Homework 8

Tejas Kamtam

305749402

CS 131 - Fall 2023

Problem 1

Interfaces are not needed in dynamically-typed languages because the type of a variable is not known until runtime. So, the interpreter will not know what type of interface to use until runtime.

Problem 2

YOu may want to use interface inheritance when you plan to create classes that will need to inherit from multiple similar superclasses but require different implementations of the same methods. For example, if you have a class that needs to inherit from both a `Shape` class and a `Color` class, you can create an interface that inherits from both of these classes and then have your class inherit from that interface. This way, you can determine which implementation of the methods you want to use.
For the opposite, you may want to use subclass inheritance if you want to build on an existing class and add more functionality to it. For example, if you have a `Shape` class and you want to create a `Circle` class, you can have the `Circle` class inherit from the `Shape` class and then add more functionality to it.

Problem 3

Part a

- Class A has no supertypes
- Class B has supertypes A
- Class C has supertypes A
- Class D has supertypes A, B, C
- Class E has supertypes A, C
- Class F has supertypes A, B, C, D

Part b

A function `void foo(B b)` can be called with objects of type B, D, or F.

Part c

No, a function `void bar(C c)` called with objects of type `A` will not work because `A` is not a subtype of `C`.

Problem 4

Inheritance, subtype polymorphism, and dynamic dispatch differ in their relationship between classes and subclasses. Inheritance is the relationship between a superclass and a subclass. Subtype polymorphism is the relationship between a superclass and a subclass where the subclass can be used in place of the superclass. Dynamic dispatch is the relationship between a superclass and a subclass where the subclass can override the superclass's methods.

Problem 5

We can't use subtype polymorphism in dynamically typed languages because the type of a variable is not known until runtime. So, the interpreter will not know what type of interface/class to use until runtime, it instead uses duck typing. Dynamic dispatch can be used in dynamically typed languages because the interpreter will know what type of class to use at runtime. An example of this is in Python, where you can use the `super()` function to call a method from a superclass.