Family class

"Family class stores all relevant family information:"
"Mother and Father, marriage and divorce dates"
"Ordered collection of children"

new
"Override new to initialize the object"

Object subclass: #Family

addChild: aPerson
"Adds a person to the OrderedCollection of children"

children
"Accesses the instance variable children, which is an OrderedCollection referencing all the children of this Family"

divorceDate
"Accesses the instance variable divorceDate of the Family, which is a Date of when the parents divorced"

divorceDate: aDate
"Set the instance variable divorceDate of the Family, which is a Date of when the parents divorced"

father
"Returns the father of this Family"

father: aPerson
"Sets the father of this Family"

marriageDate
"Accesses the instance variable marriageDate of the Family, which is a Date of when the parents married"

marriageDate: aDate
"Set the instance variable marriageDate of the Family, which is a Date of when the parents married"

mother
"Returns the mother of this Family"

mother: aPerson
"Sets the mother of this Family"
initialize
  "Initialize the instance variables"

printChildren
  "Return a string of the children that are in this family"

printFamily
  "Returns a string of parent information and children information"

printParents
  "Returns a string of all the parent information and marriages/divorces"

Person class
  "Person class stores all relevent individual information:"
  "name, gender, aliases, parent and participant families"
  "birthday, birth location, death date, and misc information"

new
  "Overrides new so that we can initialize the Person object"

Object subclass: #Person

addAlias: nickname
  "Adds an alias for the person"

addOwnFamily: aFamily
  "Adds a Family to the OrderedCollection of ownFamilies"

born: aDate location: aString
  "Sets the birthday and location of birth for a person"

check
  "Runs checkPerson on the family tree to check for missing vital information"

checkPerson
  "Checks for missing vital information"

fullName
  "Returns the person’s givenName and surName"
genderIsMale
"Returns true if self is a male, otherwise returns false."

getAncestors
"Returns an ordered collection of all ancestors of me, which doesn’t include me"

getDescendants
"Returns an ordered collection of all descendants of me, which doesn’t include me"

getFamilyTree
"Returns an ordered collection of all ancestors, descendants, and me"

givenName
"Returns the person’s givenName"

givenName: firstName
"Sets the person’s givenName"

hasChild: aPerson with: aSpouse
"Adds aPerson as a child of self and aSpouse"

hasFather: aPerson
"Adds aPerson as the father of the parentFamily"

hasMother: aPerson
"Adds aPerson as the mother of the parentFamily"

hasSibling: aPerson
"Will add aPerson as a child of the parentFamily of self. This means that both aPerson and self point at the same

isFemale
"Sets the person’s gender to female."

isMale
"Sets the person’s gender to male."

married: aPerson on: aDate
"Adds aPerson to a Family of self and married on aDate"

married: aPerson on: aDate divorced: aDivorceDate
"Adds aPerson as part of a Family with self that married on aDate and divorced on divorceDate"
name: aString
  "Lets you set a person’s givenName and surName and will delimit on the space"

ownFamilies
  "Return the ownFamilies to be accessed. The ownFamilies variable is a OrderedCollection of pointers to any families"

parentFamily
  "Return the parentFamily to be accessed. The parentFamily consists of a pointer to the parents of this person and a list of pointers to the Families that this person is a parent of."

parentFamily: aFamily
  "Sets the current parentFamily to aFamily, a new family."

personView
  "Returns the personView associated with a person, which is an object that will do all the graphics stuff, drawing boxes and etc."

personView: aPersonView
  "Sets the personView associated with a person, which is an object that will do all the graphics stuff, drawing boxes and etc."

record: keyObject as: valueObject
  "NOTE: if the same record is added again, it will just update the value because miscInfo is a Dictionary"
  miscInfo add: keyObject -> valueObject.

searchOwnFamiliesChildren: aPerson
  "Returns an OrderedCollection of all Families that contain this person as a child"

searchOwnFamiliesParents: aPerson
  "Returns an OrderedCollection of all Families that contain this person as a parent"

surName
  "Returns the person’s surName"

surName: lastName
  "Sets the person’s surName"

initialize
  "Initialize instance variables"

graphicsVisualize
"Show the person and associated relationships"

**printChildren**
"Return a string containing the children of my parents"

**printInfo**
"Return a string of all a person’s vital info except for the miscellaneous information"

**printMiscInfo**
"Return a string of all a person’s miscellaneous information"

**printParentFamily**
"Return a string of a person’s parent family"

**printParents**
"Return a string containing my parents info"

**visualize**
"Show the person and associated ancestors and descendants"

**PersonView class**
"Handles all of the visualization of the Person and Family data objects"

**Object subclass: #PersonView**

**timesOccur: value in: aDictionary**
"This is a function written specifically for the Person visualize method.
This will take in a dictionary, and find out how many times a certain value exists and return that."

**draw**
"Draws my box and puts my name in it"

**drawInfo: place**
"This will draw the Detailed info for a person at a given place if the click is in my box"

**drawLine: aPersonView**
"Draws a line between myself’s top and aPersonView’s bottom"

**ifClicked: click**
"This will draw the Detailed info for a person at a given place if the click is in my box"
Query class

"Stores possible query information for later use with search"

generalSearch: searchString
"Search ALL general information based on searchString"

livedIn: location
"Search ALL livedIn values for location"

livedOn: alive
"Search ALL livedOn values for alive date"