1. **Screen List**

In this section, the classes involved now include inheritance. If you look closely at the screens, some will have the same shape as others, but with an extra section added on to the bottom.

This represents the idea of inheritance, in which classes that extend other classes gain all the functionality of the parent class, along with adding new parts of their own.

1. **Inheritance Tree**

The inheritance tree button at the top will bring up a diagram showing the inheritance structure of the classes used in this question. If you are unsure what inherits from what, take a look at this diagram.

1. **Screen List, DropRegion**

Begin by dragging one of the variable types that inherit from another type into the central area.

(Transition on animation complete where gen = 2)

1. **Object List, DropRegion**

Now place an instance whose type is the parent of the variable type you have placed.

(Transition on placement when object = parent of screen)

1. **ObjectList, Drop region**

Although the instance seems to “fit” under the screen for the variable type, it does not fill up the entire space. There are some slots in the screen that are empty. When the machine tries to get the information from these sections, there would be nothing there, causing it to error.

Similarly, in polymorphism, instances of a parent type cannot be placed inside a variable of a child type, as they do not contain all the functionality of that child type.

(clear screen)

1. **ScreenList, Objectlist, DropRegion, ClearButton**

Now place a variable type of a class that is a parent of another class, along with an object that is of a class that inherits from the variable type. If you need to change your selection, press the clear button.

(transition on placement when gen = 1 and other = child)

1. **DropRegion**

As we can see, the instance now fully fits within the variable type. The instance appears to have extras extruding the screen, but that’s not a problem. What’s important is that all the required methods and fields of the parent class can be accessed, and any “child-type extras” on the instance will be ignored.

Similarly, in polymorphism, if a child-type instance is assigned to a parent-type variable, only the methods and fields defined in the parent class may be accessed by the compiler. The extras in the child-type instance are ignored, as the compiler only sees guides for the parent type.

Questions will now ask you about parent and child classes and the relationship between the two.