# An introduction to software testing

**Marcello Bonsangue**

**UESTC, 10-14 July 2017**

Universiteit Leiden
The Netherlands

**Discover the world at Leiden University**

# Assignment

# Unit Testing for a Voting System

A voting system is a component that:

1. Is activated by an initiator
2. collects a vote from a group of external voter objects
3. compiles a report,
4. and returns it to the initiator.

It can be used, for example:

- to detect termination of a group of objects
- to elect a leader of a group of objects

# Unit Testing for a Voting System

```
public class Census
{
public int voting(vector<Voter> voters)
}
```

```
public interface Voter {
    public boolean vote();
}
```

The method voting(…) of the class Census expects an array of Voter objects. Once called, it

1. It calls the method vote() from each instance of Voter in the array voters
2. compiles a report in the form of an integer value,
3. and returns the report to the callee.

When a method vote() of a Voter object is called, it returns a Boolean value, its vote.

# Unit Testing for a Voting System

```
public class Census
{
public int voting(vector<Voter> voters)
}
```

```
public interface Voter {
    public boolean vote();
}
```

The expected behaviour of the method voting() of the class Census is given by the following informal specification:

1. if at least one voter voted false then the returned value must be the number (necessarily positive) of voter that have voted false
2. if all voters voted true then the returned value must be 0
3. if 1. and 2. do not hold then the returned value must be -1
4. every valid (i.e. non-null) voter must vote
5. no voter can vote more than once.

# Unit Testing for a Voting System

```
public class Census {
public int voting(vector<Voter> voters)
}
```

```
public interface Voter {
    public boolean vote();
}
```

1. Design a test suite for testing an implementation of the class Census with respect to all requirements 1, 2, 3, 4 and 5.
2. Implement in Java (or C++) an automated test environment for the unit testing of any implementation of the class Census with respect to your test suite. The testing environment get as input an implementation of the class Census of correct type and return if this implementation passes the tests or not. Since you do not have the code of Census yet you cannot change or manipulate it.
3. Give a possibly faulty implementation of the class Census in Java (or C++)
4. Apply your test environment to your implementation of the class Census and check if it pass your test.

# Interfaces

## Java

```
public class Census {
     public int voting(vector<Voter> voters);
}
```

```
public interface Voter {
    public boolean vote();
}
```

## C++

```
#include <vector>

class Census  {
    public: int voting(vector<Voter> voters);
}
```

```
class Voter {
   public: virtual boolean vote() = 0;
}
```

# Concluding words

Thank you!

You can always send an e-mail to:
m.m.bonsangue@liacs.leidenuniv.nl


liacs.leidenuniv.nl


Coming soon ...


莱顿大学科学学院


liacsCS


@ul_liacs


groups/2084197


User: liacsmedialab


liacs

Universiteit
Leiden
The Netherlands