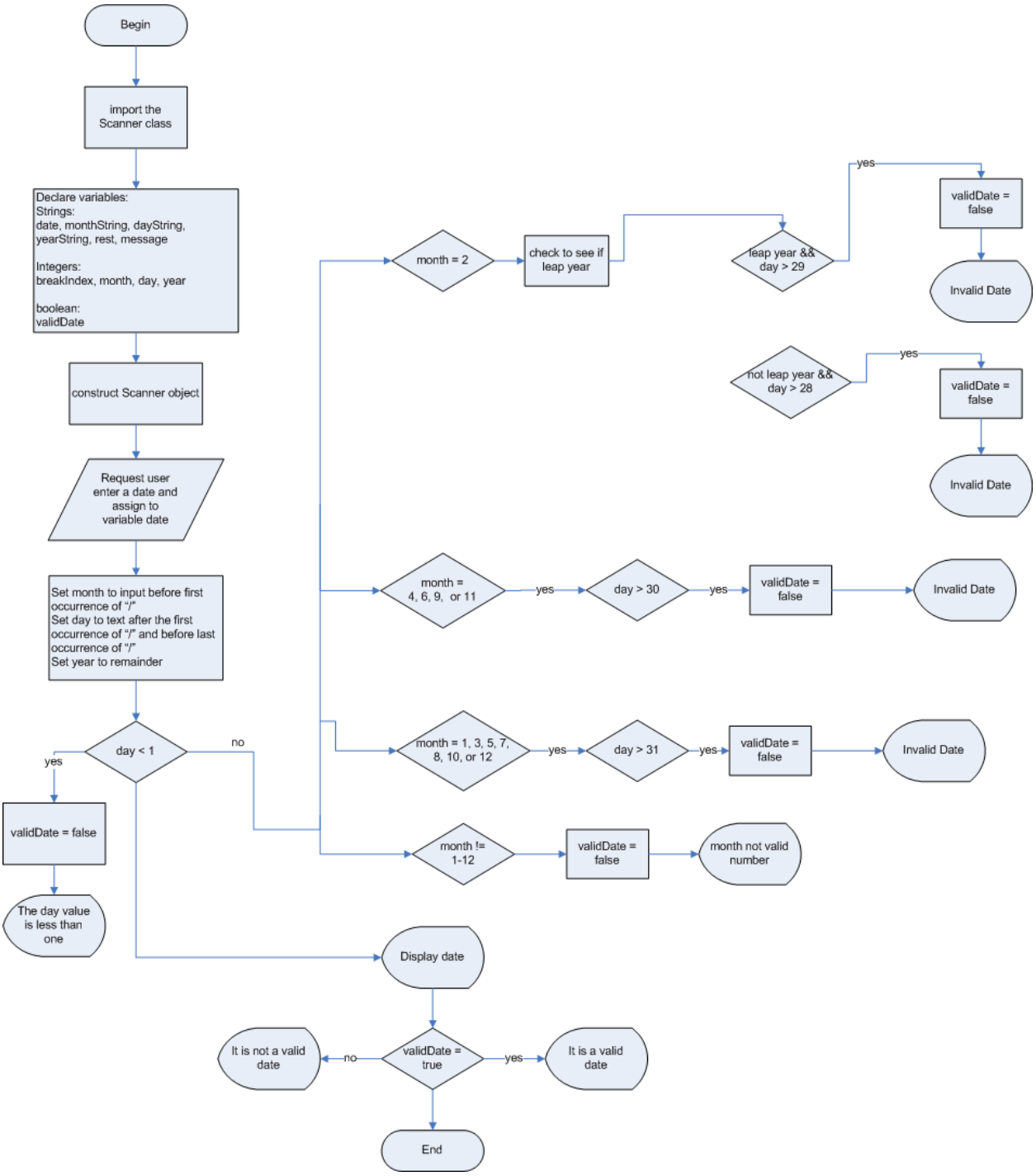
CIT 149: Java I
Chapter 3 Lab 2

In this lab we will request that the user enter a date and check to see if the date is valid. The flowchart for this program is:



1. Open a new document in TextPad and save the program as CheckDate.java.
2. Type the appropriate block comments that will include your name, the date, and the purpose of the program. The purpose is to determine a valid date.
3. Type the following import statement that will import the Scanner class:

   import javax.swing.JOptionPane;

4. Type the class header and opening brace.
5. Type the main method header and opening brace.
6. Type the following code that will declare several variables.

```
String date, monthString, dayString, yearString, rest, message=" ";
int breakIndex, month, day, year;
boolean validDate= true, leapYear;
```

7. For the *message* and *validDate* variables we gave initial values. When using if statements where a variable may or may not receive a value it is necessary to give default values to the variable(s). Once we create an if statement you will see the necessity of this.
8. Instead of requesting information at the command prompt we will use the JOptionPane methods. Type the following code that will request input from the user and assign their response to the variable *date*:

```
date = JOptionPane.showInputDialog(null,"Please enter a date to be checked in the form of mm/dd/year");
```

9. Using the indexOf() method we will check for the first occurrence of "/" and assign the content before this occurrence to the variable *monthString*. Type:

```
breakIndex = date.indexOf("/");
monthString =  date.substring(0, breakIndex);
```

10. All this was done previous you should not need any additional explanation. Since we cannot check to see if the month is a valid month for a String we have to convert this to an integer. Type:

```
month = Integer.parseInt(monthString);
```

    - The parseInt() method converts its argument to an integer data type. Here we assign this conversion to the variable *month*.

11. As done previously we need to assign other parts of the *date* to the variables of *day* and *year*. Type:

```
rest = date.substring(breakIndex+1);
breakIndex = rest.indexOf("/");

dayString = rest.substring(0, breakIndex);
yearString = rest.substring(breakIndex+1);
```

    - Notice that we assign these values to String variables. This means we will have to convert the values to integers as well. Type:

```
day = Integer.parseInt(dayString);
year = Integer.parseInt(yearString);
```

12. Our next step is to print out the entire date by typing:

```
JOptionPane.showMessageDialog(null,"date is " + month + ":" + day + ":"+ year );
```

13. We will now check to see if the *day* entered is less than 1. This would mean that it is an invalidate date. Type:

```
if(day<1)
{
   validDate = false;
   message = "The day value is less than one.  ";
}
```

15. In the above code we set new values for the variables of *validDate* and *message* if the variable *day* is less than 1. Since the variables of *validDate* and *message* may or may not receive a value we had to give them default values early in the program. If this is not done, an error will result.
16. Next we will create a switch statement by typing:

```
switch(month)
{
   case 2:  // February
      leapYear  = (year%400 == 0) || ((year%4 == 0) && (year%100 != 0));
      if(leapYear && day>29)
      {
         validDate = false;
         message = "The day value is greater than 29 in February in a leap year.  ";
      }
      else if(!leapYear && day>28)
      {
         validDate = false;
         message = "The day value is greater than 28 in February in a non leap year.  ";
      }
      break;
   case 4:  case 6:  case 9:  case 11:  // April, June, September, and November
      if(day>30)
      {
         validDate = false;
         message = "The day value is greater than 30 in a month with just 30 days.  ";
      }
      break;
   case 1: case 3: case 5: case 7: case 8: case 10: case 12: // January, March, May, July,
August, October, and December
      if(day>31)
      {
```

```
            validDate = false;
            message = "The day value is greater than 31 in a month with just 31 days.  ";
          }
          break;
       default:
          validDate = false;
          message = message + "The month value is not from 1 to 12.  ";

   }
```

- Notice that the cases within the switch statement are not in any particular order? The order of cases does not matter in the least. We also include multiple cases together since they will be responsible for the same things. Let's go through each:
    - for case 2 (which means of the variable *month* equals 2) the variable *leapYear* equals true or false depending on certain things.
        - if the result of the variable *year* divided by 400 is 0 *leapYear* will be set to true.
        - if the result of the variable *year* divided by 4 is 0 *leapYear* will be set to true
        - if the result of the variable *year* divided by 100 does not equal 0 then *leapYear* will be set to true
        - if *leapYear* AND *day* are greater than 29 then *validDate* is set to false and the *message* variable receives a new value.
        - else if *leapYear* IS NOT true AND *day* is greater than 28 then *validDate* is set to false and *message* is given a new value.
    - for cases 4, 6, 9 and 11:
        - if *day* is greater than 30, since these months only have 30 days, then *validDate* is set to false and *message* receives a new value
    - for cases 1, 3, 5, 7, 8, 10 and 12:
        - if *day* is greater than 31 (since these months have 31 days) then *validDate* is set to false and *message* receives a new value
    - if the user entered an invalid month then validDate is set to false and a different message displays.

17. Let's complete the program by typing:

```
   JOptionPane.showMessageDialog(null,"Your date was " + date);
      if(validDate)
      {
         JOptionPane.showMessageDialog(null,"It is a valid date.");
      }
      else
      {
         JOptionPane.showMessageDialog(null,"It is not a valid date." + "\nThe reason it
is invalid: " + message);
      }

   }
```

```
}
```

18. This code should be fairly easy for you to follow.
19. Compile the program and fix any errors you may have.
20. Compress both the .java and .class file into a single zip or rar file and submit to the appropriate drop box.