Deliverable 5

Fault Injection

For deliverable 5 we implemented 5 faults into our code that would most but not all of the corresponding test cases to fail.

---

1. The first fault injected to our code was in the containsUpperAndLowerCase method. Original if
statement.

```
public static boolean containsUpperAndLowerCase(String test) {
        if (test != null) {
                Pattern pattern = Pattern.compile("^(?=.*?[A-Z])(?=.*?[a-z])[\\w|\\W]*$");
                Matcher matcher = pattern.matcher(test);
                return matcher.matches();
        }
        return false;
}
```

in this method, we removed "(?=.*?[A-Z])" *from Pattern pattern = Pattern.compile("^(?=.?[A-Z])(?=.?[a-z])[\w|\W]$");*

Original
results.

09	true is returned when string contains upper and lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"Wright"	true	true	Passed
10	false is returned when string contains only lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"wright"	false	false	Passed
11	false is returned when string contains only upper case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"WRIGHT"	false	false	Passed
12	false is returned when string is null	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	""	false	false	Passed

Test results after fault
injection.

09	true is returned when string contains upper and lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"Wright"	true	true	Passed
10	false is returned when string contains only lower case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"wright"	false	true	Failed
11	false is returned when string contains only upper case letters	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	"WRIGHT"	false	false	Passed
12	false is returned when string is null	OpenmrsUtil.java	containsUpperAndLowerCase(String test)	""	false	false	Passed

This fault as you can see causes the second test case to fail because the pattern didn't see any uppercase letters.

---

2. The second fault was injected into the format method.

Original Format method

```
public static String format(Date date, Locale locale, FORMAT_TYPE type) {
        if (date == null || locale == null || type == null) {
                return "";
        }
        //log.debug("Formatting date: " + date + " with locale " + locale);

        DateFormat dateFormat;

        if (type == FORMAT_TYPE.TIMESTAMP) {
                dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);
        } else if (type == FORMAT_TYPE.TIME) {
                dateFormat = DateFormat.getTimeInstance(DateFormat.MEDIUM, locale);
        } else {
                dateFormat = DateFormat.getDateInstance(DateFormat.SHORT, locale);
        }
        return dateFormat.format(date);
}
```

Here instead of returning "dateFormat.format(date);" we return dateFormat.toString(); This causes all format test cases to fail as it now causes a java.text error.

Original results

13	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,US,DATE	5/13/98	5/13/98	Passed
14	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,JAPAN,DATE	98/05/13	98/05/13	Passed
15	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998/14:30,US,TIME	2:30:00 PM	2:30:00 PM	Passed
16	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIME	14:30:00	14:30:00	Passed
17	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,US,TIMESTAMP	May 13, 1998 2:30:00 PM EDT	May 13, 1998 2:30:00 PM EDT	Passed
18	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIMESTAMP	1998/05/13 14:30:00 EDT	1998/05/13 14:30:00 EDT	Passed
19	If any of the arguments are null then method will not fail it will return an empty string	Format.java	format(Date date	5/13/1998/14:30,,TIME	null	null	Passed
20	If any of the arguments are null then method will not fail it will return an empty string	Format.java	format(Date date	,US,TIME	null	null	Passed

Test results after fault injection

13	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,US,DATE	5/13/98	java.text.SimpleDateFormat@8629ad2d	Failed
14	Take inputted date, location, and format_type:DATE and outputs the formatted date	Format.java	format(Date date	5/13/1998,JAPAN,DATE	98/05/13	java.text.SimpleDateFormat@6ed4cba0	Failed
15	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998/14:30,US,TIME	2:30:00 PM	java.text.SimpleDateFormat@8400729	Failed
16	Take inputted date, location, and format_type:TIME and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIME	14:30:00	java.text.SimpleDateFormat@49ec34c8	Failed
17	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,US,TIMESTAMP	May 13, 1998 2:30:00 PM EDT	java.text.SimpleDateFormat@9aeab5b	Failed
18	Take inputted date, location, and format_type:TIMESTAMP and outputs the formatted date	Format.java	format(Date date	5/13/1998 /14:30,JAPAN,TIMESTAMP	1998/05/13 14:30:00 EDT	java.text.SimpleDateFormat@186995c2	Failed

---

3. With this fault inside of the for loop in the method isStringinArray we changed the if statement "(str.equals(anArr))" to (!str.equals(anArr))" Original code

```
public static boolean isStringInArray(String str, String[] arr) {
        boolean retVal = false;

        if (str != null && arr != null) {
                for (String anArr : arr) {
                        if (str.equals(anArr)) {
                                retVal = true;
                        }
                }
        }
        return retVal;
}
```

Original results

001	true returned when string is in array	OpenmrsUtil.java	isStringinArray(String str	"Sam"&"Sam","Wright"	true	true	Passed
02	false returned when string is not in array	OpenmrsUtil.java	isStringinArray(String str	"Dylan"&"Sam","Wright"	false	false	Passed
03	false returned when string is null	OpenmrsUtil.java	isStringinArray(String str	""&"Sam","Wright"	false	false	Passed

Test results after fault injection

001	true returned when string is in array	OpenmrsUtil.java	isStringinArray(String str	"Sam"&"Sam","Wright"	true	true	Passed
02	false returned when string is not in array	OpenmrsUtil.java	isStringinArray(String str	"Dylan"&"Sam","Wright"	false	true	Failed
03	false returned when string is null	OpenmrsUtil.java	isStringinArray(String str	""&"Sam","Wright"	false	true	Failed

here this causes the first test case to pass and the other two for this method to fail.

4. For convertToInteger(Long long) we changed a "||" to "&&" Original code

```
public static Integer convertToInteger(Long longValue) {
        if (longValue < Integer.MIN_VALUE || longValue > Integer.MAX_VALUE) {
                throw new IllegalArgumentException(longValue + " cannot be cast to Integer without changing its value.");
        }
        return longValue.intValue();
}
```

Original results

07	Long value is turned into an Integer value	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483647	2147483647	2147483647	Passed
08	Long value is too large to be turned into an Integer, it throws the correct exception	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483648	java.lang.IllegalArgumentException: 2147483648 cannot be cast to Integer without changing its value.	java.lang.IllegalArgumentException: 2147483648 cannot be cast to Integer without changing its value.	Passed

Test results after fault injection

07	Long value is turned into an Integer value	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483647	2147483647	2147483647	Passed
08	Long value is too large to be turned into an Integer, it throws the correct exception	OpenmrsUtil.java	convertToInteger(Long longvalue)	2147483648	java.lang.IllegalArgumentException: 2147483648 cannot be cast to Integer without changing its value.	-2147483648	Failed

the fault caused the int to become negative because the long was too long to be converted to an int. Since the fault affected the if statement that catched this, it came out as negative instead of outputting and exception.

---

5. For our final fault we chose to change the ".equalsIgnoresCase" to just ".equals" in the method nullSafeEqualsIgnoreCase. Original code

```
public static boolean nullSafeEqualsIgnoreCase(String s1, String s2) {
        if (s1 == null) {
                return s2 == null;
        } else if (s2 == null) {
                return false;
        }

        return s1.equalsIgnoreCase(s2);
}
```

Original results

21	returns true if strings are equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Sam"	true	true	Passed
22	returns false if strings are not equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Wright"	false	false	Passed
23	returns false if one of the strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam",""	false	false	Passed
24	returns true if strings are equal but cases are different	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"SAM","sam"	true	true	Passed
25	returns true if both strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"",""	true	true	Passed

Test results after fault injection

21	returns true if strings are equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Sam"	true	true	Passed
22	returns false if strings are not equal	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam","Wright"	false	false	Passed
23	returns false if one of the strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"Sam",""	false	false	Passed
24	returns true if strings are equal but cases are different	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"SAM","sam"	true	false	Failed
25	returns true if both strings are null	OpenmrsUtil.java	nullSafeEqualsIgnoreCase(String s1	"",""	true	true	Passed