In-class 10: Logic coverage for source code
Due 8-November-2018 by 11:59pm

Names:

Purpose: Understand and practice applying logic coverage to program source code
Instruction: Work with your neighbors in groups. Consider an implementation of the old engineering joke: Good, Fast, Cheap.

```java
public final class GoodFastCheap {
    boolean good  = false;
    boolean fast  = false;
    boolean cheap = false;

    public void makeGood () {
        good = true;
        if (fast && cheap) { cheap = false; }
    }

    public void makeFast () {
        fast = true;
        if (good && cheap) { good = false; }
    }

    public void makeCheap () {
        cheap = true;
        if (fast && good) { fast = false; }
    }

    public void makeBad ()       { good = false; }
    public void makeSlow ()      { fast = false; }
    public void makeExpensive () { cheap = false; }

    public boolean isSatisfactory() {
        if ((good && fast) || (good && cheap) || (fast && cheap)) {
            return true;
        } return false;
    }

    public static void main(String[] args) {
        // Question:  How well do the following tests exercise the clauses?
        GoodFastCheap gfc = new GoodFastCheap();     // g f c
gfc.isSatisfactory();   // F F F
        gfc.makeGood();      gfc.isSatisfactory();   // T F F
        gfc.makeFast();      gfc.isSatisfactory();   // T T F
        gfc.makeCheap();     gfc.isSatisfactory();   // T F T
        gfc.makeSlow();      gfc.isSatisfactory();   // T F T
    }
}
```
Focus on the predicate in `isSatisfactory()`. We will start with the truth table. (Note: in reality, if tools are available, use the tools. For practice purpose and help you prepare for the exam: we will fill the truth table ourselves)

<table>
<thead>
<tr>
<th>Row#</th>
<th>g</th>
<th>f</th>
<th>c</th>
<th>P</th>
<th>Pg</th>
<th>Pf</th>
<th>Pc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- List possible test pairs that satisfy Predicate Coverage (PC)

- List possible test pairs that satisfy Clause Coverage (CC)

- List possible test pairs that satisfy Correlated Active Clause Coverage (CACC) (The first step is to identify the possible CACC pairs for each clause and then assemble the result into a CACC-adequate test set for all clauses.)
• List possible test pairs that satisfy Restricted Active Clause Coverage (RACC) (The first step is to identify the possible RACC pairs for each clause and then assemble the result into a RACC-adequate test set for all clauses.)

• At least 4 tests are needed for a RACC-adequate test. Why?

• What are the possible assertions for the JUnit tests?
• Does the given sequence (the `main()` method) achieve RACC?

• What happens if we refactor the predicate in `isSatisfactory()`

```java
public boolean isSatisfactory() {
    if (good && fast) return true;
    if (good && cheap) return true;
    if (fast && cheap) return true;

    return false;
}
```

---

**Grading rubric**

[Total: 10 points]: Done (or provide evidence of your attempt)

---

**Submission**

Submit the paper in class before you leave or take screen shots of your in-class exercise and submit them to Collab/inclass10. *Everyone submits this in-class exercise*, even if you work with partners.