Java:

Java has data types

(1) primitive data type - stores basic data
   - built into Java language

int: 10, -5, 0 ([-2 billion, +2 billion])

double: 5.63, -7.1, 10.0

char: 'a', 'A', '7', '8'

boolean: true, false

Each data type has: range of values
   : set of operations

ex: int operations: + - * / %= =, <, >, <=, >=, ==

int x;
   declaration: set type + name

x = 5;
   assignment / definition: set value

int y = 10
   initialization: declare + assign

int sum = x + y;
   initialization + operation

<br>

Sample Code>
2. Object data type - stores advanced data
   "Object-oriented programming"
   (a) built into Java (like primitive data)
   (b) built by programmer <later>

   a) ex. String "stores" sequence of characters

   methods (operations) +, =, toUpperCase(), toLowerCase(), equals()...

   String a = "Hello"
   String b = "hello"

   String c = a.toLowerCase();

   System.out.println(a.equals(b)); -> false
   System.out.println(c.equals(b)); -> true

   <Sample code>

   For primitive data, operations are basic
   For objects, methods are more advanced
   Might be wondering: which methods can be performed?

   class - defines (data values) + methods of an object

   => Java API for String class

   Notice: methods have input/output
`ex: String replace(char oldChar, char newChar)`

- Return type: `<may be void>`
- Method name: `<may be empty>`
- Parameters/arguments/input:

To Do: Convert bitter to BIGGER

```java
public class convertString {  // save as convertString.java

    public static void main(String[] args) {

        String s1 = "bitter";
        String s2 = s1.replace(‘t’, ‘g’);
        String s3 = s2.toUpperCase();

        // print and skip to new line
        System.out.println("new String is: "+s3);

        // print and stay on same line
        System.out.print(s3);
        System.out.println("done");

    }

    }
```

Output:

new String is BIGGER
BIGGER done

-sample code above-
Strings try to mimic primitive data types.

Most other classes require an import to be used.

**Scanner object** - reads user input

```java
import java.util.Scanner;

public class InputExample {

    public static void main(String[] args) {
        int x = 5;
        int y = 7;
        System.out.println(x + y);

        Scanner scan = new Scanner(System.in);

        System.out.println("Enter a day:");
        String day = scan.nextLine();
        System.out.println("Happy " + day);
    }
}
```
To input date:

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

Suppose: <Show code>.

User's input:

```
14
```

Ideas to fix?

=> Use another `nextLine()` after `nextInt()`.

```
int year = scan.nextInt();
scan.nextLine(); // reading + ignoring "In"
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

```java
double x = 5.0;
int y;

y = x;  // won't compile

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

*What happens here?*

```java
x = 5.75;

y = (int) x;  // rounds down
```