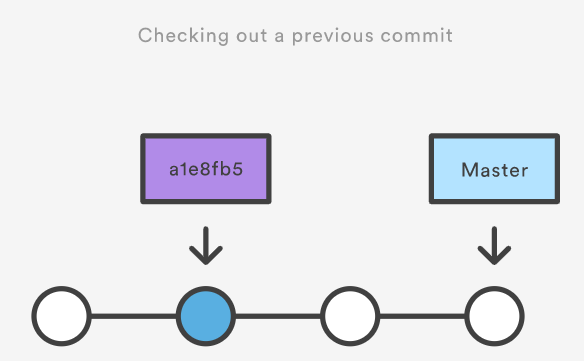
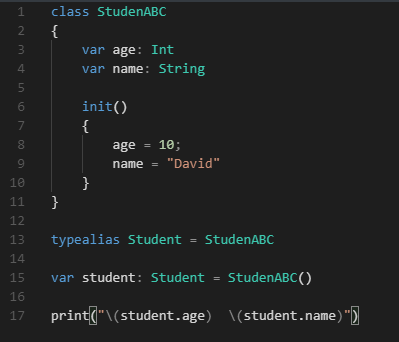
* Version Control System (VCS)
  + SVN / Subversion
  + Git
  + Mercurial
  + TFS
* BLOB (Binary Large Object)
* Git vs SVN:
  + Git : distributed version control system, each developer gets their own local repository, complete with a full history of commits
    - Git store each version of a file as a full file -> faster to look up history
  + SVN: each developer gets a working copy that points back to a single repository
    - SVN stores previous version of a file binary file -> take time and CPU to look up history
* Git commands:
  + Git init: create an empty Git repository with subdirectories for objects, refs/heads, refs/tags, and template files.
    - Initial HEAD file
  + Git HEAD: a reference to the last commit of the currently check out branch (we can HEAD as “current branch”)
    - Whenever we use git checkout to change branch, HEAD changes
    - HEAD usually point to a master or some other local branch
    - When you check out a previous commit, HEAD no longer points to a branch – it points directly to a commit => it’s called “detached HEAD” state
    - 
  + Git clone: point to an existing repo and make a clone or copy of that repo into a new directory, at another location
  + Git log: reviewing the history of a repository
  + Git checkout:
    - There are three distinct entities:
      * Files
      * Commits
      * Branches
* SDLC (Software Development Life Cycle)
  + Who is CSM (Certified Scrum Master)
    - CSM is person who take documents (resources, finance, bottle neck), in charge of the process of development, who does not have to know about technical.
  + Program manager:
  + Lead: managing team, support team for any impediments that stopping them.
  + Agile:
    - Feasibility:
* Swift:
  + Swift playground
    - Libraries:
      * UIKit:
        + Offers all the libraries to support UI development

Ex: Scene, buttons

* + - * Foundation:
        + Integer, String, Data

Data: A byte buffer in memory

* + Type Inference: automatically determines the type of variable according to the value assigned to the variable when a type is not implicit
  + Automatic Reference Counting:
  + Type Safety Feature: compiler gives an error once a variable is assigned any data type which is not compatible to the type of the variable
  + Type Annotations: explicitly providing the data type to any variables or constants
  + Naming Conventions:
    - Variable and constants should be descriptive enough to tell their functionality or usages
    - Camel casing: firstName
    - Snake casing: first\_name
    - Pascal casing: FirstName
    - Unicase or Nocase: firstname
    - Train casing: First-Name
  + Conversion of types in Swift
    - String Interpolation
      * Let string = “\(doubleValue)”
    - String Concatenation (appending two or more strings)
      * Let string =”\(double value) is your weight for \(toDate)”
  + Type Aliases:



* Tuples: a way to group multiple variables to one variable (compound type)
* Optional values: values that might be nil (nullable)
  + var myConditionalValue: String? = nil
  + // Use ? to declare optional variable
  + Access optional value
    - Using Conditionals



* + - Using Optional Binding

* + - 