PV260 Software Quality: Static Code Analysis and Manual Code Review

Name and UČO:

2 Findings

```
public String getFullName(Person person){
        return person.firstName() + " " + person.surname();
}
```

klodye 22/3/2015 19:11
Comment [1]: person may be null (null dereference)

klodye 22/3/2015 19:11
Comment [2]: person may be null (null dereference)

----

3 Findings

```
public void foo(String name){
        if(name.equals("")){
                System.out.println("This one is The Nameless");
        }
        if(name == "John"){
                System.out.println("This one is not original at all.");
        }
}
```

klodye 22/3/2015 19:12
Comment [3]: name may be null (null dereference)

klodye 22/3/2015 19:19
Comment [4]: when checking against Empty String, .isEmpty() should be used

klodye 22/3/2015 19:12
Comment [5]: two objects checked for quality via '==' (should use .equals())

----

1 Finding

```
public void foo(String s){
        if(s == null){
        } else {
                s = s + "I am king of the world.";
        }
}
```

klodye 22/3/2015 19:13
Comment [6]: Unnecessary if condition

----

2 Findings

```
public void foo(String s){
        String k = s;
        if(s != null & !s.isEmpty()){
                System.out.println(s + " go home. You are drunk!");
        }
}
```

klodye 22/3/2015 19:13
Comment [7]: k variable is never used

klodye 22/3/2015 19:13
Comment [8]: && should be used

----

2 Findings

```
public String foo(){
        int k = 0;
        String result = null;
        if(k > 2){
                result = "'k' is greater than 2";
        }
        return result;
}
```

klodye 22/3/2015 19:14
Comment [9]: condition is always false

klodye 22/3/2015 19:22
Comment [10]: null is always returned

2 Findings

```
public String foo(){
	String result = "";
	int j = 0;
	while( j < 256484){
		result = result + "," + j;
	}
	return result;
}
```

klodye 22/3/2015 19:15
Comment [11]: use of String concatenation in while loop (StringBuilder or StringBuffer should be used)

klodye 22/3/2015 19:16
Comment [12]: j is always the same (infinite loop)

----

1 Finding

```
public boolean customIsEmpty(String s){
	if(s == null || (s != null && s.trim().isEmpty())){
		System.out.println("'s' is empty or null");
	return true;
	}
	return false;
}
```

klodye 22/3/2015 19:17
Comment [13]: statement always true at this point

----

2 Findings

```
public void writeIntoFile(String fileName){
	try{
		File result = new File(fileName);
		FileOutputStream fos = new FileOutputStream(result);
		// do writing
		fos.flush();
	}catch(IOException ioe){}
}
```

klodye 22/3/2015 19:17
Comment [14]: fos is never closed

klodye 22/3/2015 19:17
Comment [15]: empty catch clause

----

1 Finding

```
public void writeIntoFile(String fileName) throws IOException{
	File result = new File(fileName);
	FileOutputStream fos = new FileOutputStream(result);
	// do writing
	fos.flush();
	fos.close();
}
```

klodye 22/3/2015 19:18
Comment [16]: fos may not get closed in case IOException is thrown