The CS Majors

Nimish Gautam (gte671u)
Adam Davis (gte459u)
Namit Bhatia (gte276u)

CS Project: Milestone 6
Description of Each Class and Methods

- **Person:**
  - Holds information about a Person and all of their relationships with other persons.
  - **Methods:**
    - `check` – Checks the validity of every person in this person’s family tree and prints out a string reporting what is missing, what information was mirrored, and what is conflicting.
    - `isMale` – Sets this person’s gender to be male.
    - `isFemale` – Sets this person’s gender to be female.
    - `givenName: <string>` -- Modifier for given name.
    - `surName: <string>` -- Modifier for surname.
    - `addAlias: <string>` -- Adds an alias to the person’s list (OrderedCollection) of aliases.
    - `hasChild: <person>` -- Adds a child to the person’s list of children (OrderedCollection) and also adds this child to its appropriate family.
    - `hasChild: <person> with: <person>` -- Same as above except adds the second person to a list of would-be parents called **spouses**.
    - `hasFather: <person>` -- Sets the person’s father. Also adds this father to the appropriate family or defines a new family for the father and the child if one doesn’t already exist.
    - `hasMother: <person>` -- Sets the person’s mother. Also adds this mother to the appropriate family of the child if one already exists. If one doesn’t, it defines a new family with only the mother and the child in it.
    - `hasSibling: <person>` -- Adds a sibling to the person’s list of siblings.
    - `married: <person>` -- Add a new marriage object to the person’s list of marriages.
    - `married: <person> on: <date>` -- Same as above except includes date of marriage.
    - `married: <person> on: <date> divorced: <date>` -- Same as above except includes date of marriage and divorce.
    - `record: <record> as: <info>` -- Add this record “definition” to the Dictionary called **recordMisc**.
    - `born: <date> location: <place>` -- Modifier for birthTime and birthPlace.
    - `died: <date> location: <place>` -- Modifier for deathTime and deathPlace.
    - `allRelatives` – Returns a collection of all the people related to this person (including the object that calls it).

- **Family:**
  - Holds information about a nuclear family. Every family has a husband a wife, their children as well as the marriage and divorce date (if any) of the husband and wife.
Methods:

- addChild: – Allows for the addition of a new child to this family. This child must have both the mom and dad of this family as his or her mother and father, or, if there’s only one parent, the child must have that parent defined as his or her mom or dad.
- children – Accessor for a list of all the children of this family.
- Children: - Modifier for defining the list of children in the family.
- divorceDate – Accessor for getting the divorce date of the mom and dad in the family.
- divorceDate: - Modifier for defining the divorce date.
- husband – Accessor for getting the person who is the father in the family.
- wife – Accessor for getting the person who is the mother in the family.
- husband: - Modifier to assign the husband in the family.
- wife: - Modifier to assign the wife in the family.
- marriageDate – Accessor for getting the date of marriage of the father and mother in the family.
- marriageDate: - modifier for the marriage date.
- removeChild: - Method to remove a specified child from the child list in the family.

Marriage:
- Holds the information about a marriage: date of marriage, date of divorce (if any), and the Person who is the spouse.

Methods:

- getDate: Gets date of marriage.
- getDivorce: gets the date of the divorce of this marriage
- getSpouse: gets the person object that is the spouse in this marriage
- setSpouse: sets the spouse of this marriage

Population:
- This class contains two lists of objects (which are implemented as OrderedCollections), Persons and Families.

Methods:

- People: Accessor for class (global) variable People.
- addPerson: Adds a person to the collection, People.
- printPeople: Prints out the names of all the people for testing purposes.
- resetPeople: Sets the People variable to nil for testing purposes.
- export: exports all code needed to regenerate the population to the transcript
- generateGEDCom: Exports the entire genealogy that is currently defined into a GEDCom file so that the genealogy map can be regenerated from the globally acclaimed GEDCom file format.
- **readGEDCom**: Allows for the creation of a genealogy map from a GEDCom file.
- **Families**: Accessor for global variable Families.
- **addFamily**: Adds a new family to the list of families in the population.
- **findChildInFamilies**: This returns a family which has the child in question defined as its child.
- **findFamWithMother:father**: This returns a family which has the specified mother and father.
- **findFatherInFamilies**: This function recurses through Population Families and returns an ordered collection of all families with inMother as mother.
- **printFamilies**: Used for printing all the families for testing.
- **resetFamilies**: Used to set the collection of families to nil (when there has been a reset of population).
- **resolveFatherOf:as**: This function handles the creation (if necessary) of a family with the child and the dad in question, or adds the child to the family with this father as the husband if it is already existent. This takes into account that the father might have several different marriages and will not put the child in the wrong family.
- **resolveMotherOf:as**: This function handles the creation (if necessary) of a family with the child and the mom in question, or adds the child to the family with this mother as the wife if it is already existent. This takes into account that the mother might have several different marriages and will not put the child in the wrong family.
- **resolveSpouseOf:as**: Creates a family with these two in it if necessary.

**Query**:
- This class is responsible for searching the entire population of people for certain information and returning the corresponding people in a list. All of its methods return a list of Person objects in the form of an OrderedCollection.
- **Methods**:
  - **givenName**: <string> -- Returns all people who have that given name.
  - **surName**: <string> -- Returns all people who have that surname.
  - **isMale** -- Returns all people who are male.
  - **isFemale** -- Returns all people who are female.
  - **hasChild**: <person> -- Returns all people whose child this is.
  - **hasChild**: <person> with: <person> -- Returns all people who have had that child with that person.
  - **hasParent**: <person> -- Returns all persons that have that person as a father or mother.
  - **hasSibling**: <person> -- Returns all people who have that sibling.
  - **married**: <person> -- Returns all people who married that person.
  - **married**: <person> on: <date> -- Same as above except includes date of marriage.
- **born: <date>** -- Returns all people who were born on that date.
- **died: <date>** -- Returns all people who died on that date.
- **bornIn: <place>** -- Returns all people who were born in that place.
- **livedIn: <place>** -- Returns all people who lived in that place at anytime in their life. This is accomplished by searching the birth place, death place and general information.
- **livedOn: <date>** -- Returns all people who lived on this date. This is accomplished by making sure that their birth date is before than this date and their death date is after it.
- **generalSearch: <string>** -- Returns all people who have that string existing anywhere in their general information records.

- **GUIController:**
  - This class is responsible for creating the controls, in the form of buttons, for the user, and for providing a graphic way of editing and accessing people.
  - **Methods:**
    - **addAliases** – method for adding aliases in the graphic edit window
    - **addMisc** – method for adding misc. info to the person being created/edited
    - **createNewPersonFromInfo** – parses information from a person edit window and reincorporates it into a person (or if the person didn’t exist, creates them)
    - **exportInfo** – calls “Population export” (function bound to the export button)
    - **genSearchQuery** - perform a global query to find all people who have the entered info anywhere
    - **helpInfo** – displays help information
    - **initialize** – initializes the main morph object and the buttons it needs to display, also binds the buttons to their functions they call
    - **launchNewPersonMenu** – initializes “launchPersonEditMenu” with default values; primes the function to create a new person instead of editing an existing one
    - **launchPersonEditMenu** – creates a new window in which a person (pTemp) is edited. Upon clicking ‘done’ all the new information is parsed and stored in the person object that was passed in. Also, if pTemp is set to nil, this will create a new person
    - **livedInPlaceQuery** - perform a global query to find all people who have lived in this Place
    - **livedOnDateQuery** - perform a global query to find all people who have lived on that particular date
    - **openGlobalQueryMenu** - Opens the menu for calling the three global queries that we have
    - **parseExistingPerson** – takes a person passed in and puts all their information into a graphics window
    - **resetPopulation** – resets the population (calls “resetPeople” in genealogy map)
- **setGmap**- gives a reference to the genealogy map this is attached to
- **setPtemp** – sets the person reference to be edited
- **viewMiscInfo** – allows you to view the misc. info of a person graphically (with popup menus)

- **GenealogyMap**:  
  - This is the main class for representing genealogies and for creating them. Use "GenealogyMap new open" to open a new window for creating and editing a GenealogyMap. This window will include a GUIController. PersonMorphs are used to represent Person objects and are submorphs of GenealogyMap.
  
  - **Methods**:
    - **addParentsOf:** aPerson place: aPlace – Adds the parents of that person to the genealogy map. used for ancestors.
    - **addPerson:** adds this person to the people list ScrollPane and to Population.
    - **createScroller** - Returns a TwoWayScrollPane.
    - **edit**- sets the person selected to be the person GUIController edits
    - **focusOnPerson:** aPerson- Makes this person the focus of the Genealogy Map display
    - **initialize**- initializes the genealogy map
    - **makeLineMorphs**- Makes all the LineMorphs used to display relationships
    - **makeLineMorphsOn:** aPersonMorph- Makes all the LineMorphs used to display relationships.
    - **open**- Opens in world.
    - **peopleChanged**- Updates the position of people in the Genealogy map
    - **personClicked:** person- Called when that person is clicked (his PersonMorph is clicked).
    - **relationState**- gives back relation state
    - **resetPeople**- This is called when Reset Population button is pressed, resets all people and removes the morphs from the lists
    - **setRelation:** symbol with: originalPerson- sets a relation (father, mother etc.) with the original person passed in
    - **setRelation:** symbol with: originalPerson marriedOn: mDate – sets a marriage relation with the date passed in
    - **setRelation:** symbol with: originalPerson marriedOn: mDate divorcedOn: dDate – sets a marriage relation with the person passed in and the date passed in (both marriage and divorce)

- **PersonMorph**:  
  - This class is the view for representing Person objects and interacting with them in the GUI. This class is used by GenealogyMap for displaying information about people to the user and to help the user interact with and edit the people. It has a balloon text that displays the name, date of birth, place of birth, date of death, and place of death (if applicable). It also creates a CustomMenu when right clicked upon which allows the user to "check" the Person's information, do queries
(implemented by Query), and define relationships (implemented in Person), and edit the person (implemented through GUIController). Methods:

- **balloonText**: Returns the balloon text to show for this person morph.
- **birthDateQuery**: This Method initiates input for entering a date and then querying based on that.
- **birthLocationQuery**: This Method initiates input for entering a birthLocation and then querying based on that.
- **childWithNameQuery**: This Method initiates input for the user to enter the first name and the last name of a person to find all his or her children from this person's relatives.
- **deathDateQuery**: This Method initiates input for entering a deathDate and then querying based on that.
- **deathLocationQuery**: This Method initiates input for entering a deathLocation and then querying based on that.
- **genderQuery**: This Method initiates input for entering a gender and then querying based on that.
- **givenNameQuery**: This Method initiates input for entering a the givenName and then querying relatives based on that.
- **havingInfoQuery**: This Method initiates input for entering a key and its entry so a search for those people that have that info can be initiated from the relatives of the current person.
- **initialize**: Initializes the person morph's textmorph.
- **integrityCheck**: This method initiates an integrity check on the familial ties of the person and displays all the trailing ends and faults found in the familial ties.
- **openPrimaryMenu**: This opens up a new menu for every PersonMorph. This menu gives a personMorph the capability to do the following things:
  1. Check for integrity of familial ties and display the faults found.
  2. Intiate the Query menu to perform various queries.
  3. Define relationships with other people on the screen.
- **openQueryMenu**: This opens up a new menu for every PersonMorph, and is used to call a various queries.
- **openRelationMenu**: This method calls a menu that lets you define the person's relationship with another person.
- **parentWithNameQuery**: This Method initiates input for the user to enter the first name and the last name of a person to find his or her parents from this person's relatives.
- **personChanged**: Change the visualization to reflect changes.
- **siblingWithNameQuery**: This Method initiates input for the user to enter the first name and the last name of a person to find all his or her siblings from this person's relatives.
- **specifiedAliasQuery**: This Method initiates input for entering an Alias and then querying for relatives based on that.
- **spouseWithNameQuery**: This Method initiates input for the user to enter the first name and the last name of a person to find his or her spouses from this person's relatives.
- **surNameQuery**: This Method initiates input for entering a surName and then querying based on that.

### WebParser:
- Prototype class that all the other web parsers are based off of.
  - **Methods**:
    - `find: aPerson in: anOrdColl`: checks the ordered collection passed in for consistency with data already known in the person object passed in. It will add information to the person with the first match. Strings passed in are in a standardized format: name | birthdate | deathdate | deathlocation
    - `getPersonInfo: firstName surName: lastName`: abstract method, only implemented in subclasses
    - `initialize`: makes sure `webPageNameStrings` exists, subclasses will put in the name of the URL that needs to be passed in

### RootsWebParser:
- Prototype class that all the other web parsers are based off of.
  - **Methods**:
    - `getPersonInfo: firstName surName: lastName`: goes to RootsWeb.com and searches social security death index for people with given name. Returns an ordered collection of strings that find: in: parses to find a match.
    - `initialize`: initializes the `webPageNameStrings` to strings specific to rootsweb

### AncestryWebParser:
- Prototype class that all the other web parsers are based off of.
  - **Methods**:
    - `getPersonInfo: firstName surName: lastName`: goes to ancestry.com and searches social security death index for people with given name. Returns an ordered collection of strings that find: in: parses to find a match.
    - `initialize`: initializes the `webPageNameStrings` to strings specific to ancestry

### GenealogyComParser:
- Prototype class that all the other web parsers are based off of.
  - **Methods**:
    - `getPersonInfo: firstName surName: lastName`: goes to Genealogy.com and searches social security death index for people with given name. Returns an ordered collection of strings that find: in: parses to find a match.
- **initialize** – initializes the webPageNameStrings to strings specific to Genealogy.com

- **LineageWebParser**:
  - Prototype class that all the other web parsers are based off of.
  - **Methods**:
    - **getPersonInfo: firstName surname: lastName** – goes to Lineage Web and searches social security death index for people with given name. returns an ordered collection of strings that find: in: parses to find a match.
    - **initialize** – initializes the webPageNameStrings to strings specific to the lineage website