**1.**

Welcome to Polymorphism Visualisation.

This tool aims to help you understand of polymorphism concepts, and visualise the relationship between the declared variable type and the type of the instance assigned to that variable.

**2. Screen List**

The orange section on the left contains a number of “screen” shapes, each of which represents a variable type. Each of the different shapes represents a different class. The slots in the screen represent where it expects fields and methods to be for that particular class type.

**3. Object List**

The blue section on the right contains a number of paper sheets, each of which represents an instance of some class type. They contain the fields of the objects along with references to the methods for a class. The shape of an instance will match up with the shape of the class it represents.

**4.**

In OOP, the declared variable types are fixed at compile time. At runtime, multiple instances may be assigned to the variable, each with different values for their fields. Likewise, in this app, the variable type must be placed first, and is fixed, while different instances can be swapped in and out from it.

**5. Clear button**

If you want to place a different variable type, press the clear button to remove the current selection.

**6. Screen list, DropRegion**

Begin by dragging one of the variable types, and placing it in the region in the centre

(Transition on screen placed)

**7. DropRegion, Object List, Top Fade**

Now also place the matching instance from the list on the right into the middle.

(Transition on object placed)

**8. DropRegion, Object List**

These sheets represent data to be read by a machine, which uses the screen as a guide for where it should find the expected pieces of information. Similarly in OOP, the variable type acts as a guide telling the compiler what functionality and fields to expect of the instance assigned to the variable.

The tinted gaps represent private fields of the object. They cannot be directly accessed by an “outside source”, but can instead be accessed by going through the public methods represented by the non-tinted gaps in the cover.

Now try to place an instance that doesn’t match the variable type.

(Trigger on error animation completion)

**9. DropRegion, ObjectList**

As we can see variables and objects that aren’t the same type don’t fit together, and will be rejected. This is due to them having different expectations of available fields and methods, which don’t match up.

**10. Question, Question space**

Questions will appear at the top of the app, with the associated code snippet at the bottom. Questions will either get you to recreate the scenario in the code snippet, or ask you a question about the code shown.

Before you answer the question, recreate the scenario using the screen covers and instances available to assist you in answering it.