Part C: Development

Techniques used

- Java
  - Simple selection (if-else)
  - Complex selection (nested if or if with multiple conditions)
  - Loops
  - Nested loops
  - User-defined methods
  - User-defined methods with parameters
  - User-defined methods with appropriate return values
  - User-defined objects
  - Objects as data records
  - Arrays
  - Searching
  - File i/o
  - Use of additional libraries
- Product Extensibility
  - Meaningful file names
  - Meaningful variable names
  - Indentation and comments

Java

Technique	Description	Evidence

Simple selection (if-else)	Returns true if the driver's String of preferred days includes the day's day of the week.	`public boolean driverHasDay (int index, LocalDate day) {`
		`  if(drivers[index].getDays().indexOf(day.getDayOfWeek().getDisplayName`
		`  (TextStyle.FULL, Locale.ENGLISH).toUpperCase())>-1)`
		`  return true;`
		`  else`
		`  return false;`
		`} //Finds whether the driver can drive the day's day of the week`

Complex selection (nested if or if with multiple conditions)	It labels the day "WEEKEND" if the day is and then if the day is a Sunday it increments the current week. If the day is off, it labels the day "NO CARPOOL."	`if(currentDayVal==6||currentDayVal==7) {`
		`  schedule[currentWeek][currentDayVal-1] = "WEEKEND";`
		`  if(currentDayVal==7) currentWeek++;`
		`}`
		`else if(dayIsOff(currentDay, daysOff)) {`
		`  schedule[currentWeek][currentDayVal-1] = "NO CARPOOL";`
		`}`

Loops	For each driver in the drivers array, the name and the number of turns is printed.	`public void printTotalTurns() {`
		`  System.out.print("\n\n");`
		`  for(int i=0; i<drivers.length; i++) {`
		`  System.out.println(drivers[i].getName() + ": " + drivers[i]`
		`  .getTurns());`
		`  }`
		`} //Prints the total turn counts for each driver`

Nested loops	Prints each row of the schedule and for each row, each column is printed. New lines and the week's number are printed between rows.	`public void printSchedule() {`
		`  for(int i=0; i<schedule.length; i++) {`
		`  System.out.println();`
		`  System.out.printf("Week: %-10s", (i+1));`
		`  for(int j=0; j<schedule[0].length;j++) {`
		`  System.out.printf("%-20s", schedule[i][j]);`
		`  }`
		`  }`
		`} //Prints the schedule with each week on a seperate line`

User-defined methods	There is a getter method for returning the driver's name.	`public String getName() {`
		`  return name;`
		`} //Gets the name`

User-defined methods with parameters	Returns the total number of days between the parameters firstDay and lastDay.	`public int calcTotalDays(LocalDate firstDay, LocalDate lastDay) {`
		`  return (int)ChronoUnit.DAYS.between(firstDay, lastDay) + 1;`
		`} //Calculates the number of days between two dates inclusive`

User-defined methods with appropriate return values	Returns a boolean of whether the day is included in the daysOff arrayList.	`public boolean dayIsOff (LocalDate day, ArrayList<LocalDate> daysOff) {`
		`  if(daysOff.size()!=0&&day.equals(daysOff.get(0))) {`
		`  daysOff.remove(0);`
		`  return true;`
		`  }`
		`  return false;`
		`} //Checks whether the day is included on the list of daysOff`

User-defined objects	Driver objects are defined to have private variables name, days, turns, and turnsList. These variables are defaulted in the default constructor.	`public class Driver {`
		`  private String name;`
		`  private String days;`
		`  private int turns;`
		`  private ArrayList<LocalDate> turnsList;`
		
		`  public Driver() {`
		`  name = "";`
		`  days = "";`
		`  turns = 0;`
		`  turnsList = new ArrayList<LocalDate>();`
		`  } //Default constructor`

Objects as data records	The driver's name and preferred days are saved into the driver object's private Strings name and days.	`System.out.print("\nDriver "+(i+1)+" name: ");`
		`drivers[i].setName(kb.nextLine());`
		`System.out.print("Days "+drivers[i].getName()+ " prefers to drive`
		`  : ");`
		`drivers[i].setDays(kb.nextLine().toUpperCase());`

Arrays	The schedule is a 2D array of Strings, where each cell is filled with the driver's name or another label. drivers is a 1D array of Driver objects that contains all the drivers in the carpool.	`private String[][] schedule;` `private Driver[] drivers;`

Searching	The array of drivers is searched for the driver who can drive on a certain day and has the least turns. If a driver has less turns than the current best driver, then that driver becomes the best driver.	`int bestDriver=-1;`
		`for(int i=0; i<drivers.length; i++) {`
		`  if(driverHasDay(i, day)) {`
		`  if(bestDriver==-1||drivers[i].getTurns()<drivers[bestDriver]`
		`  .getTurns()) {`
		`  bestDriver = i;`
		`  }`
		`  }`
		`}`
		`return bestDriver;`

File i/o	A file of Strings of dates is imputed and filled into an arrayList of LocalDates.	`public static ArrayList<LocalDate> createDateList(File data, Scanner sc,`
		`  DateTimeFormatter formatter) throws FileNotFoundException {`
		`  ArrayList<LocalDate> days = new ArrayList<LocalDate>();`
		`  while(sc.hasNextLine()) {`
		`  LocalDate day = LocalDate.parse(sc.nextLine(), formatter);`
		`  days.add(day);`
		`  }`
		`  return days;`
		`} //creates an ArrayList of LocalDates from a text file of Strings`

Use of additional libraries	java.io.File is used to work with files and java.util.Scanner is used to scan inputs from the file or the user.
	java.util.ArrayList is used to create arrayLists and java.util.LocalDate creates LocalDate.
		`import java.io.File;`
		`import java.io.FileNotFoundException;`
		`import java.util.Scanner;`
		`import java.util.ArrayList;`
		`import java.time.LocalDate;`
		`import java.time.temporal.ChronoUnit;`
		`import java.time.format.DateTimeFormatter;`
		`import java.time.format.TextStyle;`
		`import java.util.Locale;`

Product Extensibility

Technique	Description	Evidence
Meaningful file names	Files are based on what they are used for. The daysOff text file is where the String of days off is inputted and the Runner is where the code to run the program is. The Driver class has information for driver objects and the Carpool class contains the schedule array and drivers array.	daysOff.txt **Runner.java** Driver.java Carpool.java
Meaningful variable names	The Carpool's schedule is named schedule and the array of Drivers is named divers.	`private String[][] schedule;` `private Driver[] drivers;`

Indentation and comments	The code is indented properly and the method is commented at the end.	`public int bestDriver (LocalDate day) {`
		`  int bestDriver=-1;`
		`  for(int i=0; i<drivers.length; i++) {`
		`  if(driverHasDay(i, day)) {`
		`  if(bestDriver==-1||drivers[i].getTurns()<drivers[bestDriver]`
		`  .getTurns()) {`
		`  bestDriver = i;`
		`  }`
		`  }`
		`  }`
		`  return bestDriver;`
		`} //Finds the driver who can drive the day and has the least turns`

Appendix C

Third client interview

Developer: I've finished the prototype for the Carpool Scheduler. Would you like to take a look at it?

Client: Sure, I'm so excited to see it!

Developer: You have to input the first and last day of the school schedule, the days off in the school schedule, and the carpool drivers' name and their preferred days. Is that okay with you?

Client: That seems alright.

(Client inputs information into the program)

Client: Wow that works so fast! The turns are split so evenly and everyone only has to drive their preferred days. This is great!

Developer: I'm glad you like it! Is there anything that you would like me to add?

Client: If you could, would it be possible to create a list for each driver of the days that they have to drive?

Developer: Ok, I'll add that in and then send you a video demonstrating how the product works.

Client: Sounds good!