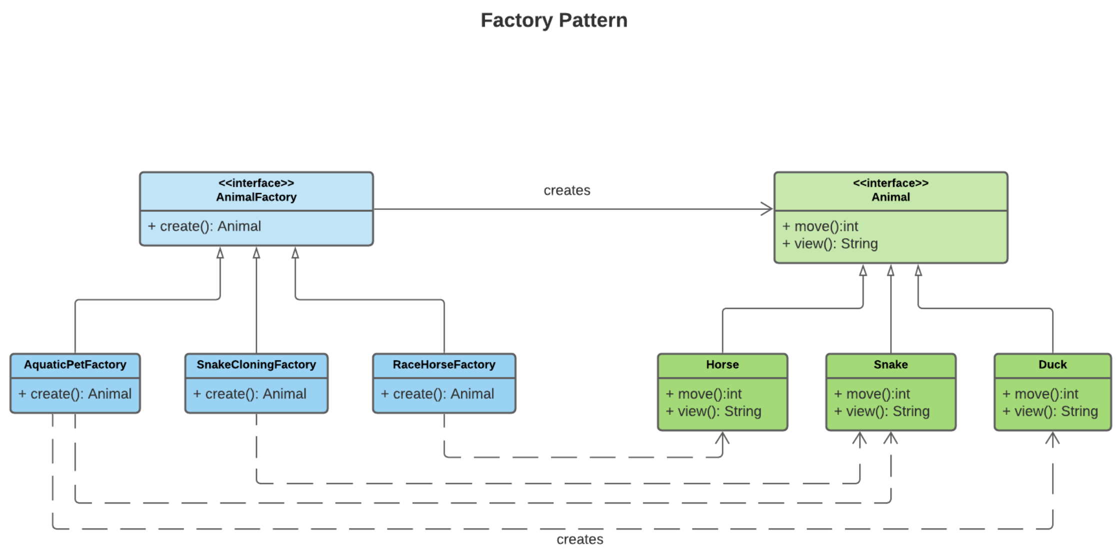
Factory Pattern

Design Pattern Details

Title	Factory Method Pattern
Description	The factory method pattern is a creational design pattern that abstracts the creation of object to concrete implementations of various factories. This allows developers to change how objects are created, modify specifications for new objects, and otherwise decouple implementation details from just getting a new object.
Anatomy diagram	 Factory Pattern <<interface>> AnimalFactory + create(): Animal creates <<interface>> Animal + move():int + view(): String AquaticPetFactory + create(): Animal SnakeCloningFactory + create(): Animal RaceHorseFactory + create(): Animal Horse + move():int + view(): String Snake + move():int + view(): String Duck + move():int + view(): String creates
Key to anatomy diagram	1. AnimalFactory factories create Animal objects 2. A Horse, Duck, or Snake object “is-a” animal 3. An AnimalFactory concretion - like AquaticPetFactory “is-a” Animal Factory

Specifications

Usage	**As a creational design pattern, the Factory Pattern or more specifically Factory Method pattern is used to abstract how objects are created. This is usually accomplished either with inheritance or more flexibly implementing an “Interface” - whether this is done with a shared set of methods in language that might use duck typing (like python) or more strictly with an actual Interface like in Java.**
Variations	The Factory Pattern takes on a variety of different flavors. More commonly and discretely, Factory Method pattern is used to abstract and delegate how different types of objects are created with various configurations. Alternatively, at a higher level Abstract Factory can be used to abstract not only factories but the types of objects created. More information can be found below. Factory Method Pattern: http://www.blackwasp.co.uk/factorymethod.aspx Abstract Factory Pattern: http://www.blackwasp.co.uk/abstractfactory.aspx
Behavior	Factories create new objects based on their concrete implementation. This can change dynamically at runtime.