UNIVERSITY OF Hull

Static Members and Design

Rob Miles

Department of Computer Science

3b 08120 Programming 2

Static Class Members

```
class Test {
    public static void Main ()
    {
    }
}
```

- We have seen static many times in our programs
  - The `Main` method must be declared static
- Now we find out what it means

The Meaning of Static

- Static means "part of the class"
  - Since the class is always there, static therefore also means "always there"
- When a program starts running there are no instances of any classes in existence
- This means that the `Main` method must be static, otherwise it would not be there to call

Making Classes

- A class contains members
  - Data fields and methods
- The class file itself just tells the runtime system how to make a class instance

"An object is an instance of a class"

- The object is only created when we make the instance

Static Class Members

```
Account a; // declare the reference
a = new Account(); // make the object
a.PayInFunds(50);  // use the object
```

- The `PayInFunds` method is not static
- We need to have an instance of the class to call it on
- This is sensible, because we only want to pay money into an existing account

Making a Static method

- Sometimes we might want to do something with an account without actually having created one
- We only allow accounts for people who have a particular income and age
- We need to use this method before we create the account
- If it returns false we don't allow the account to be created

The Allowed Method

```
public bool Allowed (decimal income, int age)
{
    if ( (income >= 10000) && (age >= 18) )
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

- You can have an account if you earn over 10,000 pounds and you are 18 or over years old
- This method takes the income and age of a potential account holder and returns true if they are allowed an account

Problems Using Allowed

- We can only call `Allowed` once we already have an account instance
- That is no good to us, since we want to use it to decide whether or not an account can be created
- By the time we could use the method we have already made the instance

Making Allowed Static

```
public static bool Allowed (decimal income, int age)
{
    if ( (income >= 10000) && (age >= 18) )
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

- If we make the method `static` it is now part of the `Account` class, not a particular instance of the class
- We don't need an `Account` to call it

Using a static method

```
if ( Account.Allowed (income, age))
{
    Console.WriteLine ("Account Allowed");
}
else
{
    Console.WriteLine ("Not Allowed");
}
```

- A static method is called from the class
- It can't be called on an instance since there is no instance present

Static Methods

- Static methods also let a class provide utility methods for other classes
- The `Math` class provides a lot of static methods that can be used in programs
- Whenever you need to provide a behaviour without an enclosing object you should use a static method

Adjusting Allowed Age + Income

```
public static bool Allowed (decimal income, int age)
{
    if ( (income >= 10000) && (age >= 18) )
    {       return true;
    }
    else
    {
        return false;
    }
}
```

- At the moment the income and age values are hard-wired as 10,000 and 18 in the code
- I don't like this – I would much rather have variables that represent these values

Age and Income limits

```
public bool Allowed (decimal income, int age) {
  if ( (income >= minIncome) && (age >= minAge) ) {
      return true;
   }
   else {
      return false;
   }
}
```

- This version of **`Allowed`** uses two class members, **`minIncome`** and **`minAge`** to test against
- We can alter these to change the limits
  - If the bank decides to allow 17 year old account holders it is easy to change the age limits

`minIncome` and `minAge`

- These variables should be part of the **`Account`**
- But they can't be part of an **`Account`** instance
- They must exist when we have no accounts
- They must be made static too
- Static methods can only ever use static data members

Declaring Static Data Members

```
private static int minAge = 18;
private static int minIncome = 10000;
```

- These fields are part of the class, not an instance
- These members have been made private so that they are not visible outside the class
- If they were public they could be seen and changed by code running outside the class
- In this particular situation you do not want these values to be changed by other classes

Static in our Programs

```
class Program
{
      static void Main(string[] args)
      {
      }
}
```

- Up until now all the programs we have written have been in a single class data has been used directly by methods in that class
- Because we never make an instance of that class this means that all our data must be declared as a static member

Other uses for Static data

```
private static float interestRate = 0.05f;
```

- Whenever we need a value which is to be stored once for all class instances, we can use a static member
- This means that if we change it, all the class instances pick up the new value
- Changing the interest rate value above will change the interest value for all accounts

Static Summary

- Static means "always present"
  - Part of the class, not an instance
  - Accessed by the class name
- It does not mean "cannot be changed"
- Data members and methods can be made static
- Static methods can only make use of static member variables