COSC 1436
Lab 03: Keyboard inputs and `if-else` Statements

You can use the `Scanner` class to input values from the keyboard. This requires two setup steps. At the beginning of your program use the following statement:

```
import java.util.Scanner;
```

in your main program declare a Scanner object:

```
Scanner keyboard = new Scanner(System.in);
```

At this point you can simply prompt the user for an input:
Example:

```
System.out.print("Enter a value: ");
```

// And the enter a response

```
name = keyboard.nextLine();
```

In Java, there are often several different ways to do the same thing. The following table shows 2 different ways to get input from the keyboard and put them into a variable. Method 1 is easier to remember, but does not work in some cases, such as in some text windows. Another case is when you try to read an `int` or `double`, followed by reading a `char`. Here, you might get an error when reading the `char`, or the `char` could be blank.

Method 2 is a little more complicated, but works in more cases. That's why we teach you Method 2. We are including Method 1 so that you will recognize it in case you read someone else's code.

Note: for Strings, we are only teaching you one method.

Data Type	Method 1	Method 2
Byte	byte num=0; num=keyboard.nextByte();	byte num=0; num= Byte.parseByte(keyboard.nextLine());
short	short num=0; num=keyboard.nextShort();	short num=0; num=Short.parseShort(keyboard.nextLine());
int	int num=0; num=keyboard.nextInt();	int num = 0; num=Integer.parseInt(keyboard.nextLine());
long	long num=0; num=keyboard.nextLong();	long num=0; num = Long.parseLong(keyboard.nextLine());
float	float num=0.0; num=keyboard.nextFloat();	float num=0.0; num=Float.parseFloat(keyboard.nextLine());
double	double num=0.0; num=keyboard.nextDouble();	double num = 0.0; num=Double.parseDouble(keyboard.nextLine());
char	char symbol= 'A'; symbol=keyboard.next().charAt(0);	char symbol= 'A'; symbol =keyboard.nextLine().charAt(0);
String	String name="A"; name = keyboard.nextLine();	

Conditional statements (`if – else`)

In the lectures we have examined the way in which the programmer can test conditions and execute specific blocks of code depending on the **boolean result.**

These are called decision structures. So far, we have seen `if - else`
The `if` structure has a number of options depending on how the programmer needs to test.

Simple `if` condition

```
if (condition is true){
   Execute statement1;
   Execute statement2;
   …
}
```

`if-else`

```
if (condition is true){
  execute statement1;
  execute statement2;
  …
}
else {
  execute statement3;
  execute statement4;
  …
}
```

Cascade of `if-else`

```
if (condition1 is true){
  execute statement1;
  execute statement2;
  …
}
else if (condition2 is true){
   execute statement3;
   execute statement4;
   …
   }
     else {
       execute statement5;
       execute statement6;
        …
     }
```

Logical Operators

A logical operator is one that combines one or more Boolean values into an expression than has a Boolean result. In Chapter 3, three logical operators are identified:

- The NOT operator `!`
- The AND operator `&&`
- The OR operator `||`

These operators are most often associated with structures like the `if` and `if..else`. The logical operators are used to modify or to join expressions so that multiple tests can be made within an if statement.

One example of a logical operator used in `if` statement is:

```
if (x > 20 && x < 40){
   …
}
```

and can be interpreted as "if x is greater than 20 AND x is less than 40". The same expression could be written in a nested if as:

```
if( x > 20) {
      if(x < 40) {
            …
      }
}
```

Expression	Meaning
`(x>y) && (a<b)`	Is x greater than y AND is a less than b?
`(x==y) || (x==z)`	Is x equal to y OR is x equal to z?
`!(x>y)`	Is the expression x>y NOT true?

The way in which conditions are tested to see if they are true or not relies on **Relational Operators and Logical Operators**.

Relational Operators

Construct expressions that result in a Boolean value, either TRUE or FALSE.
The relational operators are:

==	is equal to	!=	not equal to	<	is less than
>	is greater than	>=	greater than or equal to	<=	less than or equal to

Expression	Meaning
x>y	Is x greater than y?
x<y	Is x less than y?
x>=y	Is x greater than or equal to y?
x<=y	Is x less than or equal to y?
x==y	Is x equal to y?
x!y	Is x not equal to y?

Program 1 (50 Points):

Start a new project (Lab 03). and write a program that asks the user to type in the number of a month and then tells the user how many days are in that month. If the user types in something other than a number from 1-12, output "`That's not a month!`"

While you could do this program using only `if` statements, you must use some combination of `if-else` statements to get full credit.

Example 1:

```
Please type in a month: 9
That month has 30 days.
```

Example 2:

```
Please type in a month: 2
That month has either 28 or 29 days.
```

Example 3:

```
Please type in a month: 23
That's not a month!
```

Program 2: Logical Operators (40 Points)

A ***leap year*** is a special year containing one extra day, i.e. 366 days in a year. A year is a leap year, if the year is exactly divisible by 4 but and not divisible by 100. Year is also a leap year if it is exactly divisible by 400.

Create a new program in your Lab03 project to calculate whether a year is a leap year.

Step by step descriptive logic to check leap year:

1. Input year from user. Store it in some variable, say year
2. If year is exactly divisible by 4 and not divisible by 100, then it is leap year.
3. Or if year is exactly divisible by 400 then it is leap year.
4. A year that does not comply with the previous conditions is not a leap year.

Example:

```
Enter year to check: 2023
2023 is not a leap year
```

Test your program using the following years:

1. 2023 is not a leap year (not divisible by 4).
2. 2024 is a leap year (divisible by 4, but not divisible by 100).
3. 2100 is not a leap year (while it is divisible by 4, it is also divisible by 100 but not divisible by 400).
4. 2000 is a leap year (while it is divisible by 100, it is also divisible by 400).

Hint: Look in your notes about using the modulus operation `%` and divisibility.

Multiple Choice Questions (10 Points)

1) What will be the value of x after the following statements are executed?

```
int x = 80;
int y = 65;
if (y > x){
    x = x - y;
}
```

a. 15

b. 80

c. 65

d. -15

2) What will be the value of x after the following statements are executed?

```
int x = -10;
switch (x) {
  case -10:
    x = x + 10;
    break;
  case 12:
    x = x * -1;
    break;
}
```

a. 10

b. -10

c. 0

d. 12

3) What will be displayed after the following statements are executed?

```
int y = 20;
if (y == 20){
      int x = 5;
      x += y;
      System.out.println(x);
}
```

a. 20

b. 25

c. 5

d. 0

4) What is the output of the following code after execution?

```
int a=2;
int b=5;
int result;

result=a+b;
System.out.print(result);
```

a. 7

b. 5

c. 1

d. Error

5) What is the output of the following code after execution?

```
int a=1;
int b=2;
int result;

result=a/b;
System.out.println(result);
```

a. 0.5

b. Error

c. 0

d. 1/2

As in previous labs, put your multiple choice answers in comments at the end of your 2nd program, and submit only the .java files for your 2 programs.

Make sure your programs follow the commenting rules handed out with Lab 02.