

```
package generator;
import java.util.ArrayList;

/**
 * Class that extends from Instruction, that stores all the information about an assert instruction
 * @author Sergio Alcocer
 */
public class Assert extends Instruction {

    public static final int ASSERT_TRUE = 0;
    public static final int ASSERT_FALSE = 1;
    public static final int ASSERT_EQUALS = 2;

    private int assertType;
    private String objectInvolved;
    private String methodAssociated;
    private ArrayList<String> params;
    private String restCode;

    /**
     * Constructor of the class
     * @param assertType Type of the assert
     * @param objectInvolved Name of the object involved on the assert
     * @param methodName name of the method in which is being declared
     */
    public Assert(boolean assertType, String objectInvolved, String methodName) {
        super(Instruction.T_ASSERT);
        this.assertType = (assertType)?ASSERT_TRUE:ASSERT_FALSE;
        this.objectInvolved = objectInvolved;
        this.methodAssociated = methodName;
        params = new ArrayList<String>();

        updateCode();
    }

    /**
     * Constructor of the class for the Assert Equals
     * @param methodName
     * @param obj1
     * @param obj2
     */
}
```

```
    */
    public Assert(String methodName, String obj1, String obj2){
        super(Instruction.T_ASSERT);
        assertType = ASSERT_EQUALS;
        methodAssociated = methodName;
        params = new ArrayList<String>();
        params.add(obj1);
        params.add(obj2);
        updateCode();
    }
    /**
     * Adds a parameter piece of code to the list
     * @param paramData parameter code to be added
     */
    public void addParam(String paramData){
        params.add(paramData);

        updateCode();
    }

    private void updateCode(){
        if (assertType == ASSERT_EQUALS){
            code = "assertEquals(" + params.get(0) + ", " + params.get(1) + ");";
        }else{
            code = "assert" + ((assertType == ASSERT_TRUE)?"True(": "False(") + objectInvolved + "." + methodAssociated + "("
;
            if(params.size() > 0){
                for (int i = 0; i < params.size() - 1; i++){
                    code += params.get(i) + ", ";
                }
                code += params.get(params.size()-1) + ")";
            }else{
                code += ")";
            }

            code += restCode + ");";
        }
    }
}
```

```
/**
 * Sets the rest of the code (code that would be concatenated to the end of the assert Code)
 * @param _code rest of the code to be set
 */
public void setRestCode(String _code) {
    restCode = _code;
    updateCode();
}

/**
 * Gets the rest of the code (code that is concatenated to the end of the assert Code)
 * @return code rest of the code
 */
public String getRestCode() {
    return restCode;
}

/**
 * Gets the type of the assert
 * @return The type of the assert
 */
public int getAssertType() {
    return assertType;
}

/**
 * Gets the object involved (the one used to call the function inside the assert)
 * @return object involved
 */
public String getObjectInvolved() {
    return objectInvolved;
}

/**
 * Gets an array with all the parameters' code
 * @return ArrayList<String> with the parameters' code
 */
public ArrayList<String> getParams() {
    return params;
}
}
```