Application for differential testing between Java and Kotlin using auto generated test cases

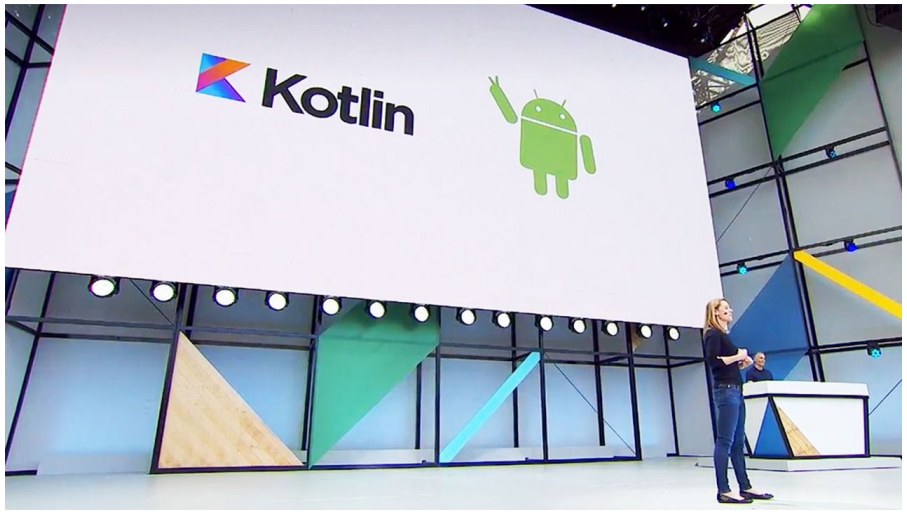
Team 4

Java - Kotlin Migration

Nowadays, Kotlin is used widely in many areas like Android development.

So recently, Java to Kotlin Migration occurs frequently.

After migrating the Java code into Kotlin code, we need to ensure two codes are identical.



Java - Kotlin Migration

```java
public class Stack<T> {
    private int capacity = 10;
    private int pointer  = 0;
    private T[] objects = (T[]) new Object[capacity];

    public void push(T o) {
        if(pointer >= capacity)
            throw new IllegalArgumentException("Stack exceeded capacity!");
        objects[pointer++] = o;
    }

    public T pop() {
        if(pointer <= 0)
            throw new IllegalArgumentException("Stack empty");
        return objects[--pointer];
    }

    public boolean isEmpty() { return pointer <= 0; }

}
```

Stack.java

```kotlin
class Stack<T> {
    private val capacity = 10
    private var pointer = 0
    private val objects = arrayOfNulls<Any>(capacity) as Array<T?>
    fun push(o: T) {
        require( value: pointer < capacity) { "Stack exceeded capacity!" }
        objects[pointer++] = o
    }

    fun pop(): T? {
        require( value: pointer > 0) { "Stack empty" }
        return objects[--pointer]
    }

    val isEmpty: Boolean
        get() = pointer < 1
}
```

Stack.kt

Is it translated successfully?

Java - Kotlin Migration

Java and Kotlin both generate the class file by compile which can run over JVM.

They use same standard library.

So, we made a program to generate and execute tests on Java and Kotlin class file to compare those outputs based on differential and metamorphic testing.

```
1  import java.util.Arrays
2  import java.util.Comparator
3
```

java import

```
1  import java.util.Arrays;
2  import java.util.Comparator;
3
```

kotlin import

Methodology - Generating test-suites

We used Evosuite to generate the tests-suites.

It generates JUnit tests covering different coverage criteria, like lines, branches, outputs and mutation testing from the target java class file.

The generated tests always pass over the target java class.

Now we can run test-suites over kotlin class file too.

```
public void test1() throws Throwable {
    Stack var1 = new Stack();
    var1.push((Object)null);
    Assert.assertFalse(var1.isEmpty());
    var1.pop();
    Assert.assertTrue(var1.isEmpty());
}
```

Example Test (Stack.java)

Methodology - Result

If all tests were passed, it means there are no critical mistake occurred in migration.

```
All 6 tests passed in kotlin class file.
```

If some tests were failed, it means there is a problem in migrated kotlin file. So, the migration was unsuccessful and should be rectified.

```
3 of 6 tests (test0, test1, test3) failed in kotlin class file.
```

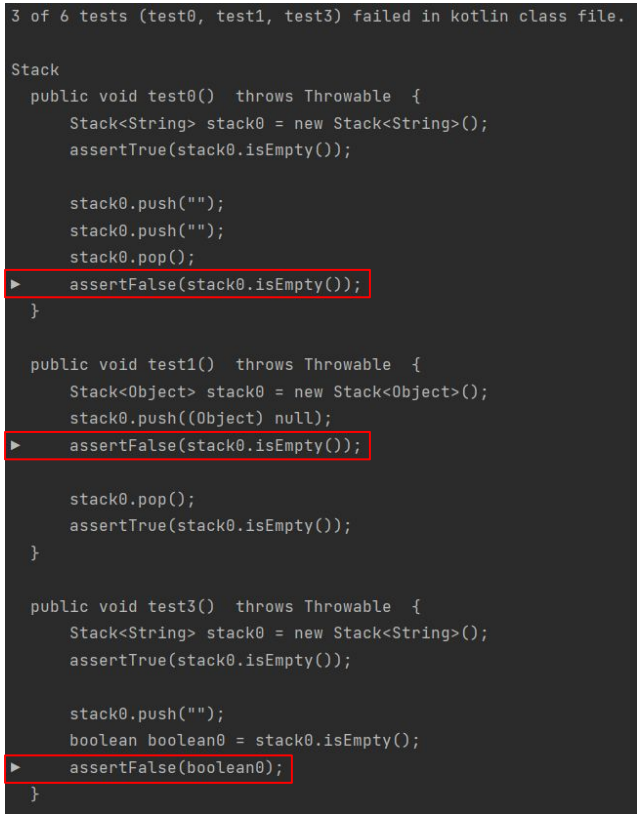
Methodology - Failure annotation

The program will show which tests were failed and annotate the failed lines of the tests for debugging.

In this example, we can easily detect mutation occured in Stack.isEmpty() function.

After modifying it, we can re-run the program to check whether the modification works correctly.

We can repeat it until all test-suites are passed.



Comparing Stack.java and Stack.kt

Evaluation

Evaluation Suite

- For evaluate effectiveness of our application
- Consisted of several pairs of Java and Kotlin
- Some of suite has mutations

Simple computations

Java

```java
class Outer01 {
    int num;
    void displayInner() {
        Inner01 inner = new Inner01();
        inner.print();
    }

    void increase() {
        num = num + 1;
    }

    void decrease() {
        num = num - 1;
    }

    int getNum() {
        return num;
    }
}
```

Kotlin

```kotlin
class Outer01 {
    var num = 0

    fun displayInner() {
        val inner = Inner01()
        inner.print()
    }

    fun increase() {
        num = num + 1
    }

    fun decrease() {
        num = num - 1
    }
}
```

Conditionals

Java

```java
if (itemNames == null || itemNames.length < 1) {
    throw new IllegalArgumentException(
      "STATUS command requires at least one data item name"
    );
}
```

Kotlin (identical)

```kotlin
require(
  !(itemNames == null || itemNames.size < 1)
) { "STATUS command requires at least one data item name" }
```

Kotlin (killed mutant)

```kotlin
require(
  itemNames != null && itemNames.size >= 0
) { "STATUS command requires at least one data item name" }
```

Inheritance

Java (inheriting Apache Commons)

```java
// Apache Commons net package, compiled with Java
import org.apache.commons.net.imap.IMAP;
import org.apache.commons.net.imap.IMAPCommand;
import org.apache.commons.net.imap.IMAPReply;

public class IMAPClient01 extends IMAP
{
    // implementation here
}
```

Kotlin (inheriting Apache Commons)

```kotlin
// Apache Commons net package, compiled with Java
import org.apache.commons.net.imap.IMAP
import org.apache.commons.net.imap.IMAPCommand
import org.apache.commons.net.imap.IMAPReply

class IMAPClient01 : IMAP() {
    // implementation here
}
```

String formats

Java

```
args
  .append('{')
  .append(message.getBytes(IMAP.__DEFAULT_ENCODING).length)
  .append('}'); // length of message
final int status = sendCommand(
  IMAPCommand.APPEND, args.toString()
);
```

Kotlin (identical)

```
args
  .append('{')
  .append(message.toByteArray(charset(__DEFAULT_ENCODING)).size)
  .append('}') // length of message
val status = sendCommand(
  IMAPCommand.APPEND, args.toString()
)
```

Kotlin (mutant not killed)

```
args
  .append(message.toByteArray(charset(__DEFAULT_ENCODING)).size)
val status = sendCommand(
  IMAPCommand.APPEND, args.toString()
)
```

Kotlin Specific Syntax - Nullable(?) syntax

Java

```
public String foo(String bar) {
    bar.toUppercase() // ⟶ if bar== null, NullPointerException!
}
```

Kotlin

```
fun foo(bar: String): String { // if bar == null, exception here!
    bar.toUppercase()
}
```

Kotlin with Nullable input parameter

```
fun foo(bar: String?): String? {
    return bar?.toUppercase() // if bar == null, returns null
}
```

Kotlin Specific Syntax - Syntactic sugar

Java

```java
public class Circle {
    private final int radius;

    public Circle(int radius) {
        if (radius ⩽ 0) throw new IllegalArgumentException("Radius must be bigger than 0.");
        this.radius = radius;
    }

}
```

Kotlin

```kotlin
class Circle(private val radius: Int) {

    init {
        require(radius > 0) { "Radius must be bigger than 0." }
    }
}
```

Kotlin Specific Syntax - Data Class

Java

```java
public class User01 {

    private String email;

    private String name;

    private String password;

    private Integer age;

    private Double height;

    // getter, setter

    public User01(String email, String name, String password, Integer age, Double height) {
        this.email = email;
        this.name = name;
        this.password = password;
        this.age = age;
        this.height = height;
    }
}
```

Kotlin Data Class

```kotlin
data class User01(
    var email: String? = null,
    var name: String? = null,
    var password: String? = null,
    var age: Int? = null,
    var height: Double? = null
)
```

Kotlin Specific Syntax - Data Class toString()

Java

```
User@15db9742
```

Kotlin Data Class

```
User(email=example@kaist.ac.kr, name=Sungmin Im, password=pass, age=26, height=182.5)
```

Kotlin Specific Syntax - Object

Java - Util Class with only static methods

```java
public class ArraySorter {

    public static byte[] sort(final byte[] array) {
        Arrays.sort(array);
        return array;
    }

    public static int[] sort(final int[] array) {
        Arrays.sort(array);
        return array;
    }

    // ...

    public static <T> T[] sort(final T[] array, final
Comparator<? super T> comparator) {
        Arrays.sort(array, comparator);
        return array;
    }

}
```

Kotlin Singleton Class(Object)

```kotlin
object ArraySorter {

    @JvmStatic
    fun sort(array: ByteArray?): ByteArray? {
        Arrays.sort(array)
        return array
    }

    @JvmStatic
    fun sort(array: IntArray?): IntArray? {
        Arrays.sort(array)
        return array
    }

    @JvmStatic
    fun <T> sort(array: Array<T>?, comparator:
Comparator<in T>?): Array<T>? {
        Arrays.sort(array, comparator)
        return array
    }

}
```

Kotlin Specific Syntax - Object

Java - Util Class with only static methods

```java
public class ArraySorter {

    public static byte[] sort(final byte[] array) {
        Arrays.sort(array);
        return array;
    }

    public static int[] sort(final int[] array) {
        Arrays.sort(array);
        return array;
    }

    // ...

    public static <T> T[] sort(final T[] array, final
Comparator<? super T> comparator) {
        Arrays.sort(array, comparator);
        return array;
    }

}
```

Kotlin Class(Companion Object)

```kotlin
class ArraySorter {

    companion object {

        @JvmStatic
        fun sort(array: ByteArray?): ByteArray? {
            Arrays.sort(array)
            return array
        }

        @JvmStatic
        fun sort(array: IntArray?): IntArray? {
            Arrays.sort(array)
            return array
        }

        @JvmStatic
        fun <T> sort(array: Array<T>?, comparator:
Comparator<in T>?): Array<T>? {
            Arrays.sort(array, comparator)
            return array
        }
    }

}
```

Evaluation

It works for…

- Simple computations and conditionals
- Inheritances including those from Java classes
- Syntactic sugars in Kotlin

It does not work for…

- String format check
- Non-null type checking and NullPointerException
- Static methods, singletons, and companion objects (in Kotlin)

Conclusion

What we made : Application that test identity of two codes with differential testing by auto generated test code

It was effective in aspect of

- Time-saving
- Effort-saving

It was less effective in aspect of

- Can’t use when class interface was changed
- Don’t allow reasonable changes
  - e.g. destination language’s philosophical choice, …
- A test case generated by source language should be executable in destination language

Future Work

We can do same works on other JVM language

- Groovy
- Scala
- Clojure
- …

We can use other test code generator

- Randoop
- SquareTest
- ...