hierarchy showing a relationship between the classes
- Emphasises the similarities and differences between classes,
- Eliminates repetitious code in subclasses/encourages code reuse.

example 1

Breakout

1. Using the Breakout code, look for similarities and differences between the Ball, Paddle and Brick classes.
2. Make a superclass called GameObject and descend the Ball, Paddle and Brick subclasses.
3. Put any common fields and methods into GameObject superclass.
4. Put anything that is different into the individual subclasses.
5. Note the reuse of code – the subclasses can use the code written in the superclass.

---

example 2

As an employer, you have some full-time employees and some part-time employees. Create an inheritance hierarchy with an AllEmployee parent class and with FullTimeEmployee and PartTimeEmployee child classes. What would be some fields and methods for each of these classes?

---

example 3

Suppose you want to build a progam that simulates a zoo. You need to accommodate 6 animals: a lion, a hippo, a tiger, a dog, a cat and a wolf.
You could build a class for each animal and have a lot of duplicate code.
Instead start by building an Animal ancestor class that holds all the fields and methods that are common to all of our six animals.
Next create subclasses that inherit all the fields and methods of the ancestor and contain anything that is different from the ancestor class.

---

example 4

Using the library application from Class 18, we could extend the Book class by adding subclass EBook. This new class will inherit all the fields and methods of Book plus have additional fields, properties and methods of its own. For example, EBook may have an additional string field called format.

To show inheritance, we would write the EBook class declaration as:

```
public class EBook : Book
{
     private string format;
     ……
```

The EBook constructor passes the parameters for the base class fields to the Book class for initialisation. Control is then returned to the EBook Constructor and any new fields are then initialized.

In the EBook.cs file:

```
public EBook(int iD, string title, string format)
     : base(iD, title)
 {
       this.format = format;
 }
```

In the Book.cs file:

```
public Book(int iD, string title)
 {
       this.iD = iD;
       this.title = title;
       issued = false;
       borrower = “”;
 }
```

---

example 5

All vertebrates have legs and can eat. A vertebrate can eat a quantity of a given food.
A bird is a vertebrate, but also has a wingspan and can fly.

Set up a Vertebrate superclass and descend a Bird subclass from it.
Create a tweety object that is a Bird, has 2 legs and a wingspan of 7.5 metres. Make tweety eat 5 sunflower seeds and fly.
Create a barney object that is a Vertebrate and has 4 legs. Make barney eat 4 mosquitoes.

---

example 6

Spot the errors:

1. Order a BLT when you click the button.

```
public class Sandwich
{
     protected bool toasted;
     protected int slicesOfBread;
     ....

     public int CountCalories()
     {
          ......
     }
}
```

```
public class BLT : Sandwich
{
      private int slicesOfBacon;
      private int amountOfLettuce;
      .....

      public int AddASideOfFries()
      {
           ......
      }
}

private void button1_Click(object sender, EventArgs e)
{
      BLT myBLT = new BLT();
      BLT.Toasted = true;
      Sandwich.SlicesOfBread = 3;
      myBLT.AddASideOfFries();
      myBLT.SlicesOfBacon += 5;
      MessageBox.Show("My sandwich has " + myBLT.CountCalories + "calories.");
      return myBLT;
}
```

2. Send a fire fighting plane on fire fighting mission.

```
public class Aircraft
{
      protected double airSpeed;
      protected double altitude;

      ....
      public void TakeOff()
      {
           ......
      }

      public void Land()
      {
           ......
      }
}

public class FirePlane : Aircraft
{
      private double bucketCapacity;

      .....

      public void FillBucket()
      {
           ......
      }
}

private void fireFightingMission()
{
      FirePlane myFirePlane = new FirePlane();
      new FirePlane.BucketCapacity = 500;
      Aircraft.Altitude = 0;
      myFirePlane.TakeOff();
      myFirePlane.AirSpeed = 192.5;
      myFirePlane.FillBucket();
      Aircraft.Land();
}
```

example 7

Watch:

What is object-oriented language? https://www.youtube.com/watch?v=SS-9y0H3Si8
Identifying an inheritance situation? https://www.youtube.com/watch?v=oZcLmje8-fg

---