Lab 5: Birthday Priorities

Pre-Lab Notes

Lab submissions will **not be accepted late** – they must be submitted by **midnight the day after they're assigned** (to account for time zone issues). Some latitude might be allowed (I won't disqualify a submission that's mere minutes late!) but that's purely discretionary.

Labs are treated differently from assignments – you're encouraged to work with fellow students, ask questions of the TA during your lab section, and generally collaborate to complete your work. You should still submit your own work, **including citing with whom you collaborated in your submission**, but using answers developed by others will not be treated as plagiarism.

Nevertheless, you are still encouraged to make a good faith effort to complete your labwork, in order to exercise your understanding of the topics it covers. Lab submissions are marked purely 1 or 0 by whether the TA believes you at least made an honest attempt, even if you didn't succeed, so the value is in what the attempt teaches you rather than simply having the right answer.

1. Priority Queues

Key-based data structures, like Priority Queues, allow us to define our own rules for which element of data our structures should return to us next, instead of using their location within the structure. This sounds simple enough, but once we take on the responsibility for deciding how data should be compared and ordered, we can end up creating a lot of work for ourselves (and a lot of potential for error).

Within each data entry is a key and a value. Sometimes these are the same thing, such as a standard Java PriorityQueue of integers, where each number is both the key used to order the data and the data value returned when asked. Other times, the key is embedded within an object, or else is an entire object itself, and the PriorityQueue will need more specific guidance to know how to compare them. That's exactly what today's lab will have you practice.

2. Outline

The goal of today's lab will be to finish a piece of code meant to take in birthdays and prioritize them according to whoever has the earliest birthday in the year. A well-meaning teammate of yours has already taken a first crack at the problem, writing the BirthdayEvent class and running a quick test that was meant to show a standard Java PriorityQueue printing them in chronological order, only for the code to throw a nasty error at run-time. It seems their PriorityQueue doesn't know how to compare BirthdayEvent objects.

Stumped, your teammate's handed their code off to you, in hopes that you can figure out what to do next.

```java
import java.util.PriorityQueue;
class BirthdayEntry{
	int day;
	String month;
	String name;
	public BirthdayEntry(int dayIn, String monthIn, String nameIn)
	{
		day = dayIn;
		month = monthIn;
		name = nameIn;
	}
	public void announceBirthday()
	{
		System.out.println("I'm " + name + ", and my birthday is " + day + " " + month);
	}
}

public class Main {
	public static void main (String[] args)
	{
		BirthdayEntry firstDate = new BirthdayEntry(13, "April", "John");
		BirthdayEntry secondDate = new BirthdayEntry(4, "December", "Rose");
		BirthdayEntry thirdDate = new BirthdayEntry(19, "February", "Dave");
		BirthdayEntry fourthDate = new BirthdayEntry(20, "December", "Jade");
		PriorityQueue<BirthdayEntry> birthdaysPQ = new PriorityQueue<BirthdayEntry>();
		birthdaysPQ.add(firstDate);
		birthdaysPQ.add(secondDate);
		birthdaysPQ.add(thirdDate);
		birthdaysPQ.add(fourthDate);
		birthdaysPQ.poll().announceBirthday();
		birthdaysPQ.poll().announceBirthday();
		birthdaysPQ.poll().announceBirthday();
		birthdaysPQ.poll().announceBirthday();
	}
}
```

2.1 Where You Come In

Your project supervisor has asked you to keep working with BirthdayEntry and the test code they've written so far, just to pick up where your teammate left off. You can define new classes or functions as needed, as well as expand the main method, but ultimately the birthdaysPQ object must begin working as intended. Be sure to add a few more BirthdayEntry's to double-check you've solved the issue!

3. Deliverables

All lab submissions will be done through CourSys at https://courses.cs.sfu.ca/. For this lab, that should include:

1. A .zip archive of your code, organized into a single Java project. To ensure compatibility, it's recommended to use IntelliJ while developing your code, then using the export function under File -> Export -> Project To .zip File. **Don't forget to include a test file**, a Main.java with a main method that prints the results of a few runs of your functions to the terminal would be best.

Be sure you tidy up your code before submitting! Check the Java code style guide posted on the course website, and do your best to provide helpful comments.