COP3330C Module 9 Practice Exercise

In this exercise we will create modules for our HappyBirthdayApplication, then build and execute the application using the modules.

NOTE: Do not try to use IntelliJ (or any other IDE, for that matter) to build or execute this practice exercise, follow the instructions included below and use the command line tools. This is a "throw-away" exercise to practice building modules, we are forcing the project to use independent compilation units that it is not really intended to support.

You will need to follow this procedure to complete the graded assignment as well.

Design Notes

Using the following folder structure of our application, we are adding a module-info.java file for each package:

```
root-directory (one level above edu)
      |
      |--mods
      |
      |--edu.fscj.cop3330c.birthday
      |   |-- (source files)
      |   |-- module-info.java
      |
      |--edu.fscj.cop3330c.dispatch
      |   |-- (source files)
      |   |-- module-info.java
      |
      |--edu.fscj.cop3330c.image
      |   |-- (source files)
      |   |-- module-info.java
```

Here's the content of birthday\module-info.java:

```
module edu.fscj.cop3330c.birthday {
   requires edu.fscj.cop3330c.dispatch;
   requires edu.fscj.cop3330c.image;

   exports edu.fscj.cop3330c.birthday;
}
```

Here's dispatch\module-info.java

```
module edu.fscj.cop3330c.dispatch {
   exports edu.fscj.cop3330c.dispatch;
}
```

And finally, image\module-info.java

```
module edu.fscj.cop3330c.image {
   exports edu.fscj.cop3330c.image;
}
```

The syntax to build the modules is tricky. We will build each module separately, starting with the lower-level dependencies and then the application module:

```
javac -p mods -d mods/edu.fscj.cop3330c.dispatch edu/fscj/cop3330c/dispatch/*.java
javac -p mods -d mods/edu.fscj.cop3330c.image edu/fscj/cop3330c/image/*.java
javac -p mods -d mods/edu.fscj.cop3330c.birthday edu/fscj/cop3330c/birthday/*.java
```

Notice we are using the short form of "-p" (one dash) instead of "--module-path" (two dashes) to specify where the modules will be built.

After building the modules, here is what the folder structure looks like (notice the new mods folder content):

```
root-directory (one level above edu)
      |
      |--mods
      |   |--edu.fscj.cop3330c.birthday
      |   |   |--(class files)
      |   |   |--module-info.class
      |
      |   |--edu.fscj.cop3330c.dispatch
      |   |   |--(class files)
      |   |   |--module-info.class
      |
      |   |--edu.fscj.cop3330c.image
      |   |   |--(class files)
      |   |   |--module-info.class
      |
      |--edu.fscj.cop3330c.birthday
      |   |--(source files)
      |   |--module-info.java
      |
      |--edu.fscj.cop3330c.dispatch
```

```
|   |--(source files)
|   |--module-info.java
|
|--edu.fscj.cop3330c.image
|   |--(source files)
|   |--module-info.java
```

Finally, we can run the application using the following command. We have to first set the Java CLASSPATH environment variable so the resource bundle's property files can be found.

```
set CLASSPATH=.
java -p mods -m edu.fscj.cop3330c.birthday/edu.fscj.cop3330c.birthday.HappyBirthdayApp
```