**1.**

Welcome to Polymorphism Visualisation.

This tool aims to assist students in their understanding of concepts relating to polymorphism, and their ability to visualise the relationship between variable type and the type of the object assigned to that variable.

**2. Screen List**

This left-hand section contains screens which represent the variable type. Each of the different shapes represent a different class. The slots in the screen represent where it expects fields and methods to be for the class.

**3. Object List**

The right-hand section contains sheets, which represents the objects which are assigned to the variables. They contain the fields of the different objects along with references to the methods for a class. The shape of an object will match up with the shape of the same variable type.

**4.**

In OOP, the variable types are fixed at compile time, while multiple objects can be passed into the variable at run time, each with different values for their fields. Likewise, in this tool, the variable type must be placed first, and is fixed, while different objects 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 place the matching object from the list on the right into the middle too.

(Transition on object placed)

**8. DropRegion, Object List**

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

The Tinted gaps represent private fields of the object. They cannot be directly accessed by an outside source such as the machine, but can instead be accessed by going through methods of the object. The non-tinted gaps represent the methods of the objects.

Now try to place an object 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 region of the screen, with their relevant code snippet at the bottom of the screen. Questions will either get you to recreate the scenario in the code snippet, or ask you a question about the snippet shown.

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