CSCI 3357: Database System Implementation
Homework Assignment 6
Due Thursday, October 26

This assignment asks you to revise the SimpleDB record manager to enable record values to be null. In particular, you must add the following two public methods to `RecordPage`:

- the method `setNull(slot, fldname)`, which sets the value of the specified field of the specified slot to null.
- the method `isNull(slot, fldname)`, which determines if the value of the specified field of the specified slot is null.

The `isNull` method is the only way a client can know if a record value is null. Suppose that you access the value of an integer field, say, by doing `rp.getInt(slot, "GradYear")`. You can't tell if the returned value is null simply by looking at it, because all integers are legal values — there is no integer that corresponds to null. Similarly, there is no special string value that can be treated as a null — for example, the string "null" and the empty string "" are both legitimate non-null strings. That is why you need a special method to tell you whether the value is actually null, regardless of what `getInt` or `getString` returns.

How then to implement a null value? Since it is unreasonable to use a particular integer or string value to denote a null, you should use a one-bit flag. In particular, say that a record contains N fields. Store N additional bits with each record and assign the value of the i^{th} bit to be 1 if the value of the i^{th} field is null and 0 if it is non-null. For this assignment we will assume that N<32, which means we can use the `EMPTY/INUSE` integer for this purpose. Bit 0 of this integer (counting from the right) denotes empty/inuse, as before. But now the other 31 bits are available to hold null-value information.

You will need to modify the `Layout` constructors so that they assign a bit position to each field in the record. (So for example if the schema has fields "A" and "B", then "A" might be assigned position 1 and "B" position 2.) Be careful to perform this assignment the same way in both constructors. Your `Layout` class should also implement the following new method, which will allow the record page to determine the bit position of any field:

```
public int bitPosition(String fldname);
```

The `setNull` and `isNull` methods will need to get and set individual bits of the empty/inuse integer. Since not all of you have learned how to do this, I have written the following two methods for you:

- The method *getBitVal* returns the value of a specified bit (from the right, and beginning at 0) of a given integer.

```
private int getBitVal(int val, int bitpos) {
     return (val >> bitpos) % 2;
}
```

- The method *setBitVal* returns the integer that results from setting a specified bit (from the right and beginning at 0) of a specified integer to a specified flag (which will be either 0 or 1).

```
private int setBitVal(int val, int bitpos, int flag) {
     int mask = (1 << bitpos);
     if (flag == 0)
          return val & ~mask;
     else
          return val | mask;
}
```

For example, the integer 6 has the bit representation …000110. Therefore the calls getBitVal(6, 1) and getBitVal(6,2) return 1, and getBitVal(6,0), getBitVal(6,3), and getBitVal(6,n) for all n≥4 return 0. Similarly, the call setBitVal(6, 1, 0) returns 4 (that is, …00100) and setBitVal(6, 0, 1) returns 7 (that is, …000111).

For your sanity, you should define constants NULL and NOTNULL to be 1 and 0 respectively.

You will also need to make changes to other methods in `RecordPage`. For example, the `searchAfter` method can no longer simply check to see if the empty/inuse value is 0 or 1. Instead, it must only check the 0^{th} bit of that integer for 0 or 1. In addition, the `setInt` and `setString` methods should set their field to non-null. On the other hand, methods that insert, delete, and format should continue to set the entire EMPTY/INUSE value to 0 or 1, because that will set all the null bits to 0 (which is NOTNULL).

This problem is tricky. Please think it through carefully. It is important to remember that the null bits for a record are stored **on disk** with that record. In particular, a record's null bits are part of the 32-bit empty/inuse integer that is stored at the beginning of each record on the page. Consequently, setting a record's value to null requires four steps:

a) Determine the location of the record in its page.
b) Grab the empty/inuse integer for that record.
c) Change the value of the appropriate bit of that integer to 1.
d) Store the revised integer back in the page.

To keep me from being confused, I wrote some additional private methods to handle the reading and writing of individual bits inside a specified slot, such as:

```
int  getFlagValue(RecordPage rp, int slot, int bitpos)
void setFlagValue(RecordPage rp, int slot, int bitpos, int val)
```

I found them much easier to use than `getBitVal` and `setBitVal` directly.

You can use my class `HW6Test` as a way to verify that your code works. I suggest that you simplify or rewrite it during debugging, so that you can test features incrementally.

When you are done, create a zip file containing *Layout.java and RecordPage.java*, and submit it Canvas.