PV260 Software Quality

NAME, SURNAME and UČO: _____

Q1. How would you refactor the following code fragments? [PLEASE answer in ENGLISH, you can also answer in natural language and/or with code/pseudocode]

```
double basePrice = _quantity * _itemPrice;
if (basePrice > 1000)
    return basePrice * 0.95;
else
    return basePrice * 0.98;
This example derives from Fowler's (et al.) book [1] - the purpose was to show the Replace Temp with Query refactoring.
A revisited solution taking into account some coding standards violations could be the following (the other "magic
```

```
if (basePrice() > MAX_THRESHOLD){
return basePrice() * 0.95;
} else{
return basePrice() * 0.98;
```

numbers" could be replaced as well):

double basePrice(){ return quantity * itemPrice;

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, Refactoring: Improving the Design of Existing Code, 1st edition. Reading, MA: Addison-Wesley Professional, 1999.

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&
  (browser.toUpperCase().indexOf("IE") > -1) && wasInitialized() &&
  resize > 0 ) {
    //Do something
  }
```

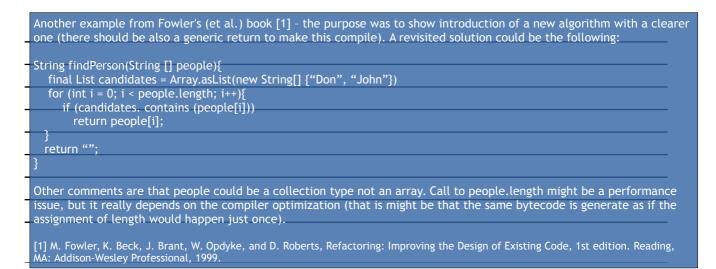
This example derives also from Fowler's (et al.) book [1] - the purpose was to show the Introducing Explaining Variable refactoring. A revisited solution taking into account some coding standards violations could be the following:

```
final boolean IS_MACOS = platform.toUpperCase(). indexOf("MAC") > -1;
final boolean IS_IEBROWSER = browser.toUpperCase(). indexOf("IE") > -1;
final boolean WAS_RESIZED = resize > 0;
```

```
if (IS_MACOS && IS_IEBROWSER && wasInitialized() && WAS_RESIZED){
//Do something
```

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, Refactoring: Improving the Design of Existing Code, 1st edition. Reading, MA: Addison-Wesley Professional, 1999.

```
String findPerson(String[] people) {
    for (int i = 0; i < people.length; i++) {
        if (people[i].equals ("Don") ) {
            return "John";
        c. }
        if (people[i].equals ("John") ) {
            return "Jack";
        }
    }
}</pre>
```



Q2. Show that the following (Java-based) code breaks the Liskov substitutability principle [PLEASE answer in ENGLISH]

```
public class Point {
    private final int x;
    private final int y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    @Override public boolean equals(Object o) {
        if (o == null || o.getClass() != getClass())
            return false;
        Point p = (Point) o;
        return p.x == x && p.y == y;
    }
    ... // Remainder omitted
}
```

This is an example from Joshua Bloch's' book [2] - there is a nice section about equals and hashcode implementations for classes. One point is that hashcode should be always overridden when equals is, but apart from this, as in [2] if we subclass from Point to add a global counter functionality we might get the following: public class CounterPoint extends Point { private static final AtomicInteger counter = new AtomicInteger(); public CounterPoint(int x, int y) { super(x, y); counter.incrementAndGet(); public int numberCreated() { return counter.get(); Then when used we can get the following: Point p = new Point(16,32);Point p2 = new Point(16,32); Point p3 = new CounterPoint(16,32); System.out.println (p.equals(p2)); // this returns true System.out.println (p2.equals(p3)); // this returns false Using instance of instead of getclass could avoid to break liskov substitutability principle, but might introduce other problems (see in [1] the discussion about reflexivity, simmetry and transitivity). In such cases, and if possible, the suggestion is to avoid inheritance and use composition (in the example, a class Counter will have a private instance of Point). [2] J. Bloch, Effective Java, 2 edition. Upper Saddle River, NJ: Addison-Wesley, 2008.