To input date use `nextInt()`,

Suppose: `<Show code>`

```java
int date = scan.nextInt();
String day = scan.nextLine();
```

User's input:

```
14 "In" day
Wednesday
```

Ideas to fix?

- Use another `nextLine()` after `nextInt()`

```java
int year = scan.nextInt();
scan.nextLine(); // reading + ignoring "\n"
String day = scan.nextLine();
```
Different data types lead to need to convert data from one type to another.

```
cast - convert the data type of a variable

double x = 5.0;
int y;

y = x;  // won't compile

y = (int) x;  // casting
```

What happens here?

```
x = 5.95;
y = (int) x;  // rounds down
```
Conditionals (if/else):

1. Input an integer from user and set a boolean valid to true.

   comparative operators: < > <= >= == !=
   logical: " " " OR " " " AND " " " NOT " "

   if (condition) { //if block
       <body>
   }

   else { //else block
       <body>
   }

   boolean valid;
   Scanner scan = new Scanner(System.in); //needs import
   int num = scan.nextInt();

   if (num > 0) { //Needed if more than 1 statement
       valid = true;
       System.out.println("valid");
   } //if block
   else { //else block
       valid = false
   }

   Output valid if num<20 or >50

   2 ways
In-class code

if (num < 20) { // if block
    valid = true;
    System.out.println("valid");
} else if (num > 50) { // else-if block
    valid = true;
    System.out.println("valid");
} else
    valid = false;

2nd way:

In-class code

if (num < 20 || num > 50)
    valid = true
else
    valid = false.

Blocks of code limit accessibility of variables

Scope - area within a program where a variable can be accessed
Rule - a variable can be accessed only within innermost section of its declaration (boolean valid).
Ex. if (num < 20) { necessary

    boolean valid2 = true;

    System.println(valid2);

Will it compile? No.

    valid2 declared here

    println() not within valid2's block

    <Sample Code> (scope.java)

Loops:

    Looping done more easily with increments.

    i += 1; i = i + 1
    i -= 1; i = i - 1
    i += 2; i = i + 2
    i -= 5; i = i - 5
    i *= k; i = i * k
"3 Types of Loops"

1. do-while: use when number of iterations is not known in advance

```
skip
```  

```
do {
    <loop body>
} while (condition);
```  

```
do {
    <loop body>
} while (condition);
```  

"Difference?" Do-while loop executes at least once

```
ex:

int count = 10; // not 0!
do {
    System.out.println(count);
    count++;
} while (count < 5);
```

"Output?" => 10 done

```
do-while: executes at least once
while: may not execute
```
3) for loop: use when number of iterations is known in advance

ex: input 10 integers from user

Scanner scan = new Scanner(System.in);
int num;

Break this up:

for (int i = 0; i < 10; i++)

1. Step 1
   - initialization (done once)

2. Steps 2, 5, 8
   - condition (repeated)

3. Steps 4, 7, 10
   - increment (repeated)

System.out.println("Enter number ");
num = scan.nextInt();

Question: how to modify above to print sum of 5 integers?

Scanner scan;

int sum = 0; // must initialize to 0!

for (int i = 0; i < 5; i++)

System.out.println("Enter number ");
num = scan.nextInt();
sum += num;