Joshua Catoe

CSCI 350

Reflection Assignment - ADT and OO

1. What are the three main requirements for a language to support the object-oriented paradigm?

**For a language to be object-oriented, it must support abstract data types (ADT’s), inheritance, and dynamic binding, mainly its allowance of polymorphism.**

1. What is the difference between a struct (C - style) and an ADT?

**A struct is just a collection of data (variables), whereas an ADT is a collection of private, internal data fields and methods that can be used to operate on said data fields.**

1. Discuss five reasons why the String ADT is an improvement on using char[] for storing text.

* **Methods that might be used on a character array have already been implemented in** String **and can be called on any** String **object.**
* **Working with a data type is much less hassle than working with an array.**
* **Operations such as concatenation have simpler syntax and are easier to use.**
* **Using** String **is much more time-efficient for the programmer.**
* **The programmer does not have to worry about the implementation of the** String **ADT (i.e. how character arrays are used in** String**).**

1. Should instance members be public or private? Why?

**Instance members should be kept private to reinforce the idea of abstraction. If they need to be accessed or changed, it should be done using methods such as getters and setters.**

1. How would you force an instance method to be bound statically instead of dynamically? Why might you do this?

**To force static binding of an instance method, simply add the** static **keyword to the method declaration. One of the main reasons for doing this is to access certain methods without instantiating an object of a class. For instance, Java has a class called** Math **that contains the static method** sin**, which computes the sin of an angle. It would be needless to have to instantiate a** Math **object just to perform a common mathematical computation.**

1. Write two simple classes and one main method in Java or C# (make your own example) and label each of the following: instance data member, class data member, getter, setter, constructor, parent class, child class, final method, abstract method, overriding method, overloaded method, instantiation of class, polymorphic method call.

static void Main(string[] args)

{

Pet pet = new Pet(“Axel”, 7, “Will”); // Instantiation of class

pet.setColor(“brown”); // Polymorphic method call

}

public class Animal // Parent class

{

private string species; // Instance data member

private int weight; // Instance data member

private string color; // Instance data member

public Animal (string species, int weight, string color) // Constructor

{

this.species = species;

this.weight = weight;

this.color = color;

}

public string getName() // Getter / overwritten method

{

return species;

}

public final void setColor(string color) // Setter / overloaded method / final method

{

this.color = color;

}

public void setColor() // Setter / overloaded method

{

color = “black”;

}

}

public class Pet : Animal // Child class

{

private string name; // Instance data member

private int age; // Instance data member

private string owner; // Instance data member

private static string alive = yes; // Class data member

public Pet(string name, int age, string owner) // Constructor

{

this.name = name;

this.age = age;

this.owner = owner;

}

public string getName() // Getter

{

return name;

}

public void setOwner(string owner) // Setter

{

this.owner = owner;

}

public abstract void giveAttention() // Abstract method

{

}

}

1. OO is powerful, but not all-powerful. Same for functional. Find an opinion article that argues OO is better than Functional, and another that argues the opposite (They’ll both be biased and incomplete, but hopefully still informative). Read the comments for better perspective. Cite both articles and discuss what you took away from them.

**Riccardo Cardin -** [**Object-Oriented Programming strikes back!**](https://rcardin.github.io/object-oriented/functional/programming/2017/08/02/object-oriented-programming-strikes-back.html)

**In this article, Cardin is arguing for OO with the argument that it is more popular because it is easier to learn. More specifically, he notes that functional programming requires much more mathematical understanding to use effectively, while OO does not, making OO easier to learn. He also makes the interesting observation that less strict languages, like OO, are easier to learn, but more prone to error, while more strict languages, like functional, are harder to learn but less error prone. While it makes sense that more people would be inclined to learn OO programming due to its less steep learning curve, I don’t believe that ease of use would necessarily make one paradigm better than another.**

**Andrew Tait -** [**Is functional programming more effective than object-oriented programming?**](https://blog.learningtree.com/functional-programming-object-orientated-programming/)

**Here, Tait is arguing for functional programming with the argument that the three staples of OO can be done in other paradigms and that the design benefits of OO are not very numerous. He mentions that encapsulation and polymorphism can be achieved under other paradigms; in other words, they are not exclusive to OO programming, even though they are some of OO’s most touted features. He also mentions that the usefulness and benefits of inheritance are overstated, which even the designers of OO would agree with. Next, he argues that design benefits in OO are slim, as viewing objects as real-world things is not very useful at a low-level. Finally, a few studies are mentioned that concluded that functional programming produces the best quality software. While Tait presents some compelling arguments, none of them lead me to believe that functional is better than OO. The fact that encapsulation and polymorphism can be used in other paradigms is really an empty argument, as functional and OO would be equal in that respect. Not seeing design benefits of OO comes from his noted bias from being a data scientist and viewing code and data separately. Lastly, quality software depends on the programmer and how skilled they are in their respective paradigm/language; it does not depend on the paradigm/language itself.**