PinPage Notes.

As I mentioned in class, there is an known commonly-encountered issue with PinPage and how parameters are handled in java. See notes below.

Thanks to Sergio Garcia for these notes.

Suppose our BufMgr implementation looks something like this:

```
public class BufMgr implements GlobalConst {
  [elided fields]
  // Suppose you're using a HashMap that maps PageIds to FrameDescs
  protected HashMap<PageId, FrameDesc> bufmap;

  public void pinPage(PageId pageno, Page mempage, int contents) {
    [elided]
    // Suppose you've selected a victim page for replacement in the
    // frame table. You've done the work needed to set the frame up,
    // and you insert it into bufmap with the pageno PageId reference
    // we received as an argument as the key.
    bufmap.put(pageno, victim_page);
    [elided]
  }
```

Now consider the follow snippet from test #1 in BMTest:

```
// Read that something back from each page.  Pin, read, unpin.
// Because there are more pages than frames, disk I/O will happen here.
System.out.print("  - Read that something back from each one\n");

for (pid.pid = firstPid.pid; status1 == PASS && pid.pid < lastPid.pid;
                pid.pid = pid.pid + 1) {

  try {
    Minibase.BufferManager.pinPage(pid, pg, PIN_DISKIO);
  } catch (Exception e) {
    System.err.print("*** Could not pin page " + pid.pid + "\n");
    e.printStackTrace();
    return false;  //If we can't pin the page, end this test
  }
  ...
}
```

Specifically, take a look at the for-loop's increment expression and the use of the pid reference for pinning a page. What are the ramifications of using the pid reference above as our key for insertion into bufmap when the codebase reuses a reference as this snippet does? [Hint: consider the nature of references in Java.] What would be a viable solution to

the problem?