

## Sample Candidate

Test ID: 120450389594801 | 9876543210 | sample@email.com

Test Date: September 18, 2022

### Core Java (Advanced Level)

26 /100



### Automata-Selenium

87 /100



### Core Java (Advanced Level)

26 / 100

#### Basic Java

33 / 100

#### OOPS Concepts

0 / 100

#### Advanced Java

40 / 100

### Automata-Selenium

87 / 100

#### Programming Practices

33 / 100

#### Functional Correctness

100 / 100

## 1 | Introduction

### About the Report

This report provides a detailed analysis of the candidate's performance on different assessments. The tests for this job role were decided based on job analysis, O\*Net taxonomy mapping and/or criterion validity studies. The candidate's responses to these tests help construct a profile that reflects her/his likely performance level and achievement potential in the job role

This report has the following sections:

The **Summary** section provides an overall snapshot of the candidate's performance. It includes a graphical representation of the test scores and the subsection scores.

The **Insights** section provides detailed feedback on the candidate's performance in each of the tests. The descriptive feedback includes the competency definitions, the topics covered in the test, and a note on the level of the candidate's performance.

The **Response** section captures the response provided by the candidate. This section includes only those tests that require a subjective input from the candidate and are scored based on artificial intelligence and machine learning.

The **Proctoring** section captures the output of the different proctoring features used during the test.

### Score Interpretation

All the test scores are on a scale of 0-100. All the tests except personality and behavioural evaluation provide absolute scores. The personality and behavioural tests provide a norm-referenced score and hence, are percentile scores. Throughout the report, the colour codes used are as follows:

- Scores between 67 and 100
- Scores between 33 and 67
- Scores between 0 and 33

## 2 | Insights

### Core Java (Advanced Level)

 26 / 100

This test measures the knowledge of basic Java constructs, OOP concepts, files and exception handling and advanced Java concepts like generics, collections, threads, strings and concurrency.

- The candidate is aware of the basic syntax and structure of Core Java (Advanced Level) but needs to put in substantial effort to improve her/his conceptual knowledge and understanding of algorithms.
- S/he should start by trying to write small programs to improve her/his programming skills.

### 3 | Response

#### Automata-Selenium

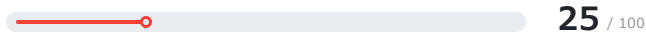
Code Replay  87 / 100

#### Question 1 (Language: Java Selenium)

A website URL is provided at the end of this section. On any given day, various users log into the website. Some login attempts are successful while some are not. A web developer has to scale up the website and therefore wants to know the count of successful logins on a given day. The arrays of usernames and the corresponding passwords used for different login attempts are given. Find the count of the successful logins for the URL.

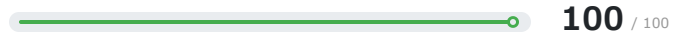
#### Scores

##### Programming Practices



Low readability, low on program structure. The source code is poorly formatted and contains redundant/improper coding constructs.

##### Functional Correctness



Functionally correct source code. Passes all the test cases in the test suite for a given problem.

#### Final Code Submitted

Compilation Status: Pass

```

1 // Sample code to read input and write output:
2
3 /*
4 import java.util.*;
5 import org.openqa.selenium.By;
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.WebElement;
8 import org.openqa.selenium.support.ui.WebDriverWait;
9 import org.openqa.selenium.remote.DesiredCapabilities;
10 import org.openqa.selenium.remote.RemoteWebDriver;
11 import java.net.URL;
12
13 public class Solution
14 {
15     public static void main(String args[] ) throws Exception
16     {
17         // Use either of these methods for input
18
19         //BufferedReader
20         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
21         String name = br.readLine(); // Read input from STDIN
22         System.out.println("Hello " + name); // Write output to STDOUT
23
24     }
25 }

```

#### Code Analysis

##### Errors/Warnings

There are no errors in the candidate's code.

##### Structural Vulnerabilities and Errors

##### Readability & Language Best Practices

Line 50: Variables are given very short names.

##### Performance & Correctness

Line 34,42: Using the '\*' form of import should be avoided - java.util.\*.

Line 73: The code consist of empty blocks.

Line 37,38: Avoid unused imports such as 'org.openqa.selenium.WebElement'

Line 47: A method/constructor shouldnt explicitly throw java.lang.Exception

Line 73: Avoid catching generic exceptions such as NullPointerException, RuntimeException, Exception in try-catch block

Line 73: Avoid empty catch blocks

```

24 //Scanner
25 Scanner s = new Scanner(System.in);
26 String name = s.nextLine(); // Read input from STDIN
27 System.out.println("Hello " + name); // Write output to STDOUT
28 }
29 }
30 */
31
32 // Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
33
34 import java.util.*;
35 import org.openqa.selenium.By;
36 import org.openqa.selenium.WebDriver;
37 import org.openqa.selenium.WebElement;
38 import org.openqa.selenium.support.ui.WebDriverWait;
39 import org.openqa.selenium.remote.DesiredCapabilities;
40 import org.openqa.selenium.remote.RemoteWebDriver;
41 import java.net.URL;
42 import java.io.*;
43 import org.openqa.selenium.Alert;
44
45 public class Solution
46 {
47     public static void main(String args[] ) throws Exception
48     {
49
50         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
51         int count = Integer.parseInt(br.readLine());
52         String email_ids = br.readLine();
53         String passwords = br.readLine();
54
55         String[] emails = email_ids.split(" ");
56         String[] password = passwords.split(" ");
57         int total_count=0;
58         WebDriver driver = new RemoteWebDriver(new URL("http://127.0.0.1:9515"),DesiredCapabilities.chrome());
59         for(int i=0;i<count;i++){
60             driver.get("https://a2z.aspiringminds.com/selenium/q0QvXGVGeNdiqBEUhVJBML93r_2B_2BiKnkPCd3jMIU2Dm40u_2Bn_2F9jwzLfMgzelifCPmYWUIUXuP_2FTNk8DMtinGtFs056GsMV81j_2F7BQvNDApY_3D/login");
61             driver.findElement(By.id("email")).sendKeys(emails[i]);
62             driver.findElement(By.id("password")).sendKeys(password[i]);
63             driver.findElement(By.id("login_button")).click();
64
65
66             try{
67                 Alert alert =driver.switchTo().alert();

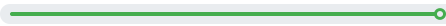
```

```

67
68     if(alert.getText().toLowerCase().contains("successful"))
69         total_count++;
70     alert.accept();
71
72     }
73     catch(Exception e){
74
75
76     }
77
78     }
79
80
81     System.out.println(total_count);
82
83 }
84 }

```

**Test Case Execution** Passed TC: 100%

Total score  5/5

<b>100%</b> Basic(5/5)	<b>0%</b> Advance(0/0)	<b>0%</b> Edge(0/0)
---------------------------	---------------------------	------------------------

**Test Cases: Deep Dive**

**Compilation Statistics**

<b>19</b>	<b>14</b>	<b>5</b>	<b>0</b>	<b>1</b>	<b>6</b>
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

---

Response time:	<b>00:39:15</b>
Average time taken between two compile attempts:	<b>00:02:04</b>
Average test case pass percentage per compile:	<b>12.63%</b>

## i Test Case Execution

There are three types of test-cases for every coding problem:

**Basic:** The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

**Advanced:** The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

**Edge:** The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

### Question 2 (Language: Java Selenium)

AM Store is an online shopping website. A web developer wants to implement the functionality of searching the products listed. Help the developer find the price of the product given the name by which the product is listed or else print -1 in case the product does not exist. The website URL is provided at the end of this section.

#### Scores

##### Programming Practices



Low readability, low on program structure. The source code is poorly formatted and contains redundant/improper coding constructs.

##### Functional Correctness



Functionally correct source code. Passes all the test cases in the test suite for a given problem.

#### Final Code Submitted

Compilation Status: Pass

```

1 // Sample code to read input and write output:
2
3 /*
4 import java.util.*;
5 import org.openqa.selenium.By;
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.WebElement;
8 import org.openqa.selenium.support.ui.WebDriverWait;
9 import org.openqa.selenium.remote.DesiredCapabilities;
10 import org.openqa.selenium.remote.RemoteWebDriver;
11 import java.net.URL;
12
13 public class Solution
14 {
15     public static void main(String args[] ) throws Exception
16     {
17         // Use either of these methods for input

```

#### Code Analysis

##### Errors/Warnings

There are no errors in the candidate's code.

##### Structural Vulnerabilities and Errors

###### Readability & Language Best Practices

Line 48: Variables are given very short names.

###### Performance & Correctness

Line 34,41: Using the '\*.\*' form of import should be avoided - java.util.\*.

Line 38: Avoid unused imports such as

'org.openqa.selenium.support.ui.WebDriverWait'

Line 46: A method/constructor shouldnt explicitly throw java.lang.Exception

```

17
18
19 //BufferedReader
20 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
21 String name = br.readLine(); // Read input from STDIN
22 System.out.println("Hello " + name); // Write output to STDOUT
T
23
24 //Scanner
25 Scanner s = new Scanner(System.in);
26 String name = s.nextLine(); // Read input from STDIN
27 System.out.println("Hello " + name); // Write output to STDOUT
T
28 }
29 }
30 */
31
32 // Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
33
34 import java.util.*;
35 import org.openqa.selenium.By;
36 import org.openqa.selenium.WebDriver;
37 import org.openqa.selenium.WebElement;
38 import org.openqa.selenium.support.ui.WebDriverWait;
39 import org.openqa.selenium.remote.DesiredCapabilities;
40 import org.openqa.selenium.remote.RemoteWebDriver;
41 import java.io.*;
42 import java.net.URL;
43
44 public class Solution
45 {
46     public static void main(String args[] ) throws Exception
47     {
48         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
49         String product = br.readLine();
50         WebDriver driver = new RemoteWebDriver(new URL("http://127.0.0.1:9515"),DesiredCapabilities.chrome());
51         driver.get("https://a2z.aspiringminds.com/selenium/YEp27CBrbrzp4a91e5BUPgl03_2FxVAv79SAMtir84jce6mzM25ImPx3cisVM1HryZT_2F5C7hnfrl0lc9uhLeMTtr8V6d5W2re0Tl87dsHXcPY_3D/product_s");
52         String product_price = "-1";
53         List<WebElement> product_list = driver.findElements(By.className("caption"));
54         for(int i=0;i<product_list.size();i++)
55         {
56             String product_name = product_list.get(i).findElement(By.tagName("h3")).getText();
57             if(product.equals(product_name)){

```



```

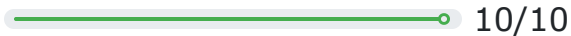
58
59         product_price = product_list.get(i).findElement(By.tag
Name("p")).getText().split("\\.")[1];
60         break l1;
61     }
62
63
64 }
65
66     System.out.println(product_price);
67
68
69
70
71 }
72 }

```

### Test Case Execution

Passed TC: 100%

Total score



**100%**

Basic(10/10)

**0%**

Advance(0/0)

**0%**

Edge(0/0)

### Test Cases: Deep Dive

### Compilation Statistics

5

Total attempts

3

Successful

2

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:14:53

Average time taken between two compile attempts:

00:02:59

Average test case pass percentage per compile:

28%

## i Test Case Execution

There are three types of test-cases for every coding problem:

**Basic:** The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

**Advanced:** The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

**Edge:** The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

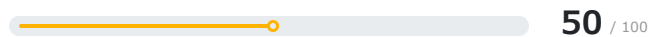
### Question 3 (Language: Java Selenium)

AM-Social website is a platform where various writers submit their blogs. The writers want to improve the content of their blogs and hence need some statistical data. They want to find the words that appear 'n' number of times in the blog.

Write an algorithm that returns the words in an alphabetical order with frequency 'n' in the blog or returns '-1' if no word exists with the given frequency. The website link is provided at the end of this section.

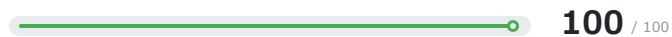
#### Scores

##### Programming Practices



High readability, low on program structure. The source code contains redundant/improper coding constructs and a few readability and formatting issues.

##### Functional Correctness



Functionally correct source code. Passes all the test cases in the test suite for a given problem.

#### Final Code Submitted

Compilation Status: Pass

```
1 // Sample code to read input and write output:
2
3 /*
4 import java.util.*;
5 import org.openqa.selenium.By;
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.WebElement;
8 import org.openqa.selenium.support.ui.WebDriverWait;
9 import org.openqa.selenium.remote.DesiredCapabilities;
```

#### Code Analysis

##### Errors/Warnings

There are no errors in the candidate's code.

##### Structural Vulnerabilities and Errors

##### Readability & Language Best Practices

Line 48: Variables are given very short names.

```

10 import org.openqa.selenium.remote.RemoteWebDriver;
11 import java.net.URL;
12
13 public class Solution
14 {
15     public static void main(String args[] ) throws Exception
16     {
17         // Use either of these methods for input
18
19         //BufferedReader
20         BufferedReader br = new BufferedReader(new InputStreameader(System.in));
21         String name = br.readLine(); // Read input from STDIN
22         System.out.println("Hello " + name); // Write output to STDOUT
23
24         //Scanner
25         Scanner s = new Scanner(System.in);
26         String name = s.nextLine(); // Read input from STDIN
27         System.out.println("Hello " + name); // Write output to STDOUT
28     }
29 }
30 */
31
32 // Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
33
34 import java.util.*;
35 import org.openqa.selenium.By;
36 import org.openqa.selenium.WebDriver;
37 import org.openqa.selenium.WebElement;
38 import org.openqa.selenium.support.ui.WebDriverWait;
39 import org.openqa.selenium.remote.DesiredCapabilities;
40 import org.openqa.selenium.remote.RemoteWebDriver;
41 import java.net.URL;
42 import java.io.*;
43
44 public class Solution
45 {
46     public static void main(String args[] ) throws Exception
47     {
48         BufferedReader br = new BufferedReader(new InputStreameader(System.in));
49         int count = Integer.parseInt(br.readLine());
50         String required_key = "-1";
51         Map<String,Integer> data = new TreeMap<>();
52         WebDriver driver = new RemoteWebDriver(new URL("http://127.0.0.1:9515"),DesiredCapabilities.chrome());
53
54         driver.get("https://a2z.aspiringminds.com/selenium/KrsKNgJQH

```

## Performance & Correctness

Line 34,42: Using the '\*' form of import should be avoided - java.util.\*.

Line 37,38: Avoid unused imports such as 'org.openqa.selenium.WebElement'

Line 46: A method/constructor shouldnt explicitly throw java.lang.Exception

```

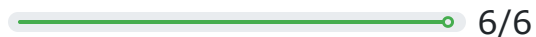
pXCO64kCUfF4oSBDdpOlzpVufCnN_2FsA18QCjJHaRqRGT2oDdSGP9
Y1rCJFA4lrFLcYhOrcr5Roj3WcaCiDkX_2FnORqyBWEBt6x4_3D/blog");
55     String text = driver.findElement(By.id("content")).getText();
56     String[] words = text.toLowerCase().split(" ");
57
58     for(int i=0;i< words.length;i++)
59     {
60         if(data.containsKey(words[i]))
61             data.put(words[i],data.get(words[i])+1);
62         else
63             data.put(words[i],1);
64     }
65
66
67     Set<String> keys = data.keySet();
68     l1:for(String key:keys){
69         if(data.get(key)==count){
70             required_key = key;
71             break l1;
72         }
73     }
74     System.out.println(required_key);
75
76 }
77 }

```

### Test Case Execution

Passed TC: 100%

Total score



**100%**

Basic(6/6)

**0%**

Advance(0/0)

**0%**

Edge(0/0)

### Test Cases: Deep Dive

### Compilation Statistics

10

Total attempts

4

Successful

6

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:20:27

Average time taken between two compile attempts:

00:02:03

Average test case pass percentage per compile:

16.67%

## Test Case Execution

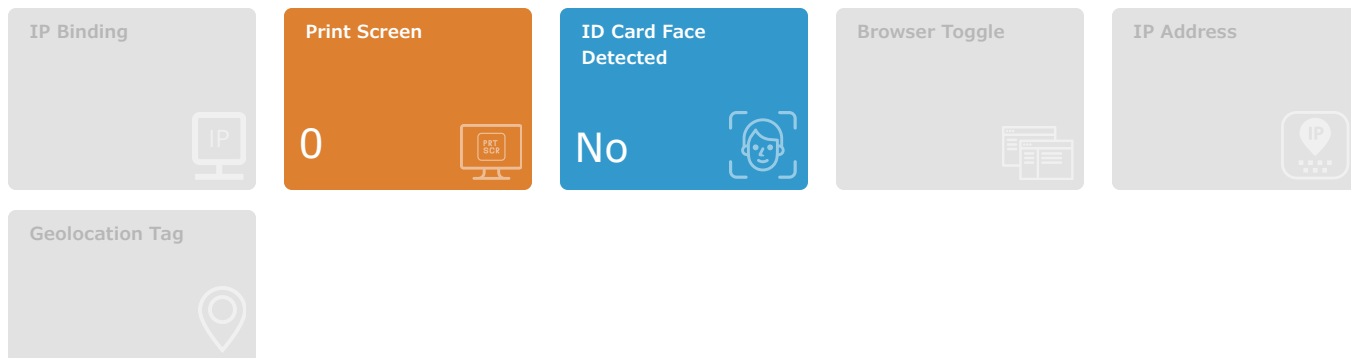
There are three types of test-cases for every coding problem:

**Basic:** The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

**Advanced:** The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

**Edge:** The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

## 4 | Proctoring



### AI Proctoring Information

- Print Screen:** The number of times the candidate attempted to take a screenshot of the assessment screen using the “print screen” function on their device. Note: This impacts proctoring index.
- ID Card Face Detected:** Looks at the candidate images captured during the assessment and flags anywhere different people appear to be present. Snapshots are included in the report.
- Browser Toggle:** Either the proportion of time the candidate spent focused on a tab/window other than that of assessment screen (%), or the number of times the candidate toggled to another tab/window (count). Note: This impacts proctoring index.
- IP Address:** Confirms that the candidate took the assessment from the specified IP address(s).
- Geolocation Tag:** Detects whether the candidate attempted the assessment from a location beyond the distance set by the administrator.