Emily Douglas

PLP 5 - Classes and Inheritance

Javascript!

1. **Does your language support objects or something similar (e.g., structs or records)?**
    1. **Are there naming conventions for objects, instance variables, or functions that people writing in your language should be aware of?**
        - i. Yes, JavaScript fully supports objects and is an object-oriented language. In JavaScript, almost everything is an object, including functions, arrays, and standard data types. Objects are collections of key-value pairs (called properties), and you can also add methods (functions) to objects.
2. **Does your language have standard methods for functions that serve a similar purpose across all objects? For example, toString() in Java and __str__ in Python allow information about the objects to be printed. Are there similar functions in your language?**
    1. Yes there are standard method functions taht are commonly used for objects.
        - i. **`toString()`** in JavaScript is similar to Java's `toString()` and Python's `__str__()`. You can override it to return a custom string representation of an object.
        - ii. **`valueOf()`** can be overridden to return a primitive value, though it's less common.
        - iii. **`JSON.stringify()`** is widely used to serialize objects into JSON format.
        - iv. JavaScript also automatically converts objects to strings when needed, calling the `toString()` method in the background.
3. **How does inheritance work (if it does)? Does your language support multiple inheritance?**
    1. Inheritance is similar to Java in Javascript, and is supported using `extends` and `super()` in class-based syntax or prototypes in older syntax.
    2. However, multiple inheritance unfortunately is not supported in Javascript.
4. **If there is inheritance, how does your language deal with overloading method names and resolving those calls?**
    1. JavaScript does not support method overloading in the traditional sense like some other languages (e.g., Java, C++). In those languages, you can

define multiple methods with the same name but different parameter types or numbers of parameters. JavaScript, however, only allows one method per name in any given scope, including within a class.

i. When two methods with the same name are defined in JavaScript, the last defined method will overwrite any previous definitions. JavaScript resolves this by only keeping the latest version of the method, ignoring the earlier ones.