

# GUI 자동화 테스트(SikuliX) 과제

**목적 :** SikuliX 라는 도구를 사용하여 주어진 프로그램의 GUI를 테스트할 것.

## 수행 방법

✓ 주어진 시스템의 GUI 테스트를 SikuliX를 이용하여 작업하고 리그레션 테스트를 수행한다.

### 1. 소개

소프트웨어 개발에서 GUI 테스트는 주어진 응용 프로그램에 대해 GUI의 적절한 기능을 보장하고 작성된 사양(Specification)을 준수하는지 확인하는 작업이다. 테스트는 일반적으로 다양한 테스트 케이스를 통해 수행된다. 다양하고 많은 테스트 케이스를 수행시키는 것은 도구를 사용하여 자동화할 수 있다. 이 과제에서는 특히 SikuliX 에 중점을 둘 것이다. SikuliX 는 스크린 샷을 사용하여 작업 단계를 자동화하는 프리웨어다. SikuliX Script는 이미지 패턴을 사용하여 키보드/마우스 이벤트를 지시하는 GUI 상호 작용을 자동화하며 Jython 및 Java 라이브러리라고 할 수 있다. SikuliX Script의 핵심은 키보드와 마우스 이벤트를 적절한 위치에 전달하는 java.awt.Robot 와 화면에서 주어진 이미지 패턴을 검색하는 OpenCV 기반 C++ 엔진 (Test creator)으로 구성된 Java 라이브러리이다. SikuliX 가 이미지 패턴을 감지하면 프로그램 된 동작을 실행한다. C ++ 엔진은 JNI 를 통해 Java에 연결되며 각 플랫폼을 위하여 따로 컴파일해야 한다. Java 라이브러리 이외에도 단순하고 명확한 명령 집합으로 Jython 계층이 제공된다. SikuliX 는 SikuliX 를 기반으로하는 Java API이다.

### 2. 준비

실습은 두 부분으로 나뉜다. 작업 영역을 설정하고 SikuliX 에 익숙해져야 한다. 두 번째 부분은 Java로 작성된 샘플 프로그램의 테스트를 자동화하여야 하는데 이 부분이 채점의 대상이다.

다음 단계에 따라 작업을 수행하라.

- “설치 가이드” 문서를 따라 진행한다.
- 작업 환경을 설정한 후 아래 예시를 살펴보고 이해해보라 (해당 방법은 Autotest.zip 에서 사용된다)

### 3. 예시

다음은 skeleton 코드에서 메소드를 사용하는 방법에 대한 사례이다. 언제든지 더 많은 메소드를 만들 수 있지만, 보고서에서 논리는 동일하게 유지되어야 한다.

- 예 1. 사용자가 정의한 유사도로 이미지를 클릭하는 방법
- 예 2. 기본 유사도로 이미지를 클릭하는 방법
- 예 3. 텍스트를 클립 보드에 복사하고 비교하는 방법
- 예 4. 주어진 위치에 텍스트를 쓰는 방법
- 예 5. 재시도를 통해 화면에 이미지가 있는지 테스트하는 방법
- 예 6. 주기적으로 사진을 클릭하는 방법
- 예 7. 화면을 마우스 오른쪽 버튼으로 클릭하는 방법

다음과 같은 메소드들이 스켈레톤 프로젝트에서 복사된다.

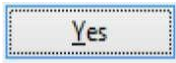
유사도 – SikuliX 는 픽셀 단위로 비교를 수행하며, 유사도는 발견된 이미지와 지정된 이미지 사이에 커버리지가 얼마나 큰지를 나타낸다.

### 예 1

```
/**
 * @param pictureUrl - location of the image - absolute or relative
 * @param similarity - from 0-1, sets the similarity of the image looked for
 * @returns
 */
click(String pictureUrl, Double similarity)
```

메소드는 0.0 에서 1.0 사이의 유사도를 제공하여 이미지를 클릭하려고 시도한다. SikuliX 가 그림을 찾지 못하면 유사도를 낮춘다(기본 유사성은 0.8).

### 예 2

예를 들어  (yes.jpg) 이미지를 클릭하기 위하여 메소드 호출은 `click("yes.jpg");`가 되어야 하며

```
/**
 * Calls out the click method with default similarity = 0.80
 * @param pictureUrl
 * @return
 */
click(String pictureUrl)
```

유사도 0.8로 click 메소드를 호출한다.

### 예 3

```
/**
 * Tries to copy text to the clipboard and the compare it to predefined text
 * @param textToCompare
 */
private void compareTextToClipboards(String textToCompare)
```

텍스트를 모두 선택한 후 미리 정의된 텍스트와 비교한다.  
복사가 작동하려면 SikuliX 가 먼저 텍스트 영역을 클릭해야 한다. 예제 3은 skeleton 프로젝트에서 시연되었다.

**예 4**

```
/**
 * Clicks on the given pictures and tries to enter text
 * @param pictureUrl
 * @param text
 */
private void write(String pictureUrl, String text)
```

주어진 그림을 클릭하고 가능한 경우 주어진 위치에 텍스트를 쓰려고 시도한다.  
예제 4 는 Skeleton 프로젝트에서도 데모 되어 있다.

**예 5**

```
/**
 * Returns true if pictures was clicked, if not, then returns false
 * @param pictureUrl
 * @return
 * @throws FindFailed
 */
private boolean verifyIfExists(String pictureUrl)
```

화면에 이미지가 있는지에 따라 Boolean 타입을 반환한다.  
또한 verifyIfExists(String pictureUrl, int retries) 메소드는 최대 RETRIES 횟수만큼 이미지를 체크하고 발견되면 true를 리턴한다.

**예 6**

```
/**
 * Tries to click on given image with given similarity for X retries
 * @param pictureUrl
 * @param similarity
 * @param retries
 */
public boolean click(String pictureUrl, Double similarity, int retries)
```

주어진 이미지를 클릭하려고 시도한다. 발견되면 true를 리턴한다. 사진이 움직일 때 유용함.

**예 7**

```
void rightClick()
```

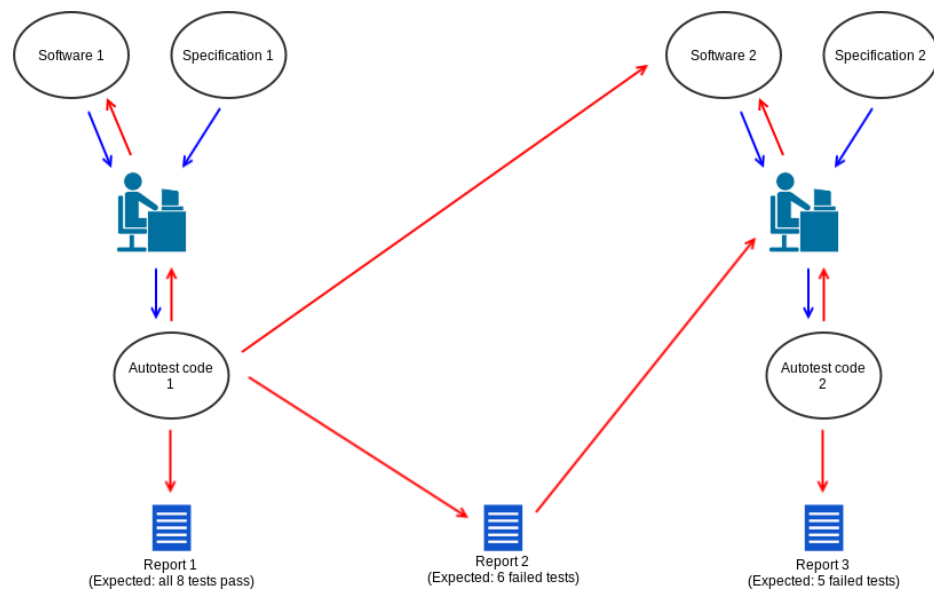
화면을 마우스 오른쪽 버튼으로 클릭한다. 위치는 미리 설정되어 있어야 하며 위치를 설정하려면 클릭 메소드로 위치를 클릭한다.

## SikuliX 요점 정리

- SikuliX 는 플래시 객체 자동화를 위한 광범위한 지원을 제공한다.
- SikuliX 는 웹 및 Windows 응용 프로그램을 자동화 할 수 있다.
- SikuliX 는 시각적 일치하는지 체크하여 화면에서 요소를 찾는다.
- SikuliX 에는 간단한 API 가 있다.
- SikuliX 는 소스 코드를 사용하여 탐색하지 않는다.

#### 4. 작업 과정

이번 과제는 다음 그림에 나타난 절차를 따라서 한다. 설명이 아래 추가되어 있다.



### 그림 1. 작업 순서

과제를 실행하기 위하여 "설치 가이드" 부분에서 가져온 skeleton 프로젝트를 기반으로 Autotest 코드를 작성해야 한다. 사양(Specifications)과 함께 두 가지 소프트웨어 버전이 제공된다.

버전 2 에는 프로그램의 핵심 기능이 변경된 새로운 기능이 도입되었다. Specification 1과 Specification 2를 비교할 때 차이점을 찾을 수 있을 것이다. 과제는 이 리그레션 테스트 단계를 자동화하는 것이다.

첫 번째 사이클에는 Specification 1의 모든 포인트를 포함하는 테스트 케이스 세트를 구축하는 것이다. 테스트 케이스를 먼저 Software1.jar 에 대해 실행하라.

두 번째 사이클은 Software2.jar 에 대해 동일한 테스트 케이스 세트를 실행하는 것이다. 이번 실행은 새 기능으로 인해 발생한 모든 테스트 실패를 감지해야 한다.

참고: Specification 변경으로 인해 오류가 발생할 수도 있다(그러나 이는 자동 테스트에 결함이 있음을 의미하지만 적절한 자동 테스트 결과를 얻으려면 수정해야 한다).

앞부분이 완료되면 반복 3 if-block에서 코드를 호출한다. Specification 2 에서 약간의 변경/업데이트를 찾을 수 있으며 이러한 변경 사항을 확인하려면 Specification 2와 Specification 1을 비교해야 한다.

작업을 요약하면:

1. 첫 번째 반복에서 software1.jar 에 대해 실행할 테스트 케이스 세트를 구축하라.
  1. Specification 1 에 제시된 각 Specification의 포인트를 다루어야 한다.
  2. 하나의 Specification 포인트는 보고서에 1줄의 행을 만들어야 한다(아래 예에서 테스트 결과를 보고서 파일에 추가하는 방법을 제시하였다) – 따라서 Specification 1 에 해당하는 보고서 파일에는 8개의 줄이 있어야 한다.
  3. 보고서를 저장하라 – 나중에 보고서를 제출하는데 모든 테스트는 통과하여야 한다.
2. Software2.jar 에 대해 테스트 케이스를 실행하라.
  1. 이전 개발 주기에서 올바르게 수행한 경우 autotest 는 자동화된 회귀 테스트하는 동안에 모든 오류를 찾아야 한다.
  2. 보고서 2 는 8개의 줄을 포함하여야 한다 (6개의 테스트는 실패).
  3. 모든 버그가 발견되지 않으면 1로 돌아간다.
  4. 보고서 1, 보고서 2 및 테스트 코드를 저장하는 것을 잊지 말 것.

## 5. 보고서

각각 자동 테스트에 의해 생성된 3개의 보고서 파일(다음 명명 스키마에 따라 이름이 지정됨)을 포함하여야 한다.

- Report1.pdf – Autotest 가 버전 1 에 대해 실행될 때 생성된다(8개의 결과를 포함해야 함).
- Report2.pdf – Autotest 가 버전 2 에 대해 실행될 때 생성된다(8개의 결과를 포함해야 함).
- Autotest1.java- Autotest 코드가 있는 파일(Software1 에 적용)
- Autotest2.java- Autotest 코드가 있는 파일(Software2 에 적용)
- 그림이 있는 리소스 폴더
- 버전 1,2 에 있는 버그들을 찾아 보고서 형식으로 작성

## 6. 평가 기준

- 코드, 보고서 없다면 점수 없음
- 제출 지연 시 점수 X / (연장 제출 X)
- 제출 할 zip 패키지는 다음 요소를 포함해야 한다.
  - 실행되는 Autotest 코드 + 리소스 폴더에 있는 캡처 파일
  - Report 1, 2 의 결과 (PDF 파일)
  - 발견한 버그들의 확인 및 내용 설명 (보고서 형식으로 상세히 작성)

## 7. 참고 사이트 및 참고 문헌

SikuliX 에 대한 자세한 정보는 다음 사이트에서 발견할 수 있다.

<https://github.com/RaiMan/SikuliX-2014/wiki/Usage-in-Java-programming>

<http://www.SikuliX.com/quickstart.html>

- [1] Matthew Haughn. "GUI testing". WhatIs.com®. Feb. 2014. Web. 15 March 2015, <http://whatis.techtarget.com/definition/GUI-testing-graphical-user-interface-testing>
- [2] SikuliX Doc Team. "How SikuliX works". doc.SikuliX.org, SikuliX X 1.0 documentation. Nov. 2012. Web. 10 February 2015, <http://doc.SikuliX.org/devs/system-design.html>
- [3] 2007-2013 Software Testing help, Selenium Vs SikuliX. Retrieved from <http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/09/SikuliX17.jpg>