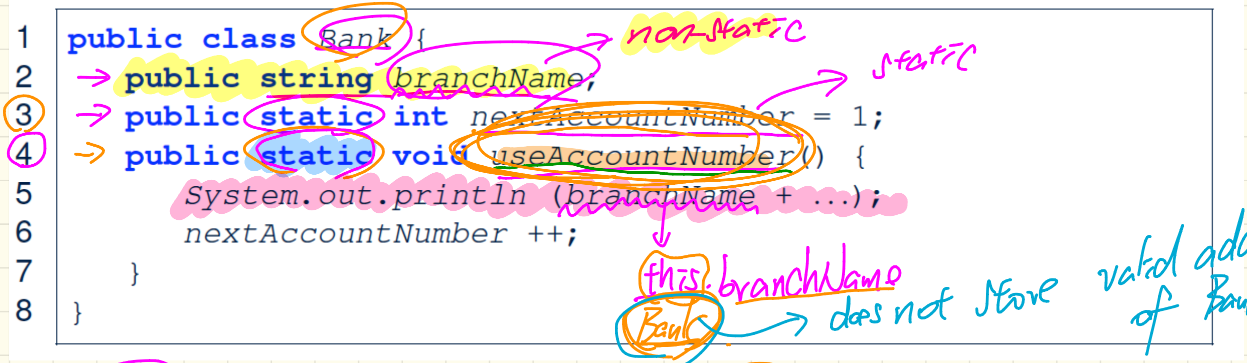
Monday March 25

Lecture 21

Use of Static Variables: Common Errors



```
public class Bank {
  public string branchName;
  public static int nextAccountNumber = 1;
  public static void useAccountNumber() {
    System.out.println (branchName + ...);
    nextAccountNumber ++;
  }
}
```

non-static

static

this.branchName

Bank → does not store valid address of Bank object ∴ not compile.

Bank.nextAccountNumber

class name, not C.O.

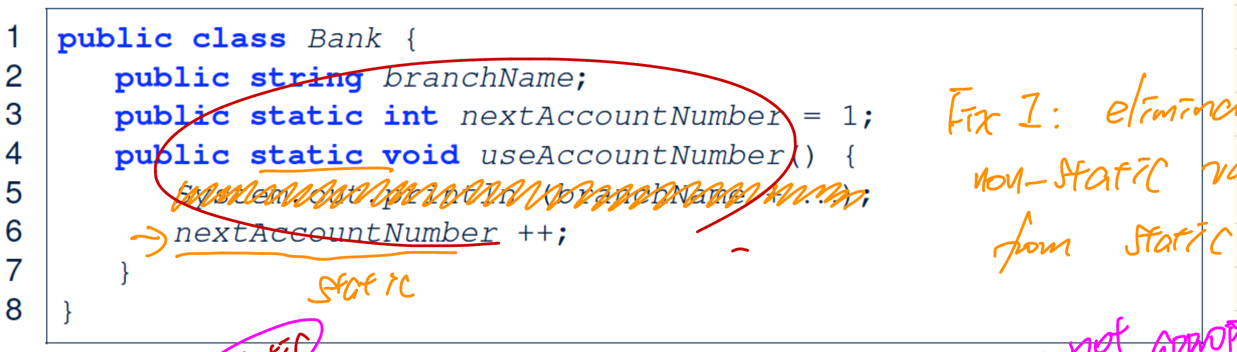
Bank.useAccountNumber()

Bank b = new Bank();

b.branchName

context object

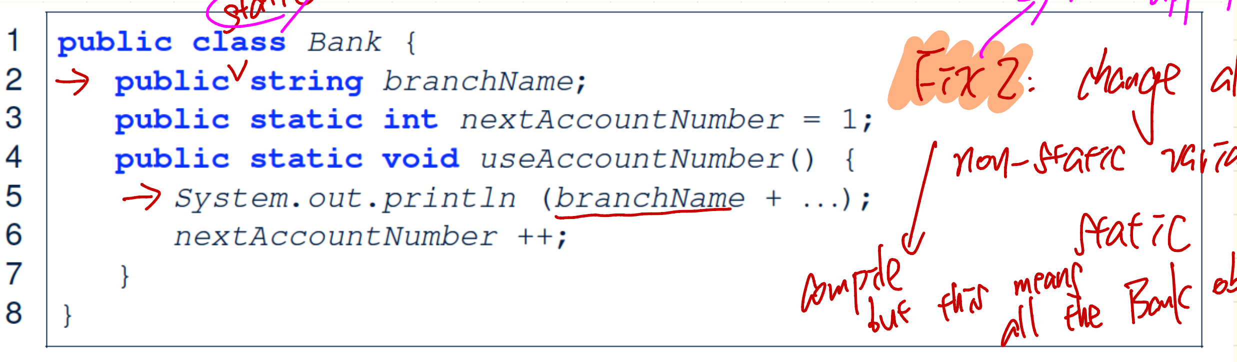
Use of Static Variables: Common Errors



```
public class Bank {
    public string branchName;
    public static int nextAccountNumber = 1;
    public static void useAccountNumber() {
        ~~System.out.println (branchName + ...);~~
        nextAccountNumber ++;
    }
}
```

static

Fix 1: eliminate all non-static variables from static methods.



```
public class Bank {
    public static string branchName;
    public static int nextAccountNumber = 1;
    public static void useAccountNumber() {
        System.out.println (branchName + ...);
        nextAccountNumber ++;
    }
}
```

Fix 2: change all non-static variables to static

not appropriate

compile but this means all the Bank objects have/shc. the same branchName

Programming Pattern: Mutator

```
class PointCollector {
  Point[] points; int nop; /* number of points */
  PointCollector() { points = new Point[100]; }
  void addPoint(double x, double y) {
    points[nop] = new Point(x, y); nop++; }
```

```
class PointCollectorTester {
  public static void main(String[] args) {
    PointCollector pc = new PointCollector();
    System.out.println(pc.nop); /* 0 */
    pc.addPoint(3, 4);
    System.out.println(pc.nop); /* 1 */
    pc.addPoint(-3, 4);
    System.out.println(pc.nop); /* 2 */
    pc.addPoint(-3, -4);
    System.out.println(pc.nop); /* 3 */
    pc.addPoint(3, -4);
    System.out.println(pc.nop); /* 4 */
```

Programming Pattern: Accessor

```
Point[] getPointsInQuadrantI() {
  Point[] ps = new Point[nop];
  int count = 0; /* number of points in Quadrant I */
  for(int i = 0; i < nop; i ++) {
    Point p = points[i];
    if(p.x > 0 && p.y > 0) { ps[count] = p; count ++; } }
  Point[] q1Points = new Point[count];
  /* ps contains null if count < nop */
  for(int i = 0; i < count; i ++) { q1Points[i] = ps[i] }
  return q1Points;
} }
```

4

return points; ✗

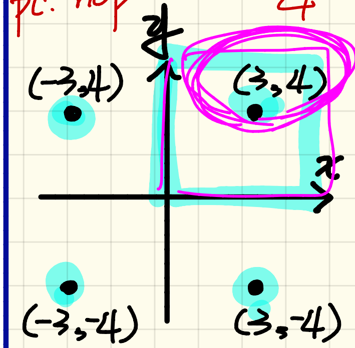
return ps; ✗

ps[0] = p; count ++;

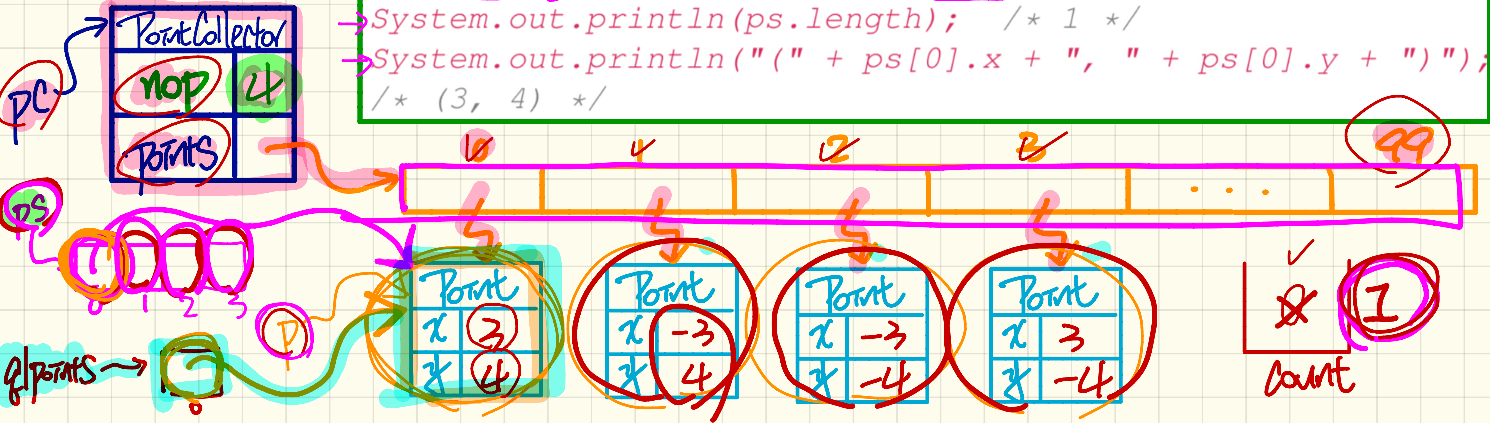
1

pc.points.length 100

pc.nop 4



```
Point[] ps = pc.getPointsInQuadrantI();
System.out.println(ps.length); /* 1 */
System.out.println("(" + ps[0].x + ", " + ps[0].y + ")");
/* (3, 4) */
```

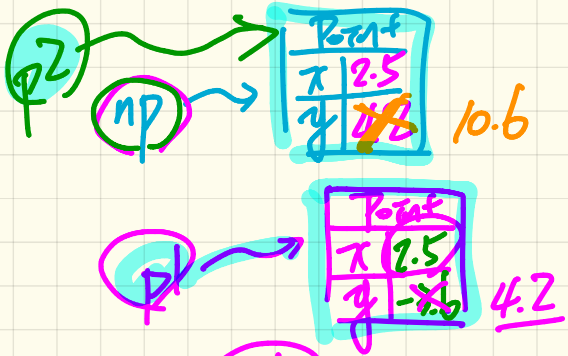


Return Type: Reference Type

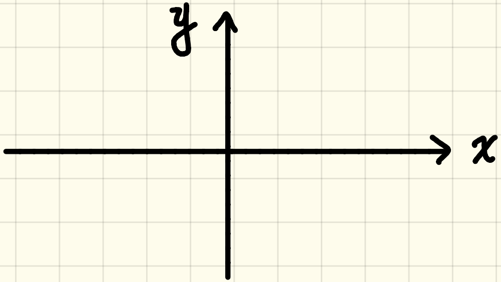
```
class Point {
  Point(double x, double y) {...}

  void moveUpBy(double units) {
    this.y = this.y + units;
  }

  Point movedUpBy(double units) {
    Point np = new Point(this.x, this.y);
    np.moveUpBy(units);
    return np;
  }
}
```

7.8 p1 p1

6.4 p1 p1 6.4



np == p1 False.



```
class PointTester {
  static void main(String[] args) {
    Point p1 = new Point(2.5, -3.6);
    p1.moveUp(7.8);
    Point p2 = p1.movedUpBy(6.4);
    System.out.println(p1 == p2);
  }
}
```

False.