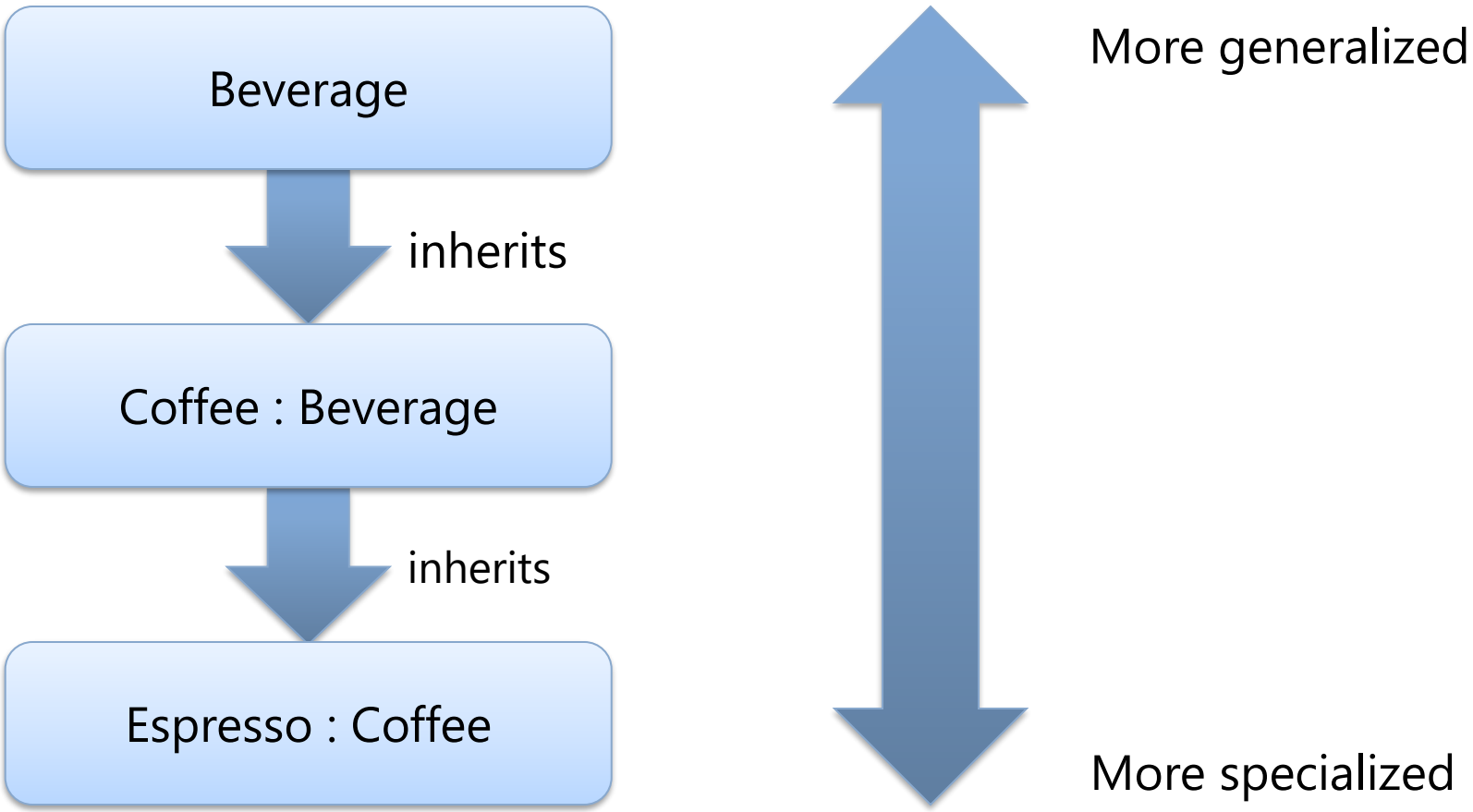
Creating a Class Hierarchy by Using Inheritance

What Is Inheritance?



The diagram shows a class hierarchy where a class named **Espresso** inherits from a class named **Coffee**, which in turn inherits from a class named **Beverage**. The inherited classes are increasingly specialized instances of the base class.

Creating Base Classes

- Use the **abstract** keyword to create a base class that cannot be instantiated

```
public abstract class Beverage
```

  - Create a class that derives from the abstract class
  - Implement any abstract members
- Use the **sealed** keyword to create a class that cannot be inherited

```
public sealed class Tea : Beverage
```

Creating Base Class Members

- Use the **virtual** keyword to create members that you can override in derived classes

```
public virtual int GetServingTemperature()
```

- Use the **protected** access modifier to make members available to derived types

```
protected int servingTemperature;
```

Inheriting from a Base Class

- To inherit from a base class, add the name of the base class to the class declaration

```
public class Coffee : Beverage
```

- To override virtual base class members, use the **override** keyword

```
public override int GetServingTemperature()
```

- To prevent classes further down the class hierarchy from overriding your override methods, use the **sealed** keyword

```
sealed public override int GetServingTemperature()
```

Calling Base Class Constructors and Members

- To call a base class constructor from a derived class, add the base constructor to your constructor declaration

```
public Coffee(string name, bool isFairTrade, int temp)
   : base(name, isFairTrade, servingTemp)
```

  - Pass parameter names to the base constructor as arguments
  - Do not use the base keyword within the constructor body
- To call base class methods from a derived class, use the base keyword like an instance variable

```
base.GetServingTemperature();
```

Abstract classes

- Can’t be instantiated
- Only function is: base class
- Also: abstract methods

Abstract classes

- Can’t be instantiated
- Only function is: base class
- Also: abstract methods

Sealed classes

- Can’t be a base class

Inheriting from Generic Types

For each base type parameter, you must either:

- Provide a type argument in your class declaration

```
public class CustomList : List<int>
```

- Include a matching type parameter in your class declaration

```
public class CustomList<T> : List<T>
```

Creating Extension Methods

- Create a static method in a static class
- Use the first parameter to indicate the type you want to extend
- Precede the first parameter with the **this** keyword

```
public static bool ContainsNumbers(this string s) {...}
```

- Call the method like a regular instance method

```
string text = "Text with numb3r5 ";
if(text.ContainsNumbers)
{
   // Do something.
}
```