

# References and Inheritance Solutions

- What is meant by a reference to a variable?
  - A reference is a variable which is an alias of another variable
  - It has its own name and type, but shares the memory object of the “bound” variable

- Is there anything unusual about references and derived class instances?
  - Normally we can only bind a reference to a variable of the same type
  - However, we can bind a base class reference to an instance of the derived class
- Does this also apply to pointers?
  - Yes, we can use a pointer to base class with an instance of the derived class

- What is meant by the terms "static type" and "dynamic type"? Give an example of each
  - The static type of a variable is the type given to it by the compiler
  - The dynamic type is the type of the variable's memory object
  - Usually these are the same, but for pointers and references to instances in an inheritance hierarchy, they can be different

- What is meant by the terms "static type" and "dynamic type"? Give an example of each
  - `int x{5};`
    - Static type `int`, dynamic type `int`
  - `Drawable *p;`
    - Static type is pointer to `Drawable`
    - Dynamic type could be pointer to `Drawable`
    - Could also be a pointer to any subclass of `Drawable`
    - The dynamic type can only be known at runtime and depends on the type of the subclass which `p` is pointing to

- Give the static and dynamic types of the following variables

<code>int x{5};</code>	<code>// Static type int, dynamic type int</code>
<code>int&amp; x{5};</code>	<code>// Static type int&amp;, dynamic type int&amp;</code>
<code>Circle circle;</code>	<code>// Static type Circle, dynamic type Circle</code>
<code>Circle&amp; circle2 = circle;</code>	<code>// Static type Circle&amp;</code> <code>// Dynamic type Circle&amp;</code>
<code>Drawable&amp; drawable = circle;</code>	<code>// Static type Drawable&amp;</code> <code>// Dynamic type Circle&amp;</code>

- Is it possible to create a reference to a derived type and bind it to a base class instance? If not, why not?
  - No, we cannot bind (for example) a Circle reference to a Drawable instance
  - A Drawable could represent itself or any of its subclasses
  - There is no reliable way in C++ to know which of these types is the correct one

- Give an example where the difference between static and dynamic typing is useful
  - Storing instances from an inheritance hierarchy in a container
  - We create a container which stores pointers to the base class
  - We can populate it with pointers to instances of any type in the hierarchy, without losing type information