The following applies to all exams: Once exam vouchers are purchased you have up to one year from the date of purchase to use it. Each voucher is valid for one exam and may only be used at an Authorized Prometric Testing Center in the country for which it was purchased. Please be aware that exam vouchers are nonrefundable for any reason.

The Sun Certified Associate for the Java Platform, Standard Edition, Exam Version 1.0

This certification exam provides an ideal entry into an application development or a software project management career using Java technologies.

This worldwide credential validates basic knowledge of Object-Oriented Concepts, UML representation of OO concepts, the Java programming language, and general knowledge of Java Platforms and Technologies.

Candidates for this exam include: entry level Java programmers, students studying to become Java programmers, project or program managers working with Java technology in the software development industry.

The Sun Certified Programmer for Java 2 Platform 5.0

This certification exam is for programmers experienced using the Java programming language. Achieving this certification provides clear evidence that a programmer understands the basic syntax and structure of the Java programming language and can create Java technology applications that run on server and desktop systems using J2SE 5.0.

Sun Certified Developer for the Java 2 Platform, Standard Edition

This certification is in two parts.

Part 1 - Developer Assignment

The assignment requires that you write working code for a small but plausible business system.

To keep the amount of work involved to a reasonable level, the programs you create will be much more restricted in capability, and much cruder in overall presentation, than anything you would actually create for a paying customer. However, the essence of the problem will be the same.

You will be graded on correctly solving the technical requirements, not on the "polish" of the finished product. Note however, that some aspects of ease of use, for example how easily the program may be started or configured, will be part of the scoring criteria. Pay careful attention to any such requirements and be sure to adhere to them.

Part 2 – certification exam

The certification exam is for programmers who are already familiar with the basic structure and syntax of the Java programming language, and who have a need to further apply this knowledge

to developing complex, production-level applications. Certification is available for the Java 2 platform.

Sun Certified Web Component Developer

The Sun Certified Web Component Developer for J2EE Platform certification exam is for Sun Certified Programmers (any edition) who are using the Java technology servlet and JavaServer Pages (JSP) application program interface (APIs) to develop Web applications.

Sun Certified Business Component Developer

The Sun Certified Business Component Developer for the Java 2 Platform, Enterprise Edition 1.3 exam is for programmers and developers who are responsible for designing, developing, testing, deploying, and integrating Enterprise JavaBeans (EJB) applications.

It is also for those specializing in leveraging the Java 2 Platform, Enterprise Edition (J2EE platform) technologies used to develop server-side components that encapsulate the business logic of an application.

Prior to beginning the Sun Certified Business Component Developer program, you must be a Sun Certified Programmer for the Java platform (any edition).

Sun Certified Developer for Java Web Services

The Sun Certified Developer for Java Web Services certification exam is for developers who have been creating web services applications using Java technology components such as those supported by the Java Web Services Developer Pack and the Java 2, Enterprise Edition 1.4 platform.

Passing this exam certifies that the candidate has achieved a standard level of proficiency with web services, as well as with the Java technologies that support web services.

Sun Certified Enterprise Architect

This exam is in 3 parts.

- Part 1 is multiple choice and tests your knowledge in the areas of general architecture and J2EE.
- Part 2 is an assignment that tests your ability to apply the knowledge you were tested on in part 1.
- Part 3 is an essay that asks questions about your assignment.

Scoring Criteria for the Architect Exam

Your project will be evaluated on a large number of objective criteria that fall into three categories:

1) Class Diagram: This category covers how well your class diagram(s) address the object model needed to satisfy the requirements.
2) Component Diagram: This category covers how well your component diagram(s) convey the structure of the architecture in satisfying the requirements.
3) Sequence/Collaboration Diagrams: This category covers how well your sequence or collaboration diagrams satisfy the requirements of the assignment.

Additionally, each category is evaluated on UML compliance. The maximum number of possible points is 100. The minimum passing grade is 70.

Sun Certified Mobile Application Developer

The Sun Certified Mobile Application Developer for the Java 2 Platform, Micro Edition, Version 1.0 certification exam is for programmers and developers who are using Java 2 Platform, Micro Edition (J2ME) technologies to develop mobile applications for cell phones or other Java enabled devices.

Passing this exam certifies that the candidate has achieved a standard level of proficiency with mobile Java technologies, as well as with the Java Technology for Wireless Industry (JTWI) specification, including the Wireless Messaging application programming interface (API) and Mobile Media APIs.

SJCP 4.0 Exam Objectives

Section 1: Declarations and Access Control

- Write code that declares, constructs and initializes arrays of any base type using any of the permitted forms both for declaration and for initialization.
- Declare classes, nested classes, methods, instance variables, static variables and automatic (method local) variables making appropriate use of all permitted modifiers (such as public, final, static, abstract, etc.). State the significance of each of these modifiers both singly and in combination and state the effect of package relationships on declared items qualified by these modifiers.
- For a given class, determine if a default constructor will be created and if so state the prototype of that constructor.
- Identify legal return types for any method given the declarations of all related methods in this or parent classes.

Section 2: Flow control, Assertions, and Exception Handling

- Write code using if and switch statements and identify legal argument types for these statements.
- Write code using all forms of loops including labeled and unlabeled, use of break and continue, and state the values taken by loop counter variables during and after loop execution.
- Write code that makes proper use of exceptions and exception handling clauses (try, catch, finally) and declares methods and overriding methods that throw exceptions.
- Recognize the effect of an exception arising at a specified point in a code fragment. Note: The exception may be a runtime exception, a checked exception, or an error (the code may include try, catch, or finally clauses in any legitimate combination).
- Write code that makes proper use of assertions, and distinguish appropriate from inappropriate uses of assertions.
- Identify correct statements about the assertion mechanism.

Section 3: Garbage Collection

- State the behavior that is guaranteed by the garbage collection system.
- Write code that explicitly makes objects eligible for garbage collection.
- Recognize the point in a piece of source code at which an object becomes eligible for garbage collection.

Section 4: Language Fundamentals

- Identify correctly constructed package declarations, import statements, class declarations (of all forms including inner classes) interface declarations, method declarations

(including the main method that is used to start execution of a class), variable declarations, and identifiers.

- Identify classes that correctly implement an interface where that interface is either java.lang.Runnable or a fully specified interface in the question.
- State the correspondence between index values in the argument array passed to a main method and command line arguments.
- Identify all Java programming language keywords. Note: There will not be any questions regarding esoteric distinctions between keywords and manifest constants.
- State the effect of using a variable or array element of any kind when no explicit assignment has been made to it.
- State the range of all primitive formats, data types and declare literal values for String and all primitive types using all permitted formats bases and representations.

Section 5: Operators and Assignments

- Determine the result of applying any operator (including assignment operators and instance of) to operands of any type class scope or accessibility or any combination of these.
- Determine the result of applying the boolean equals (Object) method to objects of any combination of the classes java.lang.String, java.lang.Boolean and java.lang.Object.
- In an expression involving the operators &, |, &&, || and variables of known values state which operands are evaluated and the value of the expression.
- Determine the effect upon objects and primitive values of passing variables into methods and performing assignments or other modifying operations in that method.

Section 6: Overloading, Overriding, Runtime Type and Object Orientation

- State the benefits of encapsulation in object oriented design and write code that implements tightly encapsulated classes and the relationships "is a" and "has a".
- Write code to invoke overridden or overloaded methods and parental or overloaded constructors; and describe the effect of invoking these methods.
- Write code to construct instances of any concrete class including normal top level classes and nested classes.

Section 7: Threads

- Write code to define, instantiate and start new threads using both java.lang.Thread and java.lang.Runnable.
- Recognize conditions that might prevent a thread from executing.
- Write code using synchronized wait, notify and notifyAll to protect against concurrent access problems and to communicate between threads.
- Define the interaction among threads and object locks when executing synchronized wait, notify or notifyAll.

Section 8: Fundamental Classes in the java.lang Package

- Write code using the following methods of the java.lang.Math class: abs, ceil, floor, max, min, random, round, sin, cos, tan, sqrt.
- Describe the significance of the immutability of String objects.
- Describe the significance of wrapper classes, including making appropriate selections in the wrapper classes to suit specified behavior requirements, stating the result of executing a fragment of code that includes an instance of one of the wrapper classes, and writing code using the following methods of the wrapper classes (e.g., Integer, Double, etc.):
    - o doubleValue
    - o floatValue
    - o intValue
    - o longValue
    - o parseXxx
    - o getXxx
    - o toString
    - o toHexString

Section 9: The Collections Framework

- Make appropriate selection of collection classes/interfaces to suit specified behavior requirements.
- Distinguish between correct and incorrect implementations of hashcode methods.

SJCP 5.0 Exam Objectives

Section 1: Declarations, Initialization and Scoping

- Develop code that declares classes (including abstract and all forms of nested classes), interfaces, and enums, and includes the appropriate use of package and import statements (including static imports).
- Develop code that declares an interface. Develop code that implements or extends one or more interfaces. Develop code that declares an abstract class. Develop code that extends an abstract class.
- Develop code that declares, initializes, and uses primitives, arrays, enums, and objects as static, instance, and local variables. Also, use legal identifiers for variable names.
- Develop code that declares both static and non-static methods, and - if appropriate - use method names that adhere to the JavaBeans naming standards. Also develop code that declares and uses a variable-length argument list.

- Given a code example, determine if a method is correctly overriding or overloading another method, and identify legal return values (including covariant returns), for the method.
- Given a set of classes and superclasses, develop constructors for one or more of the classes. Given a class declaration, determine if a default constructor will be created, and if so, determine the behavior of that constructor. Given a nested or non-nested class listing, write code to instantiate the class.

Section 2: Flow Control

- Develop code that implements an if or switch statement; and identify legal argument types for these statements.
- Develop code that implements all forms of loops and iterators, including the use of for, the enhanced for loop (for-each), do, while, labels, break, and continue; and explain the values taken by loop counter variables during and after loop execution.
- Develop code that makes use of assertions, and distinguish appropriate from inappropriate uses of assertions.
- Develop code that makes use of exceptions and exception handling clauses (try, catch, finally), and declares methods and overriding methods that throw exceptions.
- Recognize the effect of an exception arising at a specified point in a code fragment. Note that the exception may be a runtime exception, a checked exception, or an error.
- Recognize situations that will result in any of the following being thrown: ArrayIndexOutOfBoundsException,ClassCastException, IllegalArgumentException, IllegalStateException, NullPointerException, NumberFormatException, AssertionError, ExceptionInInitializerError, StackOverflowError or NoClassDefFoundError. Understand which of these are thrown by the virtual machine and recognize situations in which others should be thrown programatically.

Section 3: API Contents

- Develop code that uses the primitive wrapper classes (such as Boolean, Character, Double, Integer, etc.), and/or autoboxing & unboxing. Discuss the differences between the String, StringBuilder, and StringBuffer classes.
- Given a scenario involving navigating file systems, reading from files, or writing to files, develop the correct solution using the following classes (sometimes in combination), from java.io: BufferedReader,BufferedWriter, File, FileReader, FileWriter and PrintWriter.
- Develop code that serializes and/or de-serializes objects using the following APIs from java.io: DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream and Serializable.
- Use standard J2SE APIs in the java.text package to correctly format or parse dates, numbers, and currency values for a specific locale; and, given a scenario, determine the appropriate methods to use if you want to use the default locale or a specific locale. Describe the purpose and use of the java.util.Locale class.

- Write code that uses standard J2SE APIs in the java.util and java.util.regex packages to format or parse strings or streams. For strings, write code that uses the Pattern and Matcher classes and the String.split method. Recognize and use regular expression patterns for matching (limited to: . (dot), * (star), + (plus), ?, \d, \s, \w, [], ()). The use of *, +, and ? will be limited to greedy quantifiers, and the parenthesis operator will only be used as a grouping mechanism, not for capturing content during matching. For streams, write code using the Formatter and Scanner classes and the PrintWriter.format/printf methods. Recognize and use formatting parameters (limited to: %b, %c, %d, %f, %s) in format strings.

Section 4: Concurrency

- Write code to define, instantiate, and start new threads using both java.lang.Thread and java.lang.Runnable.
- Recognize the states in which a thread can exist, and identify ways in which a thread can transition from one state to another.
- Given a scenario, write code that makes appropriate use of object locking to protect static or instance variables from concurrent access problems.
- Given a scenario, write code that makes appropriate use of wait, notify, or notifyAll.

Section 5: OO Concepts

- Develop code that implements tight encapsulation, loose coupling, and high cohesion in classes, and describe the benefits.
- Given a scenario, develop code that demonstrates the use of polymorphism. Further, determine when casting will be necessary and recognize compiler vs. runtime errors related to object reference casting.
- Explain the effect of modifiers on inheritance with respect to constructors, instance or static variables, and instance or static methods.
- Given a scenario, develop code that declares and/or invokes overridden or overloaded methods and code that declares and/or invokes superclass, overridden, or overloaded constructors.
- Develop code that implements "is-a" and/or "has-a" relationships.

Section 6: Collections / Generics

- Given a design scenario, determine which collection classes and/or interfaces should be used to properly implement that design, including the use of the Comparable interface.
- Distinguish between correct and incorrect overrides of corresponding hashCode and equals methods, and explain the difference between == and the equals method.
- Write code that uses the generic versions of the Collections API, in particular, the Set, List, and Map interfaces and implementation classes. Recognize the limitations of the non-generic Collections API and how to refactor code to use the generic versions.

- Develop code that makes proper use of type parameters in class/interface declarations, instance variables, method arguments, and return types; and write generic methods or methods that make use of wildcard types and understand the similarities and differences between these two approaches.
- Use capabilities in the java.util package to write code to manipulate a list by sorting, performing a binary search, or converting the list to an array. Use capabilities in the java.util package to write code to manipulate an array by sorting, performing a binary search, or converting the array to a list. Use the java.util.Comparator and java.lang.Comparable interfaces to affect the sorting of lists and arrays. Furthermore, recognize the effect of the "natural ordering" of primitive wrapper classes and java.lang.String on sorting.

Section 7: Fundamentals

- Given a code example and a scenario, write code that uses the appropriate access modifiers, package declarations, and import statements to interact with (through access or inheritance) the code in the example.
- Given an example of a class and a command-line, determine the expected runtime behavior.
- Determine the effect upon object references and primitive values when they are passed into methods that perform assignments or other modifying operations on the parameters.
- Given a code example, recognize the point at which an object becomes eligible for garbage collection, and determine what is and is not guaranteed by the garbage collection system. Recognize the behaviors of System.gc and finalization.
- Given the fully-qualified name of a class that is deployed inside and/or outside a JAR file, construct the appropriate directory structure for that class. Given a code example and a classpath, determine whether the classpath will allow the code to compile successfully.
- Write code that correctly applies the appropriate operators including assignment operators (limited to: =, +=, -=), arithmetic operators (limited to: +, -, *, /, %, ++, --), relational operators (limited to: <, <=, >, >=, ==, !=), the instanceof operator, logical operators (limited to: &, |, ^, !, &&, ||), and the conditional operator (? :), to produce a desired result. Write code that determines the equality of two objects or two primitives.