Homework 7

Tejas Kamtam

305749402

CS 131 - Fall 2023

Problem 2

The template approach is fast and catches errors early by doing type-checking at each call of the function with distinct types (as it builds tempalted versions of the function each time a unique type set of parameters are called), but it can take a long time to compile to generate unique versions for each combination of usage. Generics create a balance between speed and readability by allowing for a single function to be run and created at compile time, but they also contribute to more specific code (suprisingly generics are less "generic" as they require type-generic operations only, unless inheriting from some class/interface). Duck typing is quite different to the other two as it is used in dyunamically typed languages as type checking is not required at compile time. It makes code easy to read and flexible but might slow down the program and lacks safety during compile time. If designing a new language, I think it's a good idea to go for a mix, using generics for a balance of speed and readability, ensuring clear error messages during compilation. The choice depends on what the language is trying to achieve.

Problem 3

Dynamically typed languages do not require type checking at compile time, and therefore cannot support parametric polymorphism as it requires type checking at compile time to ensure that the types are correct. This is because parametric polymorphism allows for a function to be written generically so that it can handle values identically without depending on their type. This is not possible in dynamically typed languages as they do not require type checking at compile time, and therefore cannot ensure that the types are correct when called.

Problem 4

In statically typed languages that use templates, we can accomplish something like duck-typing by using templates to create a generic function that can be used for multiple types. This is similar to duck typing in dynamically typed languages as it allows for a function to be used for multiple types, but it is different as it requires type checking at compile time to create unqique versions of the function for each type. This means that duck typing in statically typed languages that use templates is safer than duck typing in dynamically typed languages as it ensures that the types are correct at compile time, but it is also slower.

Problem 6

In JavaScript there are no classes, only objects, so we can create uniform sets of objects that all have the same methods/fields by creating individual objects with the same fields, or by "prototyping" objects - a new development that makes individual objects behind the scenes. The prototype property allows us to add new properties and methods to an object's prototype, and all objects created from the same prototype will inherit the same properties and methods.

Problem 8

Because the class `IShape` contains "pure virtual functions," any deriving class or object of that class must first implement the functions in `IShape`, thus it works as an interface - allowing the creation of subclasses that derive from it but not allowing the creation of objects of that class. However, the pointer is allowed to be created because it does not have to instantiate the pure virtual functions, it can just be a pointer to represent some interfaced object - like how a void pointer can be used to represent any type of pointer without the actual object having to be created.