C# 4.0
Parameters

Oliver Sturm

http://www.oliversturm.com

Outline

- **Optional Parameters**
  - Declaring methods with optional parameters
  - Calling methods with optional parameters
- **Named Parameters**
  - Calling methods with named parameters
- **Combining Named and Optional Parameters**
- **Overload Resolution**
- **Unexpected Behavior**

Optional Parameters – Declaration

```
void Method(int a = 20, string b = "Text") {
  // ...
}
```

- Parameters can have default values
- Optional parameters must always follow mandatory ones
- In IL, optional parameters are decorated with the attributes Optional and DefaultParameterValue

Optional Parameters – Calling Methods

```
void Method(int a = 20, string b = "Text") {
  // ...
}

Method(42, "The Answer");
Method(42);
Method( );
```

- The method can still be called just like before
- One or more of the optional parameters can be left out in a call
- For an optional parameter, the default value becomes the value of the local variable when no value is passed explicitly

Named Parameters

```
void Method(int a = 20, string b = "Text") {
  // ...
}

Method(42, b: "The Answer");
Method(a: 42, b: "The Answer");
Method(b: "The Answer", a: 42);
Method(b: "Different text");
```

- Names of parameters can be used in method calls
- Named and unnamed parameters can be combined
- The order of parameters can be different from the declaration
- In combination with Optional Parameters, parameters can be skipped

Overload Resolution

- General old-style resolution rules based on parameter types still apply
- Overloads matching the exact number of parameters given in a call are preferred
- Parameter names are used for resolution purposes, if necessary
- As usual, the existence of overloads doesn't lead to any warnings, so be careful!

Unexpected Behavior

```
public class Human {
  public virtual void Calculate(int x, int y) { }
}

public class Man : Human {
  public override void Calculate(int y, int x) { }
}
```

- When parameter names change in overridden methods, calls with named parameters can give unexpected results

Summary

- Optional parameter allow the definition of default values for parameters that aren't included in a method call
- Named parameters mean you can mention a parameter name explicitly when calling a method
- These features have been introduced mainly to support legacy APIs
- Recommendations:
  - Use optional parameters sparingly in newly created APIs
  - Consider using named parameters when calls are long and confusing, but be aware that this also points at unintuitive and structurally complex APIs

- The C# 4.0 Language Specification is the place to go for the technical detail on things like overload resolution

References

- C# 4.0 Language Specification: http://osturm.me/cs40spec