```java

/*
 * Mitch Feigenbaum
 * Period 5
 * February 1, 2022
 * On my honor, I pledge that I have neither given nor received unauthorized assistance on this assignment or test.
 */
import java.util.Calendar;
import java.util.Scanner;

/**
 * The FeigenbaumU4 class is a class that provides methods for checking if a
 * given month and date are valid, getting a calendar date in ordinal form,
 * getting an astrological sign with a message, and a CLI in which a client
 * can run the program (the main method).
 */
public class FeigenbaumU4 {
	/**
	 * The main method is a CLI that allows a client to interactively
	 * input their birthdate to get an output of their astrological sign
	 * with a unique message. The method first creates a Calendar object.
	 * This calendar object is used to store the current month and
	 * date into the todaymonth and todayday variables. A scanner is
	 * opened to collect the clients input. Then the program enters
	 * a do while loop that prompts the client to enter their birthdate,
	 * checks that the date is valid with the checkDate method, and stores
	 * The client's birthmonth and birthdate into separate integer
	 * variables. After the client's birthmonth and birthday are defined,
	 * the program checks if it is the client's birthday and if so
	 * initializes a birthday message. Finally a print format statement is
	 * created that displays the client's age in ordinal form (using the
	 * birthdate method), a birthday message in the event that the current
	 * date is their birthday, their astrological sign, and a unique
	 * message (using the sign method)
	 * <p>
	 *
	 * @param args added for semantics
	 * @see checkDate
	 * @see birthdate
	 * @see sign
	 */
	public static void main(String[] args) {
		Calendar today = Calendar.getInstance();
		int todaymonth = today.get(Calendar.MONTH) + 1;
		int todayday = today.get(Calendar.DAY_OF_MONTH);
		Scanner scanNum = new Scanner(System.in);
		int birthMonth;
		int birthDay;
		do {
			System.out.print("What month were you born in? (number): ");
			birthMonth = scanNum.nextInt();
			System.out.print("What day (number): ");
			birthDay = scanNum.nextInt();
			if (!checkDate(birthMonth, birthDay))
				System.err.println("Error: date does not exist.");
		} while (!checkDate(birthMonth, birthDay));
		String birthDayMessage = "";
		if (birthMonth == todaymonth && birthDay == todayday)
			birthDayMessage = "Happy Birthday to you!";
		System.out.printf("Your birthday is:\t%s\t%s%n%s",
				birthdate(birthMonth, birthDay), birthDayMessage, sign(birthMonth, birthDay));
	}

	/**
	 * The birthdate method is used to return a textual version of a date
	 * based on an integer representation of the month and date. First an
	 * array of all 12 calendar months is initialized. then a String
	 * format statement is returned that contains a textual representation
	 * of the month, and an ordinal representation of the date. (using the
	 * toOrdinal method)
	 * <p>
	 *
	 * @param m An integer representation of the month
	 * @param d An integer representation of the date
	 * @return a formatted string containing the month and date
	 * @see toOrdinal
	 */
	public static String birthdate(int m, int d) {
		String[] months = {
```

```
 80                                  "January",
 81                                  "February",
 82                                  "March",
 83                                  "April",
 84                                  "May",
 85                                  "June",
 86                                  "July",
 87                                  "August",
 88                                  "September",
 89                                  "October",
 90                                  "November",
 91                                  "December",
 92                          };
 93                          return String.format("%s %s", months[m - 1], toOrdinal(d));
 94                  }
 95
 96                  /**
 97                   * The method sign provides an astrological sign and horoscope based
 98                   * on a provided month and date. The method iterates through a list of
 99                   * if statements that return the astrological sign and horoscope of a
100                   * certain character (eg Aquarius) provided that the month and date
101                   * are within a certain range of dates in which the character is
102                   * valid. If the criteria of an if statement is fulfilled the method
103                   * returns a string with an astrological sign and horoscope formatted
104                   * with tabs for alignment. The tabstops are meant to be used in
105                   * conjunction with the main method's print format statement which
106                   * also displays the users birthday and a happy birthday message.
107                   * <p>
108                   *
109                   * @param m An integer representation of the month
110                   * @param d An integer representation of the date
111                   * @return A string with the astrological sign and horoscope for the
112                   *         provided month and day
113                   * @see main
114                   */
115                  public static String sign(int m, int d) {
116                          return (m == 1 && d >= 20) || (m == 2 && d <= 18)
117                                  ? "Your sign is:\t\tAquarius\n" + "Horoscope:\t\tYou must hold back your desires
and temptations."
118                                  : (m == 2 && d >= 19) || (m == 3 && d <= 20)
119                                  ? "Your sign is:\t\tPisces\n" + "Horoscope:\t\tSoon you will make amends with all
you have wronged."
120                                  : (m == 3 && d >= 21) || (m == 4 && d <= 19)
121                                  ? "Your sign is:\t\tAries\n" + "Horoscope:\t\tYour home planet will be destroyed."
122                                  : (m == 4 && d >= 20) || (m == 5 && d <= 20)
123                                  ? "Your sign is:\t\tTaurus\n" + "Horoscope:\t\tHave faith in the plan."
124                                  : (m == 5 && d >= 21) || (m == 6 && d <= 20)
125                                  ? "Your sign is:\t\tGemini\n" + "Horoscope:\t\tThe best days of your life are
already over."
126                                  : (m == 6 && d >= 21) || (m == 7 && d <= 22)
127                                  ? "Your sign is:\t\tCancer\n" + "Horoscope:\t\tYou will hold back your own
success."
128                                  : (m == 7 && d >= 23) || (m == 8 && d <= 22)
129                                  ? "Your sign is:\t\tLeo\n" + "Horoscope:\t\tYou will achieve great success in the
textile industry."
130                                  : (m == 8 && d >= 23) || (m == 9 && d <= 22)
131                                  ? "Your sign is:\t\tVirgo\n" + "Horoscope:\t\tThere is light at the end of your
sorrow."
132                                  : (m == 9 && d >= 23) || (m == 10 && d <= 22)
133                                  ? "Your sign is:\t\tLibra\n" + "Horoscope:\t\tTerrible things are going to
happen."
134                                  : (m == 10 && d >= 23) || (m == 11 && d <= 21)
135                                  ? "Your sign is:\t\tScorpio\n" + "Horoscope:\t\tYour days are numbered. Keep watch
of all enemies."
136                                  : (m == 11 && d >= 22) || (m == 12 && d <= 21)
137                                  ? "Your sign is:\t\tSagittarius\n" + "Horoscope:\t\tSomeday you will travel into
the multiverse."
138                                  : (m == 12 && d >= 22) || (m == 1 && d <= 19)
139                                  ? "Your sign is:\t\tCapricorn\n" + "Horoscope:\t\tYou will achieve inner peace."
140                                  : null;
141                  }
142
143                  /**
144                   * The toOrdinal method can convert a number (from 1 to 39) from
145                   * integer form into ordinal form. Based on a number's position in a
146                   * certain range (eg 11-19), an element is chosen from an array which
147                   * represents a number's ordinal form. If a number is not composed of
148                   * only one word (eg twenty-sixth), a format string is returned
149                   * containing the numbers tenth place, a dash, and the singular
150                   * number that represents the digit in its ones place.
151                   * <p>
```

```
152          *
153          * @param d a digit 1-39
154          * @return an ordinal number
155          */
156         public static String toOrdinal(int d) {
157                 String[] singleDays = {
158                                 "first",
159                                 "second",
160                                 "third",
161                                 "fourth",
162                                 "fifth",
163                                 "sixth",
164                                 "seventh",
165                                 "eighth",
166                                 "ninth",
167                 };
168                 String[] teens = {
169                                 "eleventh",
170                                 "twelfth",
171                                 "thirteenth",
172                                 "fourteenth",
173                                 "fifteenth",
174                                 "sixteenth",
175                                 "seventeenth",
176                                 "eighteenth",
177                                 "nineteenth",
178                 };
179                 String[] tenths = {
180                                 "tenth",
181                                 "twentieth",
182                                 "thirtieth",
183                 };
184                 String[] tens = {
185                                 "twenty",
186                                 "thirty",
187                 };
188                 return (d < 10)
189                                 ? singleDays[d - 1]
190                                 : (d > 10 && d < 20)
191                                 ? teens[d - 11]
192                                 : (d % 10 == 0)
193                                 ? tenths[d / 10 - 1]
194                                 : String.format("%s-%s", tens[d / 10 - 2], singleDays[d % 10 - 1]);
195         }
196
197         /**
198          * The checkDate method verifies whether a current month and date are
199          * possible calendar positions. It iterates through a series of
200          * impossible criteria that will return false if they are applicable
201          * to the provided month and day. The date is first checked that
202          * the month is between 1 and 12 and that the date is at least one.
203          * Then the date is checked to verify that it is not too large for the
204          * current calendar month. If none of the unverifiable criteria are
205          * met, the method will return true
206          * <p>
207          *
208          * @param m the current month in integer form
209          * @param d the current day in integer form
210          * @return true if the provided month and day are valid
211          */
212         public static boolean checkDate(int m, int d) {
213                 return !(m < 1
214                                 || m > 12
215                                 || d < 1
216                                 || d > 31
217                                 || ((m == 4
218                                                 || m == 6
219                                                 || m == 9
220                                                 || m == 11)
221                                                 && d > 30)
222                                 || (m == 2
223                                                 && d > 29));
224         }
225 }
```