Feedback on grade assignment

Your Grade program passes 25 out of 25 tests. That's 100%.

General feedback

One way to write the `grade` function to pass the first block of tests is:

```
String grade(int mark) {
    if (mark < 0) return "Invalid";
    if (mark < 50) return "Fail";
    if (mark < 60) return "Pass";
    if (mark < 70) return "Merit";
    if (mark <= 100) return "Distinction";
    return "Invalid";
}
```

Note that `else` isn't needed, because `return` exits the function straight away. Also note that you don't need to write things like `if (mark >= 0 && mark < 50)` because it is normal to make use of the fact that previous cases have already been dealt with.

Purists might say that it is better not to use 'early returns', because then the function has multiple exit points, and becomes difficult to analyze or extend (should we want to do either of those things). So, it could be written instead as:

```
String grade(int mark) {
    String result;
    if (mark < 0) result = "Invalid";
    else if (mark < 50) result = "Fail";
    else if (mark < 60) result = "Pass";
    else if (mark < 70) result = "Merit";
    else if (mark <= 100) result = "Distinction";
    else result = "Invalid";
    return result;
}
```

To implement the convert function, as well as calling`Integer.parseInt` to convert the string into a number, invalid strings need to be dealt with. A direct approach involves making sure that each character is a digit. If you don't find out about the `Character.isDigit` function, you can write your own:

```
boolean isDigit(char c) { return '0' <= c && c <= '9'; }
```

If you use the hint from the assignment description to deal with each length separately, you might end up with this:

```
int convert(String mark) {
    if (mark.length() == 1) {
        if (! isDigit(mark.charAt(0))) return -1;
    }
    else if (mark.length() == 2) {
        if (! isDigit(mark.charAt(0))) return -1;
        if (! isDigit(mark.charAt(1))) return -1;
        if (mark.charAt(0) == '0') return -1;
    }
    else if (mark.length() == 3) {
        if (! mark.equals("100")) return -1;
    }
    else return -1;
```

```
    return Integer.parseInt(mark);
}
```

Note that the checks ensure that the number is between 0 and 100 as well as that it is properly formed. Also note that you have to be careful not to call `charAt(i)` when `i`is `>=` the length. These lines of code have to be thought about carefully and logically (or at least tested thoroughly).

If you are happy to use a loop, you can write something that is shorter, simpler and more general:

```
int convert(String mark) {
    for (int i = 0; i < mark.length(); i++) {
        if (! isDigit(mark.charAt(i))) return -1;
    }
    if (mark.length() > 1 && mark.charAt(0) == '0') return -1;
    int n = Integer.parseInt(mark);
    if (n > 100) n = -1;
    return n;
}
```

If you are happy to use pattern matching with regular expressions, you can replace the loop with:

```
...
    if (mark.matches(".*[^0-9].*")) return -1;
...
```

Note that this kind of pattern matches gets really horrible beyond a certain level of complexity.

If you are prepared to find out about exceptions, you can take a different approach. That is to call `Integer.parseInt` straight away, and catch any exception that it throws as a result of an invalid string.

```
int convert(String mark) {
    if (mark.length() > 1 && mark.charAt(0) == '0') return -1;
    int n;
    try { n = Integer.parseInt(mark); }
    catch (Exception err) { n = -1; }
    if (n > 100) n = -1;
    return n;
}
```

Note that the variable `n` has to be declared outside the`try` block, so it can be used later.

Style note

I meant to tidy up the skeleton that I gave you, and forgot. The `test` method is a bit long, and I don't like comments within methods - they suggest that the method should be split up or made more readable. The tests look better like this:

```
// Check a mark inside each grade band.
void testGrades() {
    claim(grade(45).equals("Fail"));
    claim(grade(55).equals("Pass"));
    claim(grade(65).equals("Merit"));
    claim(grade(75).equals("Distinction"));
}

// Check the boundaries of the grade bands.
void testBoundaries() {
    claim(grade(0).equals("Fail"));
    claim(grade(49).equals("Fail"));
    claim(grade(50).equals("Pass"));
    claim(grade(59).equals("Pass"));
```

```
    claim(grade(60).equals("Merit"));
    claim(grade(69).equals("Merit"));
    claim(grade(70).equals("Distinction"));
    claim(grade(100).equals("Distinction"));
}

// Check marks which are out of range.
void testRange() {
    claim(grade(-1).equals("Invalid"));
    claim(grade(101).equals("Invalid"));
}

// Check convert works on numbers in range
void testConvert() {
    claim(convert("0") == 0);
    claim(convert("53") == 53);
    claim(convert("100") == 100);
}

// Check convert detects invalid strings, i.e. ones containing non-digits,
// or number out of range, or leading zero. (Note: leading zeros are
// potentially ambiguous because they indicate octal in some circumstances)
void testInvalid() {
    claim(convert("x") == -1);
    claim(convert("5x") == -1);
    claim(convert("5x6") == -1);
    claim(convert("40.5") == -1);
    claim(convert("-1") == -1);
    claim(convert("101") == -1);
    claim(convert("01") == -1);
    claim(convert("099") == -1);
}

// Run the tests.
void test() {
    testGrades();
    testBoundaries();
    testRange();
    testConvert();
    testInvalid();
    System.out.println("All tests passed");
}
```