OOP Reference Sheet

- **Encapsulation**: The practice of hiding the data and behavior of a class from external access or manipulation.
- **Inheritance**: The mechanism that passes traits of a parent class to its descendants. Each object, class, or prototype can be assigned attributes and methods that **inherit** the behavior of its ancestors.
- **Implementation inheritance**: The data and methods defined on a parent class are available on objects created from classes that inherit from those parent classes
- **Prototypal inheritance:** A method of realizing implementation inheritance through process of finding missing properties on an object by delegating the resolution to a prototype object. JavaScript is one the few languages that uses *prototypal inheritance*.
- **Instance methods**: Methods that act on a given instance of the class stored in a variable (i.e. `pig.oink()`). Instance methods are the most common type of method.
- **Static methods**: Methods that are invoked on the class, not a specific instance (i.e. `Math.random()`). `Static methods` should only be reserved for special cases.
- **Properties**: A property is a specific trait of an object. Properties are also referred as *fields*, *attributes*, and *members*.
- **Methods**: A method is function defined on a specific class. *Instance methods* can only be used by instances of a class while *class methods* can be used by the class itself. Class methods can modify the all instances of a class while instance methods can only modify **one** specific instance of a class. There are two types of methods: *accessors* (get data from the object) and *mutators* (change the state of data of the object).
- **Class**: A type of object. A `class` is typically defined with PascalCase. It represents the blueprint or framework of a definition of object.
- **Instance**: A created object or specific initialization of a class. An `instance` has instance variables (properties of the created object) and instance methods (methods that can be invoked on the object, i.e. `pig.oink()`).
- **Polymorphism**: Each object, class, or prototype can be assigned attributes and methods that **extend** the behavior of its ancestors. A type of polymorphism is *subtype polymorphism* where a subclass uses inheritance and **extends** upon the functionality of a parent class. For example, *erasing* can be accomplished using a separate `Eraser`, the end of a `Pencil`, or by using "whiteout" for a `Pen`.
- **Abstraction**: Object-oriented class design should have methods and attributes that make sense for a physical representation of that object. This is closely related to Encapsulation and sometimes known as the 4th pillar of OOP.
- **The `extends` keyword**: The keyword in JavaScript that allows one class to inherit from another.
- **The `constructor` method**: The special method of a class used to initialize an object with the `new` keyword and communicate required data to initialize an object. The `constructor` is the assembly line to produce instances of an object.
- **The Law of Demeter**: A design guideline centered around *loose coupling*, where each class has limited knowledge of other classes. Practically, a method of an object can only invoke the methods (or use the properties) of the following kinds of objects: methods on the object itself, any of the objects passed in as parameters to the method and objects created in the method, any values stored in the instance variables of the object, and any values stored in global variables. (Tip: don't use more than one dot, not counting the one after "this".)