Model 1 Adding New Methods

Add the following method to the `MyBigInt` class:

```
public MyBigInt reverse() {
    String str = super.toString();
    final int N = str.length();

    // reverse the digits in the string
    StringBuilder sb = new StringBuilder(N);
    for (int i = 0; i < N; i++) {
        int j = N - 1 - i;
        sb.append(str.charAt(j));
    }
    return new MyBigInt(sb.toString());
}
```

Add the following code to the `main` method:

```
BigInteger bi1 = new BigInteger("12345678");
MyBigInt bi2 = new MyBigInt("12,345,678");
System.out.println(bi1.reverse());
System.out.println(bi2.reverse());
```

Questions (20 min) Start time:

1. Attempt to compile and run the program. Explain the error in `main`.

2. Remove the line that caused the error, and run the program. What is the result?

3. Which `toString` method (in which class) is invoked on the first line of `reverse`?

4. Explain why `reverse()` does not need to worry about the placement of commas.

5. Consider a method `isPalindrome()` that determines whether a `MyBigInt` has the same digits forward and backward. For example, `123,321` and `12,321` are palindromes, but `123,421` and `12,341` are not. How could you implement this method using one line of code?

```
public boolean isPalindrome() {



}
```

6. Why is the one-line implementation inefficient, especially for very large integers?

7. Rewrite `isPalindrome()` to be more efficient. (*Hint:* Use the source code of `reverse()` as a starting point.)

```
public boolean isPalindrome() {



}
```

8. Add your solution to *MyBigInt.java*, and make sure it works. What code can you add to `main` to test the `isPalindrome` method?