Sample Candidate

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

Test Date: September 18, 2022

Core Java (Advanced Level)	Automata-Selenium
26 /100	87 /100

Core Java (Advanced Level)

26 / 100

Basic Java	OOPS Concepts	Advanced Java
33 / 100	0 / 100	40 / 100

Automata-Selenium

87 / 100

Programming Practices	Functional Correctness
33 / 100	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

25 / 100

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

Functional Correctness

100 / 100

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 InputStreamRea
   der(System.in));
21         String name = br.readLine();          // Read input from STDIN
22         System.out.println("Hello " + name);    // Write output to STDOU
   T
23 
```

Code Analysis

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites 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

```
      //Scanner
      Scanner s = new Scanner(System.in);
      String name = s.nextLine();          // Read input from STDIN
      System.out.println("Hello " + name);    // Write output to STDOUT
   }
}
*/

// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail

import java.util.*;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import java.net.URL;
import java.io.*;
import org.openqa.selenium.Alert;

public class Solution
{
   public static void main(String args[] ) throws Exception
   {

      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int count = Integer.parseInt(br.readLine());
      String email_ids = br.readLine();
      String passwords = br.readLine();

      String[] emails = email_ids.split(" ");
      String[] password = passwords.split(" ");
      int total_count=0;
      WebDriver driver = new RemoteWebDriver(new URL("http://127.0.0.1:9515"),DesiredCapabilities.chrome());
      for(int i=0;i<count;i++){
          driver.get("https://a2z.aspiringminds.com/selenium/q0QvXGVGeNdiqBEUhVJBML93r_2B_2BiKnkPCd3jMIU2Dm40u_2Bn_2F9jwzLfMgzelifCPmYWUIUXuP_2FTNk8DMtinGtFs056GsMV81j_2F7BQvNDDApY_3D/login");
          driver.findElement(By.id("email")).sendKeys(emails[i]);
          driver.findElement(By.id("password")).sendKeys(password[i]);
          driver.findElement(By.id("login_button")).click();


          try{
              Alert alert =driver.switchTo().alert();
```

```
                if(alert.getText().toLowerCase().contains("successful"))
                    total_count++;
                alert.accept();

            }
⚠           catch(Exception e){


            }

        }


        System.out.println(total_count);

    }
}
```

Test Case Execution

Passed TC: **100%**

Total score 5/5

100%	**0%**	**0%**
Basic(**5**/5)	Advance(**0**/0)	Edge(**0**/0)

Test Cases: Deep Dive

Compilation Statistics

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

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

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

25 / 100

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

Functional Correctness

100 / 100

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     {
        // Use either of these methods for input
```

Code Analysis

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites 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

```java


        //BufferedReader
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String name = br.readLine();            // Read input from STDIN
        System.out.println("Hello " + name);    // Write output to STDOUT

        //Scanner
        Scanner s = new Scanner(System.in);
        String name = s.nextLine();             // Read input from STDIN
        System.out.println("Hello " + name);    // Write output to STDOUT
    }
}
*/

// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail

import java.util.*;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import java.io.*;
import java.net.URL;

public class Solution
{
    public static void main(String args[] ) throws Exception
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String product = br.readLine();
        WebDriver driver = new RemoteWebDriver(new URL("http://127.0.0.1:9515"),DesiredCapabilities.chrome());
        driver.get("https://a2z.aspiringminds.com/selenium/YEp27CBrbrzp4a91e5BUPgl03_2FxVAv79SAMTir84jce6mzM25ImPx3cisVM1HryZT_2F5C7hnfrl0Ic9uhLeMTtr8V6d5W2re0Tl87dsHXcPY_3D/products");
        String product_price = "-1";
        List<WebElement> product_list = driver.findElements(By.className("caption"));
        l1:for(int i=0;i<product_list.size();i++)
        {
            String product_name = product_list.get(i).findElement(By.tagName("h3")).getText();
            if(product.equals(product_name)){
                    product_price = product_list.get(i).findElement(By.tag
Name("p")).getText().split("\\.")[1];
                    break l1;
            }


        }

        System.out.println(product_price);




    }
}
```

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

Total score

10/10

100%	**0%**	**0%**
Basic(**10**/10)	Advance(**0**/0)	Edge(**0**/0)

Test Cases: Deep Dive

Compilation Statistics

5	3	2	0	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	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%**

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

50 / 100

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

Functional Correctness

100 / 100

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

Final Code Submitted — Compilation Status: Pass

```
// Sample code to read input and write output:

/*
import java.util.*;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.remote.DesiredCapabilities;
```

Code Analysis

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites 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 InputStreamRea
der(System.in));
21     String name = br.readLine();          // Read input from STDIN
22     System.out.println("Hello " + name);    // Write output to STDOU
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 STDOU
T
28   }
29 }
30 */
31
32 // Warning: Printing unwanted or ill-formatted data to output will ca
use 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 InputStreamRea
der(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://12
7.0.0.1:9515"),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

```
pXCO64kCUfF4oSBDdpOIzpVufCnN_2FsA18QCJjHaRqRGT2oDdSGP9
Y1rCJFA4IrFLcYhOrcr5Roj3WcaCiDkX_2FnORqyBWEBt6x4_3D/blog");
      String text = driver.findElement(By.id("content")).getText();
      String[] words = text.toLowerCase().split(" ");

      for(int i=0;i< words.length;i++)
       {
            if(data.containsKey(words[i]))
                 data.put(words[i],data.get(words[i])+1);
            else
                 data.put(words[i],1);

       }

       Set<String> keys = data.keySet();
       l1:for(String key:keys){
             if(data.get(key)==count){
                required_key = key;
                break l1;
             }
       }
      System.out.println(required_key);

  }
}
```

Test Case Execution

Passed TC: **100%**

Total score

6/6

100%
Basic(**6**/6)

0%
Advance(**0**/0)

0%
Edge(**0**/0)

Test Cases: Deep Dive

Compilation Statistics

10	4	6	0	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	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

IP Binding	
Print Screen	0
ID Card Face Detected	No
Browser Toggle	
IP Address	
Geolocation Tag	

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.