WRAPPER CLASSES

Each primitive data type in Java has a corresponding wrapper class in package `java.lang`. A wrapper class is used to wrap a primitive datum within an object.

<table>
<thead>
<tr>
<th>Java Primitive Data Type</th>
<th>Corresponding Wrapper Class in java.lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Byte</td>
</tr>
<tr>
<td>short</td>
<td>Short</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>long</td>
<td>Long</td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>float</td>
<td>Float</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
</tbody>
</table>

Objects of a wrapper class can be built using a constructor that expects an argument of the corresponding primitive data type.

**Examples**

```java
Double d = new Double( 2.5 );
```

```mermaid
diagram Box
    d [Double object 2.5]
```

```java
Integer i = new Integer( 100 );
```

```mermaid
diagram Box
    i [Integer object 100]
```

```java
Character c = new Character( ' $ ' );
```

```mermaid
diagram Box
    c [Character object $]
```

```java
Boolean b = new Boolean( true );
```

```mermaid
diagram Box
    b [Boolean object true]
```
Furthermore, each wrapper class except `Character` has an overloaded constructor that accepts a string argument.

### Examples

<table>
<thead>
<tr>
<th><code>Double</code> <code>d</code> = new <code>Double</code>(<code>&quot;2.5&quot;</code>);</th>
</tr>
</thead>
</table>
| ![Diagram](image1)
| `d` → `Double object`
| `2.5`

<table>
<thead>
<tr>
<th><code>Integer</code> <code>i</code> = new <code>Integer</code>(<code>&quot;100&quot;</code>);</th>
</tr>
</thead>
</table>
| ![Diagram](image2)
| `i` → `Integer object`
| `100`

<table>
<thead>
<tr>
<th><code>Boolean</code> <code>b</code> = new <code>Boolean</code>(<code>&quot;true&quot;</code>);</th>
</tr>
</thead>
</table>
| ![Diagram](image3)
| `b` → `Boolean object`
| `true`

If the string argument to one of these constructors is not a valid representation of the primitive datum, then the constructor will throw an exception.

### Example

This Java statement:

```java
Integer i = new Integer("2.5");
```

Throws the run-time error:

```
Exception in thread "main" java.lang.NumberFormatException
```

If you use a primitive datum where an object is required, the Java compiler automatically replaces it by an object of the corresponding wrapper class. This is called **autoboxing** the primitive.

### Examples

<table>
<thead>
<tr>
<th><code>Double</code> <code>d</code> = 2.5;</th>
</tr>
</thead>
</table>
| ![Diagram](image4)
| `d` → `Double object`
| `2.5`
If you use a wrapper class object where a primitive datum is expected, the Java compiler automatically unboxes the object by replacing it with the primitive datum.

**Example**
In line 1 below, the compiler automatically boxes the primitive 100 into the Integer object i. In line 2 it unboxes i in order to apply the / operator. The code outputs the value 50.

1. Integer i = 100;
2. System.out.println( i/2 );

**Exercises**

1. Write the Java statement that converts 1E6 to a Double object.
2. Write the Java statement that converts 1_000_000 to an Integer object.
3. Write the Java statement that converts true to a Boolean object.
4. Write the Java statement that converts 'A' to a Character object.