Sometimes you have a question about how Java works. You might search the internet or a textbook for your answer, but another way to answer your question is to experiment. For example, *what does typecasting do exactly?*

Typecasting is used when you know an assignment is safe and the compiler doesn't. The example below tells the compiler to treat **a** as if it were of byte type, which is safe because we know 0 is a value that byte can represent.

```
int a = 0;
byte b = (byte) a;
```

1) What happens if the typecast is removed from the assignment? Code it to find out.

2) What happens if you typecast as above and **a** has a value that **b**'s type can't represent? Discuss and implement code to experimentally find out. Hint: You can print a byte and an int as hex using System.out.printf("%02X\n", b) and System.out.printf("%08X\n", a).

3) Did you learn enough from this experiment to explain what happens when you assign a larger type integer variable to a smaller type integer variable? If so, explain what happens and verify it with more examples. If not, propose another experiment.

4) If you have byte, int, and double variables, which can be assigned to which without typecasting? Figure it out and/or verify your answer experimentally. Can you formulate a simple rule that covers your answer?

5) What happens when you typecast and assign a double variable to an int? Figure it out and/or verify your answer experimentally.

Note on typecasting: Avoid typecasting when possible. Code that uses it is harder to understand, and typecasting is often a symptom of not choosing the right type variable to begin with. On the other hand, there are times when typecasting is the best solution. For example, if you want the last byte of an int.

```
// You'll learn about & in CSC 35 and/or 60
byte b = (byte) (a & 0xFF); // Only way to get it
```