UML Description

Daisong Yu (500255731)

1. **Comp**

   Program start from Comp class, after program run, this class will create an ECB object.

2. **ECB**

   ECB object will responsible for most operations, it will read contact file and instruction file and generate output file and report file.
   ECB have a phone book object which store the records. ECB can add, delete, save and query the phone book, according to the instruction file.

3. **PhoneBook**

   PhoneBook object use an ArrayList to store record, ArrayList can have zero to many records. PhoneBook has the method to add or delete the record from the ArrayList. If PhoneBook

4. **Record**

   Record object store the information of each record, which contains the Name, Birthday, Phone, Email and Address obejct. Record object has the getter and setter of these information. If phone book not exist all the records will also not exist.

5. **Name**

Name object use String to store a name, Name can check if a phone is valid name for ECB. Name also has the getter and setter for name.

6. **DoB**

   DoB object use String to store a birthday, DoB can check if a birthday is valid birthday. DoB also has the getter and setter for birthday.

7. **Phone**

   Phone object use String to store phone, Phone can check if a phone is valid and remove leading zero. It also contains the getter and setter for phone.

8. **Email**

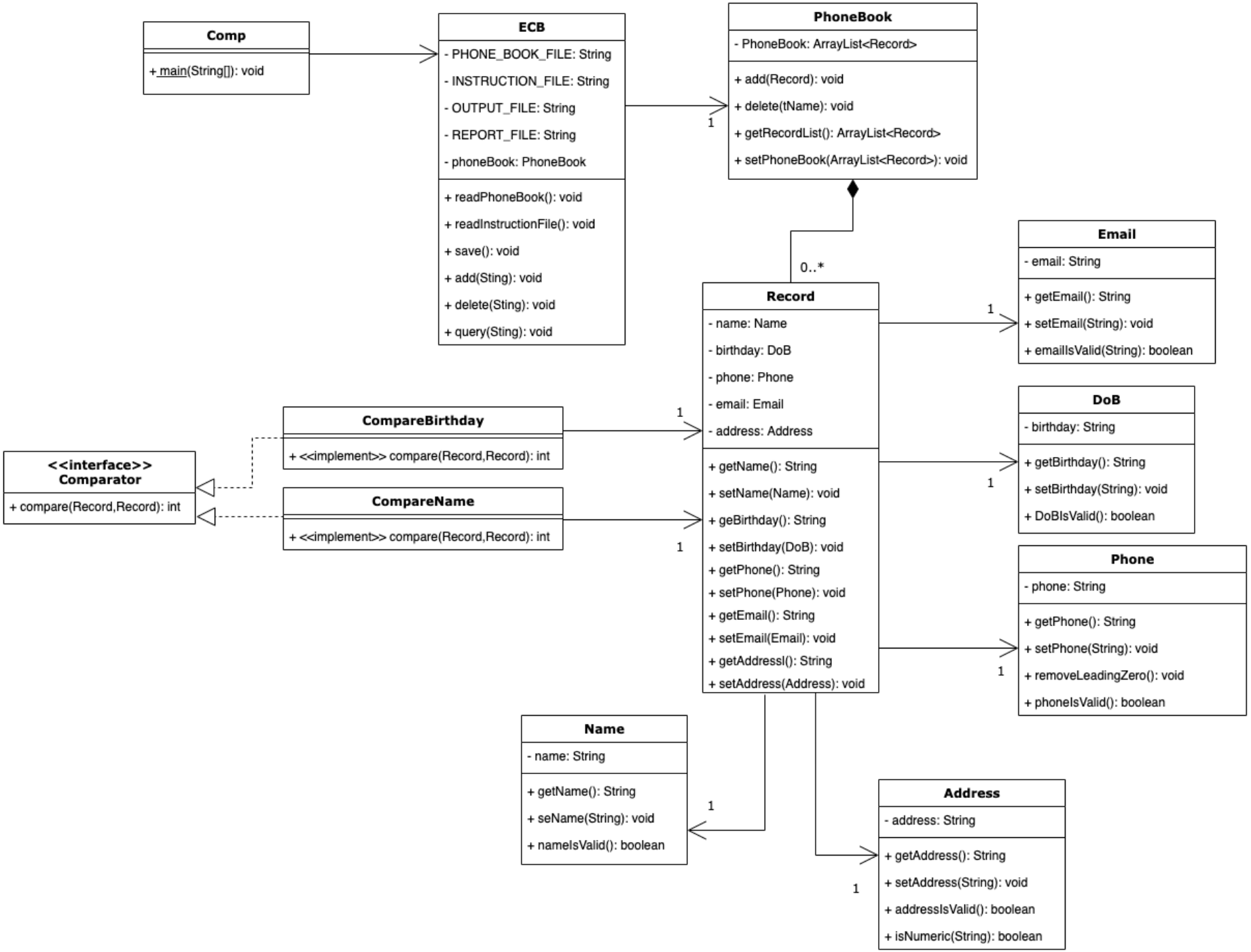
   Email object use a String to store an email, Email object can check if an email is a valid email. It also provides the getter and setter for email.

9. **Address**

   Address object use a String to store the address, if can check if an address is valid and provide getter and setter for address.

10. **CompareBirthday and CompareName**

    These two class implement interface Comparator, they are used for sort ArrayList<Birthday> and ArrayList<Name>. So they are associate with Birthday and Name class


Comp
+ main(String[]): void
ECB
- PHONE_BOOK_FILE: String
- INSTRUCTION_FILE: String
- OUTPUT_FILE: String
- REPORT_FILE: String
- phoneBook: PhoneBook
+ readPhoneBook(): void
+ readInstructionFile(): void
+ save(): void
+ add(Sting): void
+ delete(Sting): void
+ query(Sting): void
PhoneBook
- PhoneBook: ArrayList<Record>
+ add(Record): void
+ delete(tName): void
+ getRecordList(): ArrayList<Record>
+ setPhoneBook(ArrayList<Record>): void
1
0..*
Email
- email: String
+ getEmail(): String
+ setEmail(String): void
+ emailIsValid(String): boolean
Record
- name: Name
- birthday: DoB
- phone: Phone
- email: Email
- address: Address
+ getName(): String
+ setName(Name): void
+ geBirthday(): String
+ setBirthday(DoB): void
+ getPhone(): String
+ setPhone(Phone): void
+ getEmail(): String
+ setEmail(Email): void
+ getAddressl(): String
+ setAddress(Address): void
1
1
DoB
- birthday: String
+ getBirthday(): String
+ setBirthday(String): void
+ DoBIsValid(): boolean
Phone
- phone: String
+ getPhone(): String
+ setPhone(String): void
+ removeLeadingZero(): void
+ phoneIsValid(): boolean
<<interface>>
Comparator
+ compare(Record,Record): int
CompareBirthday
+ <<implement>> compare(Record,Record): int
CompareName
+ <<implement>> compare(Record,Record): int
1
1
Name
- name: String
+ getName(): String
+ seName(String): void
+ nameIsValid(): boolean
1
Address
- address: String
+ getAddress(): String
+ setAddress(String): void
+ addressIsValid(): boolean
+ isNumeric(String): boolean
1