hr | avatar
Illuminating Talent

Test Results and Interview Guide

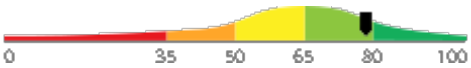
Candidate: **Elizabeth Wantsajob**
Assessment: Modern Java
Completed: July 4, 2026
Prepared for: Sara Maple
Example Company

What's Included

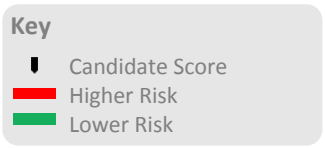
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Modern Java assessment measures key factors related to high performance and tenure in this job. Attribute types measured vary by test, but can include cognitive ability, skills, knowledge, personality characteristics, emotional intelligence, and past behavioral history. This report includes a one page summary, followed by detailed results with an embedded interview guide. Note that these results should always be used as a part of a balanced candidate selection process that includes independent evaluation steps, such as interviews and reference checks.

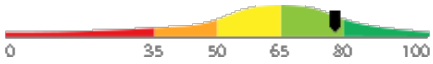
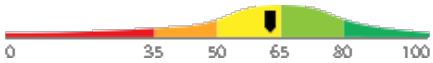
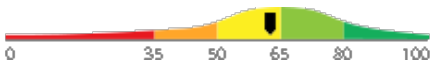
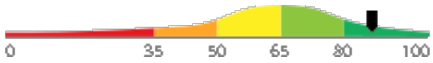
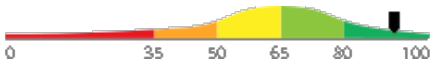
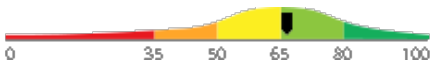
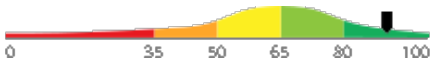
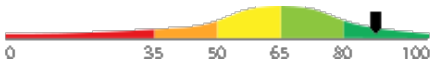
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Modern Java July 4, 2026	78	

The candidate demonstrates a solid and well-rounded knowledge of modern Java programming, including proficiency with language features, design patterns, concurrency utilities, and data processing techniques. They are likely capable of independently writing clean, maintainable business application code while applying best practices across most areas evaluated. Minor gaps in specialized or advanced topics may exist but are not expected to significantly impede performance.



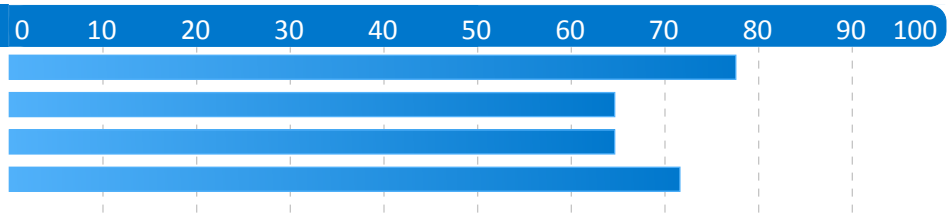
Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Exception Handling and Resource Management	78	
Lambda Expressions and Functional Interfaces (Coding Tasks)	62	
Stream API and Collectors (Coding Tasks)	62	
Lambda Expressions and Functional Interfaces	87	
Modern Java Language Features	92	
Null Safety and Optional	66	
Object-Oriented Design with Interfaces and Inheritance	90	
Stream API and Collectors	88	

Comparison

Percentile scores indicate how the candidate compares to other test-takers within various groups. The candidate scored equal to or better than the fraction of test-takers indicated by the percentile.

Test-Taker Group	Percentile
Global	78th
North America	65th
United States	65th
Example Company	72nd



Artificial Intelligence (AI) Generated Scores

This table includes one or more scores derived from a large language model AI query. AI-derived scores are non-deterministic. That is, they are not precisely repeatable. Therefore, these scores should always be treated as supplementary information and should never be used exclusively or compared to hard cutoff values.

Estimated Value	Score	Confidence	Interpretation
Knowledge, Skills, and Abilities Summary	-	-	Summary Points (AI): • (Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions. • Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles. • Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement. • Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving. • Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles. Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies. Elizabeth shows a strong aptitude in Drag and Drop Files, performing well on this technical task and suggesting she is comfortable with this type of computer interaction. This is a notable strength that would translate well into roles requiring file management and basic computer navigation tasks. In the area of Analytical Thinking and Attention to Detail, Elizabeth performs at a moderate level. She demonstrates solid ability in Recognizing and Confirming Numbers, which suggests she is careful and accurate when working with numerical data — a valuable skill in detail-oriented work environments. Her Grammar performance is adequate but leaves room for improvement, indicating she may occasionally make written communication errors. Her weakest area within this competency is Reading and Analyzing Problems, where she struggled to consistently interpret and work through written problem scenarios. This may impact her effectiveness in roles that require critical reading, written comprehension, or complex problem-solving. Elizabeth's most significant area for development is Navigating Between Screens, where she scored considerably lower than the other competencies. This suggests she may have difficulty efficiently moving through software interfaces or multi-screen workflows, which could slow productivity in roles that rely heavily on navigating computer applications or data entry systems. Overall, Elizabeth brings some useful technical strengths, particularly in file management and numerical accuracy, but would benefit from targeted development in software navigation and analytical problem-solving to be fully effective in roles that demand these skills. Computed on: April 2, 2026, 11:09:49PM EDT

Detail

Candidate:	Elizabeth Wantsajob, beth.wantsajob@gmail.com
Assessment:	Modern Java
Authorized:	July 4, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	July 4, 2026, 4:19:16PM EDT
Completed:	July 4, 2026, 4:19:16PM EDT
Overall Score:	78

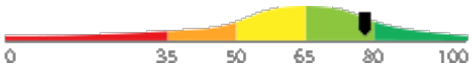
Knowledge and Skills Detail

This section contains a list of job-related knowledge areas and skills that have been evaluated. Low scores in these areas often indicate that additional learning may be required before top performance can be achieved.

Detail | Interview Guide

Exception Handling and Resource Management

Score: 78



Description:

Covers writing robust exception handling using try-catch-finally, try-with-resources for automatic resource cleanup, and creating custom exception classes. These skills are essential for writing reliable business applications that handle errors gracefully and manage resources such as file handles and database connections correctly.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and competent understanding of exception handling and resource management in Modern Java. They are generally proficient with try-catch-finally, try-with-resources for automatic resource cleanup, and creating custom exception classes, and can apply these skills effectively in most business application scenarios.

How would you design a custom exception hierarchy for a payment processing module, and how would you decide between using checked and unchecked exceptions?

1	2	3	4	5
Cannot distinguish checked from unchecked or proposes a flat, poorly structured hierarchy.		Correctly distinguishes exception types and proposes a reasonable hierarchy with some justification.		Gives a well-structured hierarchy, clearly justifies checked vs. unchecked choices, and mentions wrapping or chaining.

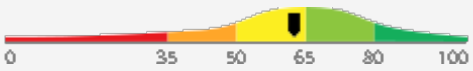
What is try-with-resources in Java and why is it preferred over a traditional try-finally block when working with resources like file streams?

1	2	3	4	5
Cannot explain the construct or does not mention automatic closing of resources.		Correctly explains automatic resource closing but may not mention the AutoCloseable interface.		Clearly explains AutoCloseable, automatic closing order, and why it reduces boilerplate and errors.

Detail

Lambda Expressions and Functional Interfaces (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

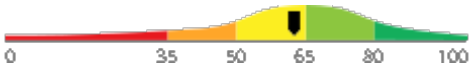
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Stream API and Collectors (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

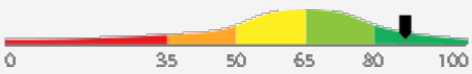
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Lambda Expressions and Functional Interfaces

Score: 87



Description:

Covers writing and using lambda expressions, method references, and built-in functional interfaces such as Predicate, Function, Consumer, and Supplier. These are the building blocks of modern Java code and are used throughout data processing, event handling, and API design in everyday business applications.

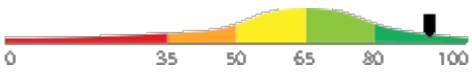
Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits a comprehensive and advanced mastery of modern Java programming techniques and patterns across all assessed areas. They demonstrate deep proficiency in functional programming, concurrency, design patterns, newer language features, file and data processing, modular code organization, and writing testable business applications. This individual is exceptionally well-prepared to take on complex Java development responsibilities and may be well-suited to serve as a technical resource or mentor for less experienced team members.

Modern Java Language Features

Score: 92



Description:

Covers practical use of features introduced since Java 14, including records for concise data classes, sealed classes for restricted type hierarchies, pattern matching for instanceof and switch expressions, text blocks, and var for local type inference. These features reduce boilerplate and improve code clarity in everyday business code.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive understanding of modern Java language features, including the practical application of records, sealed classes, pattern matching for both instanceof and switch expressions, text blocks, and local type inference. They are well-equipped to use these features proficiently to write clean, concise, and maintainable business code.

Interview Guide

How would you use the built-in functional interfaces Predicate, Function, and Consumer together to filter, transform, and process a list of employee objects in a business application?

1	2	3	4	5
Cannot describe how to combine functional interfaces or gives an incorrect approach.		Correctly describes using each interface separately but struggles to compose them together.		Fluently composes all three interfaces, possibly chaining with andThen or using method references.

Can you explain what a lambda expression is in Java and give a simple example of how you might use one in place of an anonymous class?

1	2	3	4	5
Vague or incorrect description; no working example or confuses syntax.		Correct basic definition and a simple, working example with minor gaps.		Clear explanation, correct example, and mentions functional interfaces or method references.

How would you use a sealed class combined with pattern matching in a switch expression to process different types of payment methods in a billing system?

1	2	3	4	5
Cannot describe sealed classes or write a pattern-matching switch expression.		Correctly describes sealed classes and writes a basic pattern-matching switch with minor errors.		Writes a clean, exhaustive switch expression using sealed subtypes, demonstrating type safety and clarity.

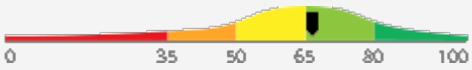
What is a Java record and in what kind of situation would you use one instead of a regular class?

1	2	3	4	5
Cannot explain what a record is or confuses it with a regular class or enum.		Correctly explains that records are immutable data carriers with auto-generated methods.		Explains records clearly, notes limitations such as immutability, and gives a practical use case like a DTO.

Detail

Null Safety and Optional

Score: 66



Description:

Covers using the Optional class to represent values that may or may not be present, avoiding null pointer exceptions, and writing safer, more expressive code. Proper use of Optional is a common expectation in modern Java codebases and directly reduces a frequent source of runtime errors in business applications.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid working knowledge of Null Safety and Optional in Modern Java. They are capable of using the Optional class effectively to represent absent values and reduce null pointer exceptions in most real-world development contexts.

Interview Guide

How would you use Optional to safely retrieve a customer's preferred shipping address from a nested object structure, and what methods would you use to provide a default address if none is found?

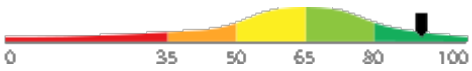
1	2	3	4	5
Resorts to null checks or cannot chain Optional methods correctly.		Uses map and orElse correctly but may not handle flatMap for nested Optionals.		Correctly chains map or flatMap with orElseGet or orElseThrow for a clean, null-safe solution.

What problem does the Optional class solve in Java, and how do you create and retrieve a value from an Optional?

1	2	3	4	5
Cannot explain the purpose or misuses Optional, such as comparing it to null directly.		Correctly explains null safety purpose and demonstrates basic creation and retrieval.		Explains purpose clearly and demonstrates map, flatMap, or orElseGet for expressive usage.

Object-Oriented Design with Interfaces and Inheritance

Score: 90



Description:

Covers applying core object-oriented principles including encapsulation, inheritance, polymorphism, and composition, as well as using interfaces with default and static methods. This underpins how business logic is structured and how design patterns such as Strategy, Factory, and Builder are implemented in Java.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive understanding of object-oriented design principles in Java, including the nuanced application of encapsulation, inheritance, polymorphism, composition, and interface design. They are highly proficient in structuring complex business logic and implementing sophisticated design patterns such as Strategy, Factory, and Builder with confidence and precision.

Interview Guide

How would you use the Strategy design pattern in Java to allow a billing system to switch between different pricing rules at runtime?

1	2	3	4	5
Cannot describe the pattern or proposes a solution that does not support runtime switching.		Describes defining a strategy interface and multiple implementations with minor design gaps.		Gives a clean design using a functional interface or strategy interface, injected at runtime, possibly with lambdas.

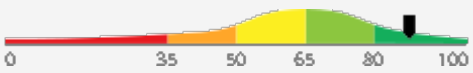
What is the difference between an abstract class and an interface in Java, and when would you choose one over the other?

1	2	3	4	5
Cannot clearly distinguish the two or gives outdated or incorrect rules.		Correctly identifies key differences but gives only a partial or generic guideline for choosing.		Clearly explains differences including default methods and gives a practical, context-driven rationale.

Detail	Interview Guide

Stream API and Collectors

Score: 88



Description:

Covers creating and using streams to filter, map, sort, and reduce collections of data, as well as using the Collectors utility to group, partition, and aggregate results. This is one of the most frequently used features in modern Java business applications for transforming and summarizing data.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive mastery of the Stream API and Collectors in Modern Java. They can confidently and efficiently leverage the full range of stream operations and Collectors utilities to transform, summarize, and aggregate complex data collections in demanding business application scenarios.

Given a list of sales transactions, how would you use the Stream API to calculate the total value of transactions grouped by product category?

1	2	3	4	5
Cannot describe a working approach or misuses stream operations.		Describes using groupingBy and summingDouble or similar, with minor syntax errors.		Writes a clean, correct solution using Collectors.groupingBy with a downstream collector.

What is a Java Stream and how is it different from a regular collection like a List?

1	2	3	4	5
Confuses streams with collections or cannot describe what a stream does.		Correctly notes streams are not storage and support pipeline operations, with minor gaps.		Clearly explains lazy evaluation, pipeline structure, and the difference from collections.

IT Coding Tasks

During the assessment, the candidate was asked to write one or more programs or scripts. Their responses are included below for review.

Question or Task	Response

Complete the provided partial C program by filling in the missing sections marked with TODO comments. Your completion must use standard C keywords and libraries.

The program declares a function called duplicate_array that:
1. Takes a const int pointer to a source array and its length as parameters.
2. Uses calloc to allocate a new int array of the same length.
3. Returns NULL if calloc fails.
4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation).
5. Returns the pointer to the newly allocated copy.

In main, the program:
1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.
2. Calls duplicate_array to create a heap-allocated copy.
3. Checks for NULL and prints an error and returns 1 if the call failed.
4. Prints each element of the duplicate using a loop.
5. Frees the duplicate array.

Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.

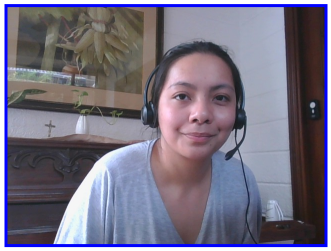
```
#include <stdio.h>
#include <stdlib.h>

int *duplicate_array(const int *src, int length) {
   /* TODO: Use calloc to allocate a new array of 'length' integers, return
      NULL if calloc fails, copy elements from src using pointer arithmetic,
      and return the new pointer. */
   calloc(303);
}

int main(void) {
   /* TODO: Declare and initialize a stack array of 4 integers: 5, 15, 25, 35,
      then call duplicate_array and store the result. Check for NULL and
      print an error message returning 1 if it failed. */
   array[4]={5,15,25,35};

   int i;

   /* Print each element of the duplicate */
   for (i = 0; i < 4; i++) {
      printf("duplicate[%d] = %d\n", i, *(duplicate + i));
   }

   /* Free the duplicate array */
   free(duplicate);
   return 0;
}
```

Question or Task	Response
	Comments (AI): The code segment has several syntax errors and incomplete implementation. The duplicate_array function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the duplicate_array function. However, the structure and intent of the code are somewhat clear, and the code attempts to follow the requirements.

Identity Confirmation Photos

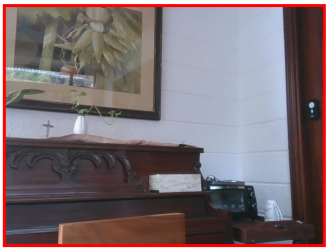
The following photos of the candidate and any identification were uploaded during the assessment session.

Photo Analysis Results	
- Risk:	Medium risk of cheating based on image inconsistencies
- Percent match among processed faces	100%
- Total images processed	17
- Total images with valid faces	14 (82%)
- Total pairs of faces compared	13
- Pairs in which faces matched	13 (100%)

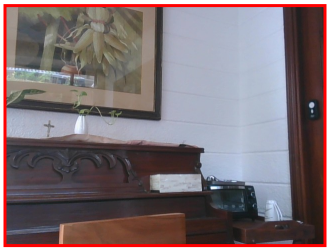


Pre/Post-Test Photo

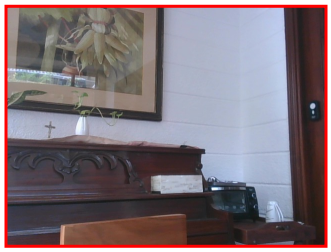
ID Photo



In-Test Error Detected (No Face Detected)

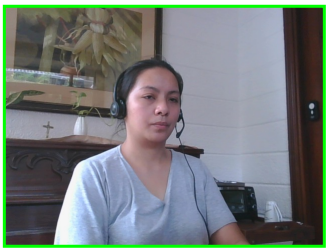


In-Test Error Detected (No Face Detected)

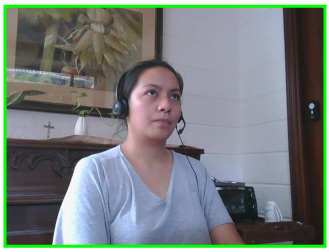


In-Test Error Detected (No Face Detected)

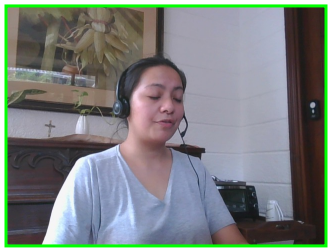
In-Test Photo



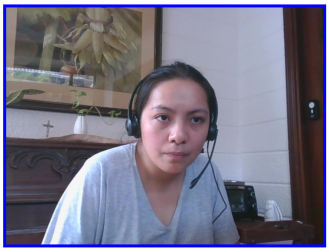
In-Test Photo



In-Test Photo



In-Test Photo



Pre/Post-Test Photo

Resume or CV

Summary Updated on

Motivated career professional with extensive experience in office administration and management. Proven track record of improving efficiency, reducing costs, and enhancing office operations through strategic initiatives and technology implementation.

Objective

I am seeking a role where I can use my many skills and my exceptional judgment and empathy for customers to make a difference to a growing company.

Education

- Associate of Applied Science in Office Administration, Portland Community College, 2020

Experience

- General Office Clerk, Paramount Office Management, 09/2023 – Present
- Administrative Assistant, Global Enterprises Inc., 04/2021 – 08/2023
- Administrative Assistant, Innovative Business Solutions Ltd., 07/2019 – 03/2021

Other Qualifications

- Microsoft Office Specialist (MOS) Certification
- Certified Administrative Professional (CAP)
- International Association of Administrative Professionals (IAAP) Certification

Report Preparation Notes

- Hiring decisions should never be based on a single source of information. The most effective use of this assessment report is as a part of a multi-faceted program of candidate evaluation that includes resume review, interviews, and reference checks.
- Overall vs Percentiles Scores: The overall score reflects the success in the test, based on the mean (average) and standard deviation of the test scores. The percentile score reflects the percentage of test-takers who scored equal or below this overall score. We recommend you use the Overall Score as your primary evaluation criteria. However, percentile scores can often be useful in comparing specific candidates against one another and with a group, such as for test takers in a certain organization or within a certain account.
- Note that comparison information is calculated based on completed instances of this assessment at that time the assessment is scored. As additional instances are completed, the comparative data may change. You can always update a report to the current values by clicking on 'Recalculate Percentiles' within the online results viewing pages at www.hravatar.com.
- Most competency scores are norm-based, which means that they can be interpreted in terms of their distance from the average or mean score. For all scales, a score equal to the mean receives a score of 65 and scores above and below this value are set so that a score change of 15 equals one standard deviation.
- For linear competencies, higher is better across the entire scale. For these scales a score between 65 and 80 (light green) represents 0 to 1 standard deviation above the mean and a score above 80 (dark green) represents more than one standard deviation above the mean. Similarly, a score of 50 - 65 (yellow) represents 0 to 1 standard deviation below the mean, while a score of 35 - 50 (orange) equates to 1 to 2 standard deviations below the mean, and a score below 35 represents more than 2 standard deviations below the mean.
- Sim ID: 20778-1, Key: 0-0, Rpt: 68, Prd: 9600, Created: 2026-07-04 16:19 EDT
- UA: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko

Score Calculation Detail

The following table provides a summary of how the overall score was calculated from each of the individual competency scores. First, all competency scores are calculated on a scale of 0-100. Note that some competencies use their color category rather than their actual numeric score in the overall calculation. For these, a standard score associated with the assigned color category is used in the overall score calculation rather than the actual numeric score. This is reflected in the ''Score Value Used'' column. Next, a weighted average of scores is computed using individual competency weights, typically set using job analysis data provided by the US Government Occupational Information Network (O*Net).

Competency	Score	How applied to overall	Score Value Used	Weight (%)
Exception Handling and Resource Management	78.4198	Numeric Score	78.4198	12.5000
Lambda Expressions and Functional Interfaces	87.1675	Numeric Score	87.1675	12.5000
Lambda Expressions and Functional Interfaces (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Modern Java Language Features	92.1569	Numeric Score	92.1569	12.5000
Null Safety and Optional	66.7021	Numeric Score	66.7021	12.5000
Object-Oriented Design with Interfaces and Inheritance	90.4830	Numeric Score	90.4830	12.5000
Stream API and Collectors	88.0812	Numeric Score	88.0812	12.5000
Stream API and Collectors (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Weighted Average:				78.6209
Final Overall Score:				78

Notes

(This area is intentionally blank - it's reserved as space for your notes.)