hr | avatar
Illuminating Talent

Test Results and Interview Guide

Candidate: **Elizabeth Wantsajob**
Assessment: Object Oriented Programming Concepts
Completed: June 27, 2026
Prepared for: Sara Maple
Example Company

What's Included

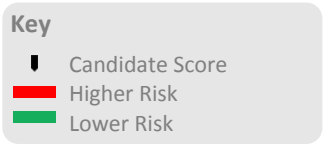
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Object Oriented Programming Concepts 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 Object Oriented Programming Concepts June 27, 2026	72	

The candidate demonstrates a solid and well-rounded understanding of object-oriented programming concepts, including class design, polymorphism, design patterns, exception handling hierarchies, and the application of SOLID principles. They are likely capable of independently designing and maintaining object-oriented programs with minimal supervision, though occasional guidance on more advanced or nuanced topics may still be beneficial.



Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Abstraction and Interfaces	69	
Classes and Objects	69	
Encapsulation and Access Modifiers	73	
Inheritance and Polymorphism	82	
Object Relationships: Composition, Aggregation, and Association	73	
SOLID Principles and Code Maintainability	66	

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	0 10 20 30 40 50 60 70 80 90 100
Global	72nd	
North America	60th	
United States	60th	
Example Company	66th	

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:	Object Oriented Programming Concepts
Authorized:	June 27, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 27, 2026, 8:10:26PM EDT
Completed:	June 27, 2026, 8:10:26PM EDT
Overall Score:	72

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

Abstraction and Interfaces

Score: 69

Description:

Understanding how to define abstract classes and interfaces to establish shared contracts between components. Includes knowing when to use an interface versus an abstract class and how these tools help separate what a component does from how it does it.

Interpretation:

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

The candidate exhibits a solid and proficient understanding of abstraction and interfaces, including when to use each construct and how they help separate a component's contract from its implementation. Minor gaps in knowledge or application may exist, but they are capable of working effectively with these concepts in most professional contexts.

How would you decide whether to use an abstract class or an interface when designing a set of related components? Walk me through your reasoning.

1	2	3	4	5
Cannot articulate a meaningful difference or applies the wrong tool to a described scenario.		Identifies key differences and makes a reasonable choice but reasoning is incomplete or partially incorrect.		Clearly articulates the tradeoffs, references shared state or default behavior vs. pure contracts, and applies sound reasoning.

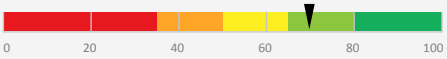
What is an interface, and how is it different from a regular class? Why might you use one?

1	2	3	4	5
Cannot define an interface or confuses it with a class or abstract class.		Correctly defines an interface and notes it cannot be instantiated but gives a limited reason for using one.		Clearly defines an interface, distinguishes it from a class, and explains its role in defining contracts between components.

Detail

Classes and Objects

Score: 69



Description:

Understanding how to define classes and create objects from them. Includes knowledge of constructors, instance variables, methods, and how objects are instantiated and used in a program.

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 object-oriented programming concepts, including inheritance, polymorphism, encapsulation, abstraction, and common design patterns. They are generally capable of applying OOP principles to design and maintain well-structured programs, with only minor gaps in more advanced areas such as complex design patterns or nuanced class relationship modeling.

Interview Guide

Walk me through how you would design a class for a real-world entity you have worked with. What fields, constructors, and methods would you include and why?

1	2	3	4	5
Describes a vague or incomplete class design with little reasoning behind design choices.		Describes a reasonable class design with some justification for fields, constructors, and methods.		Provides a well-structured class design with clear reasoning, appropriate use of constructors, and encapsulated fields.

Can you explain what a class is and how it differs from an object? Can you give a simple real-world example of each?

1	2	3	4	5
Confuses classes and objects or cannot provide a meaningful distinction or example.		Correctly distinguishes classes from objects but explanation or example lacks clarity.		Clearly explains class as a blueprint and object as an instance, with a strong, concrete example.

Detail

Encapsulation and Access Modifiers

Score: 73

Description:

Understanding how to protect the internal state of a class by restricting direct access to its fields and methods. Includes knowing how to use access modifiers such as public, private, and protected, and how to expose data safely through getters and setters.

Interpretation:

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

The candidate demonstrates a solid understanding of encapsulation principles and the use of access modifiers, including public, private, and protected. They are generally proficient in designing classes that protect internal state and expose data safely through getters and setters, with only minor gaps in knowledge.

Interview Guide

Describe a situation where poor encapsulation caused or could cause a bug or maintenance problem in a codebase. How would you fix it?

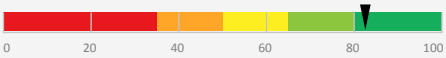
1	2	3	4	5
Cannot describe a realistic scenario or fix, or conflates encapsulation with unrelated concepts.		Describes a plausible scenario with a reasonable fix but lacks depth in the explanation.		Gives a specific, realistic scenario and a clear fix using proper access modifiers and controlled access methods.

What does it mean to make a field private in a class, and why would you do that instead of making it public?

1	2	3	4	5
Cannot explain the difference or gives an incorrect reason for using private fields.		Correctly identifies that private restricts access but gives a limited explanation of why it matters.		Clearly explains data hiding, preventing unintended modification, and the role of getters and setters.

Inheritance and Polymorphism

Score: 82



Description:

Understanding how classes can inherit fields and methods from parent classes, and how objects of different types can be treated as instances of a common parent type. Includes knowledge of method overriding and how polymorphism allows flexible, reusable code.

Interpretation:

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

The candidate demonstrates a comprehensive and advanced understanding of inheritance and polymorphism within object-oriented programming. They are well-versed in class hierarchies, method overriding, and leveraging polymorphism to write flexible, reusable, and maintainable code. This level of proficiency reflects strong readiness to apply and potentially mentor others in these concepts within a professional environment.

Can you explain the difference between method overloading and method overriding? When would you use each one?

1	2	3	4	5
Confuses the two concepts or cannot explain when each would be used.		Correctly distinguishes the two but gives limited or generic examples of when to use them.		Clearly explains both concepts with accurate examples and articulates appropriate use cases for each.

What is inheritance in object-oriented programming, and can you give an example of when you would use it?

1	2	3	4	5
Cannot define inheritance or provides an inaccurate or nonsensical example.		Correctly defines inheritance and provides a basic example but does not explain the practical benefit.		Clearly defines inheritance, gives a relevant example, and explains how it promotes code reuse.

Detail

Object Relationships: Composition, Aggregation, and Association

Score: 73

Description:

Understanding how objects relate to and depend on one another. Includes knowing the difference between composition, where one object owns another, aggregation, where objects are related but independent, and association, where objects simply interact with one another.

Interpretation:

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

The candidate exhibits a solid understanding of object relationships, including the distinctions between composition, aggregation, and association in object-oriented programming. They are likely capable of applying these concepts effectively in most practical scenarios, though some nuanced or complex cases may present challenges.

Interview Guide

Describe a class design you have worked on or would create that uses composition. Why did you choose composition over other ways of relating objects?

1	2	3	4	5
Describes a design that does not actually use composition or cannot explain the design choice.		Describes a valid use of composition but provides limited justification for the design decision.		Describes a clear, practical example of composition with strong reasoning around ownership, lifecycle, and flexibility.

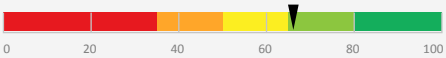
What is the difference between composition and inheritance? When would you choose one over the other?

1	2	3	4	5
Cannot distinguish composition from inheritance or gives an inaccurate explanation of either.		Correctly distinguishes the two but cannot clearly explain when to prefer one over the other.		Clearly explains both, notes that composition favors flexibility and avoids tight coupling, and gives a concrete example.

Detail

SOLID Principles and Code Maintainability

Score: 66



Description:

Understanding the five SOLID principles and how to apply them when writing and organizing object-oriented code. Includes knowing how these principles reduce tight coupling, improve cohesion, and make code easier to extend and maintain over time.

Interpretation:

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

The candidate demonstrates a solid and well-rounded understanding of SOLID principles and how to apply them when organizing object-oriented code. They are generally capable of using these principles to reduce tight coupling, improve cohesion, and write code that is easier to extend and maintain over time.

Interview Guide

Can you describe a time when you refactored or redesigned code to make it easier to maintain or extend? Which object-oriented principles guided your decisions?

1	2	3	4	5
Cannot describe a relevant refactoring experience or does not connect decisions to OOP principles.		Describes a plausible refactoring scenario and references at least one OOP principle with partial accuracy.		Describes a specific, concrete refactoring with clear connections to SOLID or other OOP principles and measurable improvements.

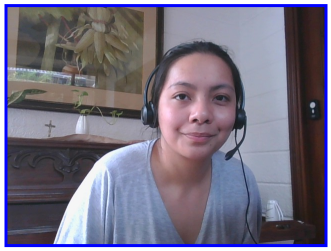
Have you heard of the SOLID principles? Can you name and briefly describe any one of them?

1	2	3	4	5
Cannot name or describe any SOLID principle accurately.		Names and describes one principle correctly but cannot connect it to practical coding decisions.		Names and accurately describes at least one principle and connects it to a real coding scenario or benefit.

Identity Confirmation Photos

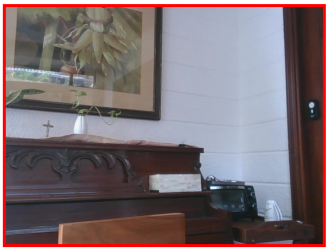
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

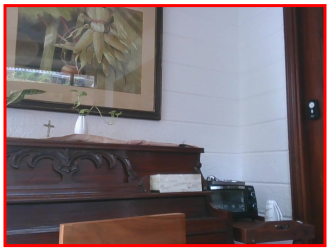


Pre/Post-Test Photo

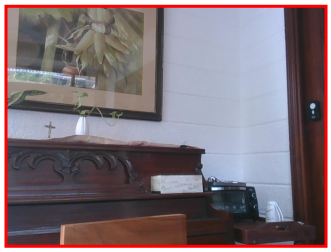
ID Photo



In-Test Error Detected (No Face Detected)

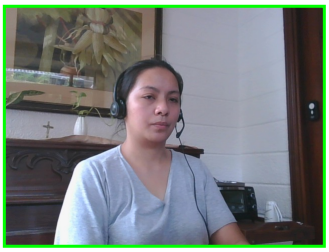


In-Test Error Detected (No Face Detected)

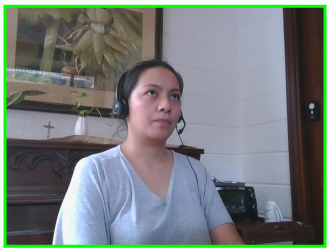


In-Test Error Detected (No Face Detected)

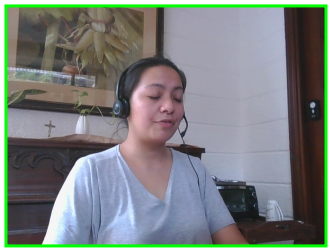
In-Test Photo



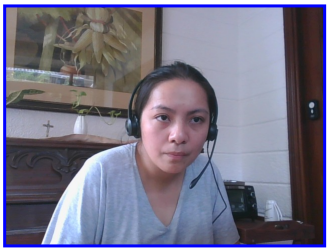
In-Test Photo



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: 20797-1, Key: 0-0, Rpt: 68, Prd: 9619, Created: 2026-06-27 20:10 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 (%)
Abstraction and Interfaces	69.1175	Numeric Score	69.1175	16.6667
Classes and Objects	69.7600	Numeric Score	69.7600	16.6667
Encapsulation and Access Modifiers	73.4521	Numeric Score	73.4521	16.6667
Inheritance and Polymorphism	82.5319	Numeric Score	82.5319	16.6667
Object Relationships: Composition, Aggregation, and Association	73.9578	Numeric Score	73.9578	16.6667
SOLID Principles and Code Maintainability	66.3476	Numeric Score	66.3476	16.6667
Weighted Average:				72.5278
Final Overall Score:				72

Notes

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