Due Date – Monday, September 30, by 11:59 pm.

Submission

(1) Zip the project folder and submit the zipped file to Canvas.
   (a) Windows – right-click the project folder and select Compress to Zip file
   (b) Mac/Linux – right click the project folder and select Compress

(2) The zip file must include the following grading items.
- **The source folder "src",** containing the Java files (*.java) [**100 points**]
  (a) MUST create a Java package to hold the source files, or **lose 2 points**
  (b) MUST use all lowercase letters for the package name, or **lose 2 points**
  (c) MUST include the **@author** tag in the comment block on top of EVERY Java class and put the name(s) who implemented the Java class, or **lose 2 points**
- **A test specification**, including the test cases designed to test the isValid() method of the Date class and the compareTo() method of the Profile class. [**15 points**]
- **A testbed main()** in the following Java classes implementing the test cases in the test specification.
  (a) Date class [**12 points**]
  (b) Profile class [**8 points**]
- The **Javadoc folder** "**doc"** to include all the .html files generated by Javadoc. [**5 points**]

(3) The submission button on Canvas will disappear after **September 30, 11:59 pm**. Do not wait until the last minute to submit the project. You are responsible for ensuring the project is well-received in Canvas by the due date and time. **You get 0 points** if you do not have a submission on Canvas. **Projects sent through emails or other methods will not be accepted.**

Project Description

RU clinic provides medical services at six locations in New Jersey, as listed below.

Bridgewater, 08807, Somerset County
Edison, 08817, Middlesex County
Piscataway, 08854, Middlesex County
Princeton, 08542, Mercer County
Morristown, 07960, Morris County
Clark, 07066, Union County

The clinic operates during the weekdays and closes on Saturdays and Sundays. Patients must make an appointment before visiting the clinic. Six timeslots are available for the appointments each day during the weekdays – 9:00 am, 10:45 am, 11:15 am, 1:30 pm, 3:00 pm, and 4:15 pm. Currently, there are eight medical service providers practicing at the six locations. The system maintains the last names, locations where they practice, and each provider's specialty, as listed below.

Patel, Bridgewater, Family
Lim, Bridgewater, Pediatrician
Zimnes, Clark, Family
Harper, Clark, Family
Kaur, Princeton, Allergist
Taylor, Piscataway, Pediatrician
Ramesh, Morristown, Allergist
Ceravolo, Edison, Pediatrician

Your team will develop a scheduler software to help the clinic manage the appointments. The clinic has given you the list of requirements below.

1. The system shall maintain an appointment calendar for all upcoming appointments.
2. The system shall keep track of the date, time, patient, and provider for each appointment.
3. The system shall uniquely identify each patient with a profile containing the patient's first name, last name, and date of birth.
4. The system shall be able to schedule a new appointment for a patient with the requested date, time, and provider.
5. The system shall not schedule an appointment if a given patient has an existing appointment with the same date and time.
6. The system shall be able to cancel an existing appointment given the patient and the date and time of the appointment.
7. The system shall reschedule an existing appointment given a different timeslot on the same day.
8. The system shall be able to display the list of appointments ordered by the patient profile and appointment date and time, or by the appointment date and time and provider, or by location.
9. The system shall maintain a list of visits for each patient when the patient's appointments have been completed.
10. The system shall be able to print the billing statements for all patients when the appointments are completed.

The software shall be an interactive system where the users enter command lines on the terminal, and the system immediately generates responses and outputs the results on the terminal. When the user enters a common line and hits the enter key, the system reads the data entered, processes the data, and immediately displays the results.

A command line always begins with a command followed by additional data tokens delimited by commas. Commands are in uppercase letter(s) and **case-sensitive**, which means the commands in lowercase letters are invalid. However, the data tokens are NOT case-sensitive; for example, “SOMERSET” and “somerset”, “JOHN DOE” and “john doe” are equivalent. Below is a list of commands the software must support.

- **S** command: to **schedule a new appointment** and add the appointment to the calendar. The user shall enter the appointment information in the following sequence: the date, timeslot, patient’s first name, last name and date of birth, and provider’s last name. Below is an example of a command line for adding a new appointment. You can assume that the user will always enter enough data tokens if a valid command is entered.

  ```
  S,9/30/2024,1,John,Doe,12/13/1989,Patel
  ```

  The date of birth and appointment date shall be given in a **mm/dd/yyyy** format. Your software must check if the data tokens are invalid in the below order.

  1. The date of an appointment is
     - an invalid calendar date,
     - today,
     - a date before today
     - a date on Saturday or Sunday
     - a date not within six months from today.
  2. The timeslot does not exist.
  3. The date of birth of a patient is not a valid calendar date, is today, or is a future date.
  4. An appointment with the same patient profile, date, and timeslot already exists.
  5. The provider does not exist.
  6. The provider is not available at the specified timeslot.

- **C** command, to **cancel** an existing appointment and remove the appointment from the calendar, for example,

  `C,9/30/2024,1,John,Doe,12/13/1989`

  The system shall use the patient's profile and appointment date and time to uniquely identify an appointment in the appointment calendar. Since the system checks the invalid data tokens before adding the appointment to the calendar, the system does not check the invalid data tokens when canceling an appointment. However, the appointment being canceled may not exist in the appointment calendar.

- **R** command to reschedule an appointment to a different timeslot on the same day with the same provider.

  `R,9/30/2024,1,John,Doe,12/13/1989,2`

  The system shall use the patient's profile and appointment date and time to uniquely identify an appointment in the appointment calendar for rescheduling. The system shall check if the specified appointment exists, AND if the new timeslot indicated at the end of the command line is valid and available from the provider in the original appointment.

- **PA** command to display the list of appointments, sorted by appointment date, time, then provider's name.
- **PP** command to display the list of appointments, sorted by the patient (by last name, first name, date of birth, then appointment date and time).
- **PL** command to display the list of appointments, sorted by the county name, then the appointment date and time.
- **PS** command to display the billing statements for all patients. Each visit is charged based on the provider's specialty, as listed below. For simplicity of the project, when this command is entered, we assume all appointments have been completed and moved from the appointment calendar to the medical record.

  FAMILY - $250,

  PEDIATRICIAN - $300,

  ALLERGIST - $350

- **Q** command to stop the execution of the software and display `"Scheduler is terminated."`

Project Requirement

1. You MUST follow the Coding Standard posted on Canvas under Modules/Week #1. **You will lose points** if you violate the rules listed in the coding standard.
2. You are required to follow the Academic Integrity Policy. See the **Additional Note #13** in the syllabus posted on Canvas. If your team uses a repository hosted on a public website, you MUST set the repository to private. Setting it to public is considered as violating the academic integrity policy. The consequences of violation of the Academic Integrity Policy are: **(i) all parties involved receive 0 (zero) on the project, (ii) the violation is reported, and (iii) a record on your file of this violation.**
3. Test cases for grading are included in the file **Project1TestCases.txt.** The associated output generated from the test cases is in **Project1Output.txt** for your reference. The data for the test cases are fictional for testing purposes. Your project should be able to ignore the empty lines between the test cases in **Project1TestCases.txt**. You MUST use the **Scanner class** to read the command lines from the standard input (**System.in**), DO NOT read from an external file, or you will **lose 5 points**.
4. The graders will run your project by copying the test cases in **Project1TestCases.txt** and pasting them to the terminal. They will compare your output with the expected output in **Project1Output.txt**. You will **lose 2 points** for each output not matching the expected output OR for each exception, causing your project to terminate abnormally.

5. Each source file (.java file) can only include one public Java class; the file name is the same as the Java class name or **-2 points**.
6. Your program MUST handle bad commands; **-2 points** for each bad command not handled.
7. You CANNOT import any Java library classes, EXCEPT the **Scanner**, **StringTokenizer, Calendar,** and **DecimalForamt** class. **You will lose 5 points** for EACH additional Java library class imported, with a **maximum of -10 points**.
8. You CANNOT use the Java library class **ArrayList** anywhere in the project OR use any Java library classes from the Java Collections, or **you will get 0 points for this project**!
9. When importing Java library classes, be specific and DO NOT import unnecessary classes or the whole package. For example, `import java.util.*;` this will import all classes in the `java.util` package. You will **lose 2 points** for using the asterisk "*" to include all the Java classes in the `java.util` package, or other java packages, with a **maximum of -4 points.**
10. You MUST implement the Java classes below. **-5 points** for each class missing or NOT used. You should define necessary constant names in UPPERCASE letters and **DO NOT** use MAGIC NUMBERs, or **you will lose points** listed in the Coding Standard. You can use Java Enum classes or a public class to define all the constants.

    You CANNOT use System.in or System.out statements in ALL classes, EXCEPT the user interface class `Scheduler.java` or the testbed `main()` methods. **-2 points** for each violation, with a **maximum of -10 points.** You must always add the @Override tag for overriding methods or **-2 points** for each violation.

(a) **Appointment class**

```
public class Appointment implements Comparable <Appointment> {
    private Date     date;
    private Timeslot timeslot;
    private Profile  patient;
    private Provider provider;
}
```

- You can add necessary constants, constructors, and methods. However, you CANNOT change or add instance variables. **-2 points** for each violation.
- You MUST override **equals()**, **toString()** and the **comapreTo()** methods, with the @Override tags. **-2 points** for each violation. The **equals()** method returns true if two appointments have the same date, timeslot, and patient; returns false otherwise. The **toString()** method returns a textual representation of an appointment in the following formats.

  10/30/2024 10:45 AM John Doe 12/13/1989 [PATEL, BRIDGEWATER, Somerset 08807, FAMILY]

(b) **Date class**

```
public class Date implements Comparable<Date> {
    private int year;
    private int month;
    private int day;
    public  boolean isValid() //check if the date is a valid calendar date
}
```

- You can add necessary constants, constructors, and methods; however, you CANNOT change or add instance variables, **-2 points** for each violation.
- You MUST override **equals()**, **toString()** and the **comapreTo()** methods, with the @Override tags. **-2 points** for each violation.

- The **isValid()** method checks if a date is a valid calendar date.
  - For the months of January, March, May, July, August, October, and December, each has 31 days; April, June, September, and November each has 30 days; February has 28 days in a non-leap year, and 29 days in a leap year. DO NOT use magic numbers for the months, days, and years. You can use the constants defined in the `Calendar class` or define your own constant names. Below are examples for defining the constant names.

```
public static final int QUADRENNIAL = 4;
public static final int CENTENNIAL = 100;
public static final int QUATERCENTENNIAL = 400;
```

  - To determine whether a year is a leap year, follow these steps:

Step 1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
Step 2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
Step 3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
Step 4. The year is a leap year.
Step 5. The year is not a leap year.

- You MUST design the test cases to test the isValid() method thoroughly, or you will **lose 6 points.** Follow the instructions under the "Test Design" section in the Coding Standard and include the test cases in the test specification.
- You MUST include a **testbed main()** in this class or **lose 12 points**. You CAN use **System.out** in the testbed main() to display the test results.

(c) **Profile class**

```
public class Profile implements Comparable<Profile>{
    private String fname;
    private String lname;
    private Date   dob;
}
```

- You can add necessary constants, constructors, and methods; however, you CANNOT change or add the instance variables, or **-2 points** for each violation.
- You must override the **equals()**, **toString()** and **compareTo()**, with the **@Override** tag, or **-2 points** for each violation.
- You MUST design the test cases to thoroughly test the **compareTo()** method, or you will **lose 7 points**. The method compares last name, first name, and then dob for sorting. Follow the instructions under the "Test Design" section in the Coding Standard and include the test cases in the test specification.
- You MUST include a **testbed main()** in this class or **lose 8 points**. You CAN use **System.out** in the testbed main() to display the test results.

(d) **Enum classes.** You cannot add or change the properties or **-2 points** for each violation.

- **Timeslot class.** This Enum class defines the timeslots of a day on weekdays.

```
public enum Timeslot {
    SLOT1 (9, 0),
    SLOT2 (10, 45),
    SLOT3 (11, 15),
    SLOT4 (13, 30),
    SLOT5 (15, 0),
    SLOT6 (16, 15);
}
```

- **Provider class.** This Enum class defines the providers' last names with two additional properties listed below.

```
private final Location  location;
private final Specialty specialty;
```

- **Location class.** This Enum class defines the city locations with two additional properties listed below.

```
private final String county;
private final String zip;
```

- **Specialty class.** This Enum class defines the providers' specialties with an additional property "charge", which defines the charge per visit.

```
private final int charge;
```

(e) **List class**

- This is an array-based implementation of a linear data structure "List" to hold the list of appointment objects. A new appointment is always added to the end of the array. An instance of this class is a growable list with an initial array capacity of 4, and it automatically increases the capacity by 4 whenever it is full. The list does not decrease in capacity.

```
public class List {
    private Appointment[] appointments;
    private int size; //number of appointments in the array

    private int find(Appointment appointment) //helper method
    private void grow() //helper method to increase the capacity by 4
    public boolean contains(Appointment appointment)//check before add/remove
    public void add(Appointment appointment)
    public void remove(Appointment appointment)
    public void printByPatient()//ordered by patient profile, date/timeslot
    public void printByLocation()   //ordered by county, date/timeslot
    public void printByAppointment()//ordered by date/timeslot, provider name
}
```

- You can add necessary constants, constructors, and methods. However, you CANNOT change or add instance variables. **-2 points** for each violation.
- You MUST implement and use the methods listed above; you CANNOT change the signatures of the methods. **-2 points** for each violation.
- You CAN use **System.out** ONLY in the three **print()** methods listed above.
- The **find()** method searches for an appointment in the list and returns the index if it is found; it returns `NOT_FOUND` if it is not in the list. Define a constant value **-1** for `NOT_FOUND`, or **-2 point**.
- You must use an "in-place" sorting algorithm to implement the sorting, i.e., the order of the objects in the array will be rearranged after the sorting without using an additional array. You CANNOT use **Arrays.sort()** or **System.arraycopy()** or any other Java library classes or utilities for sorting. You must write the code yourself to sort it. You will **lose 10 points** for the violation.

(f) **Patient class**

```
public class Patient implements Comparable<Patient> {
    private Profile  profile;
    private Visit    visits; //a linked list of visits (completed appt.)
    public int charge() {} //traverse the linked list to compute the charge
}
```

- You can add necessary constants, constructors, and methods; however, you CANNOT change or add the instance variables, or **-2 points** for each violation.
- You must override the **equals()**, **toString()** and **compareTo()**, with the @Override tag, or **-2 points** for each violation.
- The **charge()** method traverses the linked list "visits" and returns the charge of the patient, **-3 points** if this method is missing.

(g) **Visit class –** This class defines a node in a singly linked list that maintains the list of visits. You can add constructors and methods, but you CANNOT add/change the instance variables, or **-2 points** for each violation.

```
public class Visit {
    private Appointment appointment; //a reference to an appointment object
    private Visit next; //a ref. to the next appointment object in the list
}
```

(h) **MedicalRecord class**

```
public class MedicalRecord {
    private Patient[] patients;
    private int size; //number of patient objects in the array
}
```

- This is an array-based implementation of a linear data structure, "Bag," to hold a list of patient objects. The bag allows adding a patient object but does not allow removal.
- You must implement an **add()** method, or **-2 points**.
- You can add constructors and methods, but you CAN NOT add/change the instance variables, or **-2 points** for each violation

(i) **Scheduler class**

- This is the user interface class to process the command lines entered on the terminal. An instance of this class can process a single command line or multiple command lines at a time, including the empty lines. **You will lose 10 points** if it cannot process multiple command lines.
- When your software starts running, it shall display `"Scheduler is running."`. Next, it will continuously read and process the command lines until the "Q" command is entered. If the Q command is entered, display `"Scheduler terminated",` then the software stops normally. **-2 points** for each violation.
- You must define a **run()** method that includes a while loop to continuously read the command lines until a "Q" command is entered. You will **lose 5 points** if the **run()** method is missing. You MUST keep this method **under 40 lines** for readability, or you will **lose 3 points**. You can define necessary instance variables and private helper methods to handle each command.

(j) **RunProject1 class.** This is a driver class that runs your software. The graders will run this class to grade your project.

```
public class RunProject1 {
    public static void main(String [] args) {
        new Scheduler().run();
    }
}
```

11. **Test Specification –** you must use the table template in the Coding Standard, **or you will lose 15 points for this part.** You lose **1 point** for each test case missing. Use a document editor, such as Microsoft Word or Google Docs. Do not copy and paste your code to this document; or you will **get 0 points for this part**. Include this document in your project folder.
    (a) Date class – design **four** invalid and **two** valid test cases for testing the **isValid()** method,
    (b) Profile class – design test cases to test the **compareTo()** method; **three** test cases return -1, **three** test cases return 1, and **one** test case returns 0.
12. Testbed **main()** is each Java class's main() method to exercise and test the public methods defined in the class. You MUST implement the test cases designed in the Test Specification above.
    (a) Date class – 6 test cases, **-2 points** for each test case missing.
    (b) Profile class – 7 test cases, **-1 point** for each test case missing.
13. You must **generate the Javadoc** after you properly comment your code. Set the scope to 'private" so your Javadoc will include the documentation for the private data members, constructors, and private and public methods of all Java classes.

    Generate the Javadoc in a single folder and include it in the zip file to be submitted to Canvas. **You are responsible for double-checking your Javadoc folder after you generate it. Submitting an empty folder will result in 0 points for this part.** Open the Javadoc folder and look for the **index.html** file. Double-click the file and check every link to ensure all comments are NOT EMPTY. You will lose points if any description in the Javadoc is empty. You will **lose 5 points** for not including the Javadoc.