COP2805C Module 6 Practice Exercise

In this practice exercise we will use Java streams and resource bundles to create a "magic pipeline" through which we can simulate consuming/processing our dispatched birthday cards (for now we will only be printing them out). We will also print the current date in a localized format and localized Happy Birthday greeting at the top of each card based on the user's locale (a new property added to the User class).

Creating the Stream

A Stream can be created from a collection, this is done in the form of a new instance variable in the HappyBirthdayApp class using the Queue as the source:

```
private Queue<BirthdayCard> queue = new LinkedList<BirthdayCard>();
private Stream<BirthdayCard> stream = queue.stream();
```

After dispatching the cards to the queue, the main method then uses the Stream to iterate through the contents:

```
System.out.println("starting forEach");
hba.stream.forEach(System.out::print);
```

I added a toString override method to the BirthdayCard class to print out the contents of the card, this is what you see as output from the call to forEach.

```
@Override
public String toString() {
    String s = "Birthday card for " + this.getUser().getName() + "\n";
    s += getMessage();
    return s;
}
```

Think about how the consumer could do a lot more with this mechanism; one possibility would be accessing APIs to actually send the message out via SMS or email.

Localizing the Output

We now want to print the current date at the top of our birthday card and localize it based on the user's locale. A Locale property has been added to the User class for this, which also now provides a getter for that property.

Here's the code to do that, using a DateTimeFormatter which is customized by locale and applied to the user's ZonedDateTime property. There are a variety of styles we could apply to the date, I've chosen the basic FormatStyle enum value of MEDIUM which includes the name of the month (https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/format/FormatStyle.html)

```
DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM);
formatter =
```

```
            formatter.localizedBy(u.getLocale());
    msg = u.getBirthday().format(formatter) + "\n";
```

For the localization of the birthday greeting, there are four property files provided in the solution in the form of a "Birthday" resource bundle:

Birthday_en.properties -- US English birthday greeting
Birthday_fr.properties -- French birthday greeting
Birthday_de.properties -- German (Deutsch) birthday greeting
Birthday_zh.properties --Chinese (Mandarin) birthday greeting

Examine the contents of these files to see how the greeting property is implemented and note the location of the files. If you create the files in a package, IntelliJ will recognize the file extensions and group them as a resource bundle for you.

Creating a property is easy since they are simple key/value combinations, but some research is required to determine the value (see below). This exercise is a very trivial implementation, we are translating only the initial "Happy Birthday" greeting. The full birthday card greeting is not translated since that would require much more expertise in the grammatical forms of each language; developers must tread carefully here to avoid awkward or inappropriate translations. We usually depend on a localization group (either in-house or contracted) to do this for us since language fluency is required. Even if you have had a few courses in a foreign language or speak a "smattering" of a second language, do not volunteer for this task; trust me on this.

The German and French translations of "Happy Birthday" are simple to find on the Web:

German: Alles Gute zum Geburtstag
French: Joyeux Anniversaire

The Chinese version takes a little more work since we have to use Unicode values to represent the phrase in Mandarin (or "Hànzì" as the symbols are called). I used the following sites:

https://chinainternshipplacements.com/blog/happy-birthday-in-chinese-mandarin/
https://www.chineseconverter.com/en/convert/unicode

which gave me the following:

生日快乐
\u751f \u65e5 \u5feb \u4e50

The main method for the HappyBirthdayApp class has been modified to do the localization for the date as described above and for the greeting (this should probably be moved into the buildCard method). For the greeting the code attempts to load the resource bundle and find the corresponding properties file

for the user's locale, then looks up the key in the file using the ResourceBundle class's getString method. It then uses the resulting string for the greeting.

```
// see if today is their birthday
// if not, show sorry message
if (!hba.isBirthday(u))
    System.out.println("Sorry, today is not their birthday.");
// otherwise build the card
else {
    String msg = "";
    try {

        // load the property and create the localized greeting
        ResourceBundle res = ResourceBundle.getBundle(
                "edu.fscj.cop2805c.birthday.Birthday", u.getLocale());
        String happyBirthday = res.getString("HappyBirthday");

        // format and display the date
        DateTimeFormatter formatter =
                DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM);
        formatter =
                formatter.localizedBy(u.getLocale());
        msg = u.getBirthday().format(formatter) + "\n";

        // add the localized greeting
        msg += happyBirthday + " " + u.getName() + "\n" +
                BirthdayCard.WISHES;
    } catch (java.util.MissingResourceException e) {
        System.err.println(e);
        msg = "Happy Birthday, " + u.getName() + "\n" +
                BirthdayCard.WISHES;
    }
    BirthdayCard bc = hba.buildCard(u, msg);
    hba.sendCard(bc);
}
```

Notice that if we don't find the resource bundle, we have a safety net in the catch clause which simply uses the English version of the greeting.

Existing users were modified to include the locale in the constructor call and more users were created in the main method to test each locale:

```
// negative test
User u1 = new User("Dianne", "Romero", "Dianne.Romero@email.test",
        new Locale("en"), currentDate.minusDays(1));

// positive tests
// test with odd length full name and english locale
User u2 = new User("Sally", "Ride", "Sally.Ride@email.test",
        new Locale("en"), currentDate);

// test french locale
User u3 = new User("René", "Descartes", "René.Descartes@email.test",
        new Locale("fr"), currentDate);
```

```
// test with even length full name and german locale
User u4 = new User("Johannes", "Brahms", "Johannes.Brahms@email.test",
        new Locale("de"), currentDate);

// test chinese locale
User u5 = new User("Charles", "Kao", "Charles.Kao@email.test",
        new Locale("zh"), currentDate);
```

Here is the output from the program. I am no longer displaying "sending" messages now to reduce the clutter since we know that part of the code is working correctly from Module 5. We would want to continue to include those tests in our real-world test environment, however, to avoid a regression defect after adding our new features.

```
Here are the users:
Dianne Romero
Sorry, today is not their birthday.
Sally Ride
René Descartes
Johannes Brahms
Charles Kao
starting forEach
Birthday card for Sally Ride
|---------------------------------------------|
|                 Feb 27, 2023                |
|           Happy Birthday Sally Ride         |
|  Hope all of your birthday wishes come true! |
|---------------------------------------------|
Birthday card for René Descartes
|---------------------------------------------|
|                 27 févr. 2023               |
|      Joyeux Anniversaire René Descartes     |
|  Hope all of your birthday wishes come true! |
|---------------------------------------------|
Birthday card for Johannes Brahms
|---------------------------------------------|
|                  27.02.2023                 |
|   Alles Gute zum Geburtstag Johannes Brahms  |
|  Hope all of your birthday wishes come true! |
|---------------------------------------------|
Birthday card for Charles Kao
|---------------------------------------------|
|                  2023年2月27日                  |
|               生日快乐 Charles Kao              |
|  Hope all of your birthday wishes come true! |
|---------------------------------------------|
```

Oops! Notice that there is now a defect in the padding for the Chinese version of the card.
If you look closely at the vertical alignment of the Hànzì symbols you can see that they are slightly wider than the latin letters and numbers. Normally we would be displaying the characters in a GUI which would allow us to make pixel-level adjustments. We could play around with a scaling factor to adjust the text padding, but it is not a problem we will worry about here.

Another problem to note is that the symbols display correctly in the output pane of IntelliJ, but they won't display correctly in a Command Tool or PowerShell Window; instead you will see question marks (but the padding will be correct!)

```
|                  2023?2?27?                |
|              ???? Charles Kao              |
```

We could fix this by loading the appropriate language pack in Windows, but that's another problem we won't deal with in this course.