Type Casting

Type Casting

Type casting (or type conversion) is when you change the data type of a variable.

```
int numerator = 40;
int denominator = 25;
System.out.println( numerator / denominator);
System.out.println( (double) numerator / denominator);
```

TRY IT

```
1
1.6
```

`numerator` and `denominator` are integers, but `(double)` converts `numerator` into a double.

What happens if you:

- Cast only `denominator` to a double?
- Cast both `numertor` and `denominator` to a double?
- Cast the result to a double (i.e. `(double)(numerator / denominator)`)?

TRY IT

```
1
1.6
```

▼ **More Info**

If either or both numbers in Java division are a `double`, then `double` division will occur. In the last example, numerator and denominator are both `int` when the division takes place - then the integer division result is converted to a double.

Parsing Strings

What do you think the code below will print?

```
int a = 5;
String b = "3";
System.out.println(a + b);
```

TRY IT

```
53
```

When you try to print an integer and a string added together, Java will automatically convert the integer into a string. This occurs because the system attempts to perform string concatenation. This is why the code above resulted in `53`. To perform integer addition, you can convert `b` to an integer.

```
int a = 5;
String b = "3";
System.out.println(a + Integer.parseInt(b));
```

TRY IT

```
8
```

Data read from the keyboard or a file is always stored as a string. If you want to use this data, you will need to know how to convert it to the proper data type.

What happens if you:

- Parse a String to a double using `Double.parseDouble()`
- Parse a String to a boolean using `Boolean.parseBoolean()`
- Convert a different type to a string with `String.valueOf()`

Casting and Parsing

Which of the following throws an error?

Assume the following:

```
int number = 5;
double decimal = 6.2;
boolean TF = true;
String words = "text";
```

```
System.out.println(words + String.valueOf(number+decimal));
```

```
System.out.println(String.valueOf(TF) + words);
```

```
words = "3.7";
System.out.println(Integer.parseInt(words) + number);
```

```
words = "3.7";
System.out.println(Double.parseDouble(words) + decimal);
```

```
System.out.println(Boolean.parseBoolean(words));
```

```
words = "3.7";
System.out.println(Integer.parseInt(words) + number);
```

is invalid because `3.7` cannot be parsed into an int. The string must have an integer value. You can see in the other parse examples (`parseBoolean` and `parseDouble`) that words has to be re-assigned to a compatible String.