Video Name: SimpleInput

Topics:

- Keyboard input
- Wrapper classes
- try/catch NumberFormatException

Java Class(es): SimpleInput

**Scanner class**

use Scanner class to parse data

declare:

```java
Scanner scan = new Scanner(System.in);
```

use:

```java
theName = scan.nextLine();
```

Scanner functions (partial list):

- `nextLine()` – gets entire line, including spaces
- `next()` – gets next string, ends at white space
- `nextInt()` – gets integer, `InputMismatchException` if not an int
**Wrapper class**

- each primitive has an associated wrapper class
- name is similar, but with capital letter, as shown in table
- wrapper can contain a single value of the associated type (useful with collections)
- provide static conversion functions, e.g., `Integer.parseInt`

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Wrapper</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>byte</td>
<td>Byte</td>
</tr>
<tr>
<td>char</td>
<td>Char</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
<tr>
<td>long</td>
<td>Long</td>
</tr>
<tr>
<td>short</td>
<td>Short</td>
</tr>
</tbody>
</table>

**Exceptions**

- To avoid generating exception on illegal input, can use a `try/catch` block

```java
String scoreStr = scan.next();
try {
    score = Integer.parseInt(scoreStr);
} catch (NumberFormatException e) {
    System.out.println("That is not a number!");
}
```