QUESTION 1:

The code below does not compile. Infact when you try to compile it, it will give you the following error
non-static variable toPrint cannot be referenced from a static context

Change this program so that it will compile, and give an explanation to why your code works and why the old code did not work

```java
public class Question1 {
    /** The string we are going to print */
    public String toPrint = "Hello world";

    public static void main(String[] args){
        System.out.println(toPrint);
    }
}
```

SOLUTION 1:

There's two ways to solve this...
1. Put the variable in your function, as in make "toPrint" a local variable in "main".
2. It is also acceptable to make the variable static. Making "main" non-static however, is not a good idea.
3. Make an instance of the class Question1 in main, and print the instance variable of that object. For example...

```java
public class Question1 {
    /** The string we are going to print */
    public String toPrint = "Hello world";

    public static void main(String[] args){
        Question1 q = new Question1();
        System.out.println(q.toPrint);
    }
}
```
QUESTION 2:

For this question, refer to the Student class.
What is the output when you compile and run Question2?
If we took away the static property of LAST_ID what would the output of Question2 be? Why?

```java
public class Student{
    /** the last students id number */
    private static int LAST_ID = 0;
    /** the student's name */
    private String name;
    /** the students id number */
    private int id;

    /**
     * Constructor for student. Sets up information about the student
     *
     * @param _name The name of the student
     */
    public Student(String _name){
        name = _name;
        LAST_ID++;
        id = LAST_ID;
    }

    public String toString(){
        return name + "\t" + id;
    }
}

public class Question2{

    public static void main(String[] args){
        Student craig = new Student("Craig");
        Student abisola = new Student("Abisola");

        System.out.println(craig);
        System.out.println(abisola);
    }
}
```
**SOLUTION 2:**

Output before changing static property:
*Craig* 1
*Abisola* 2

Output after changing static property:
*Craig* 1
*Abisola* 1

The reason behind this is as follows…

When LAST_ID was static, it was instantiated only the first time an object was made from this class. Therefore, Craig got the ID 1, while for Abisola, the ID was 2 because the variable LAST_ID remembered the old value it had, it was not reinstatiated.

When LAST_ID was made non-static, it was forced to be reinstatiated on construction of Abisola, thus giving her an ID of 1.
QUESTION 3:

What does the following program output?

```java
public class Driver{

    private int[] myArray;

    public static void main(String args[]){
        Driver d = new Driver();
        d.doIt();
    }

    public Driver(){
        myArray = new int [8];
        myArray[3] = 5;
        myArray[5] = 3;
    }

    public void doIt(){
        for (int i = 5; i >= 0; i--){
            int place = myArray[i];
            i -= 1;
            myArray[place] = i;
        }

        for(int i = 0; i < 8; i++)
            System.out.print(myArray[i] + " ");
        System.out.println();
    }
}
```

SOLUTION 3:

Expected Output: 0 0 0 4 2 3 0 0

This question tests your ability to manage slightly more intricate array operations. The best way to solve this sort of question would be to write down the structure of the empty array and then go through the code sequentially, while constantly updating the structure as various operations are applied to the array.
<table>
<thead>
<tr>
<th>i</th>
<th>place</th>
<th>myArray</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0 1 2 3 4 5 0 0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0 1 2 3 4 0 0 0</td>
</tr>
<tr>
<td>Loop 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0 1 2 3 4 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0 1 2 3 4 2 0 0</td>
</tr>
<tr>
<td>Loop 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 1 2 3 4 0 0 0</td>
</tr>
<tr>
<td>Loop 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-</td>
<td>0 1 2 3 4 0 0 0</td>
</tr>
</tbody>
</table>