

JAVA FUNDAMENTALS AND JAVA PROGRAMMING

COURSE CODE: 5058

COURSE DESCRIPTION: This Oracle designed curriculum introduces fundamental programming concepts and terminology in an engaging manner through the creation of simple animations and interactive games using Object Oriented Programming environments. While mastering basic programming constructs using OPP, students will learn basic Java syntax. Using a popular, industry recognized Java IDE and the Java programming language, students will write, edit, compile, deploy, and debug Java programs. Java classes, arrays, stacks, strings, and the core APIs that are used to design object-oriented applications will be covered. The GridWorld case study is closely examined and used to enhance student knowledge of core Java concepts. The AP Java subset is also addressed. As in other Oracle courses, collaboration and problem solving are emphasized throughout the course.

OBJECTIVE: Given the necessary equipment, supplies, and facilities, the student will complete all of the following core standards successfully.

RECOMMENDED GRADE LEVELS: 11-12

COURSE CREDIT: 1 Carnegie unit

PREREQUISITE: Algebra 2

COMPUTER REQUIREMENT: One computer per student; Internet access; see Oracle Academy specifications for additional information

REQUIRED SOFTWARE: Internet Explorer, Alice 3, Greenfoot, Eclipse (a Java IDE), and the latest JRE (Java Runtime Environment); see Oracle Academy specifications for current versions, installation instructions, and any changes to required software list. All required software should be available as a free download.

RECOMMENDED SOFTWARE: Adobe Acrobat Professional; Mozilla Firefox; Google Chrome

RESOURCES: Oracle iLearning online curriculum provided by Oracle Academy

INDUSTRY CREDENTIALS/CERTIFICATIONS AVAILABLE:

The Oracle Certified Associate Java SE7 Programmer certification is available by passing the Oracle Java SE7 Programmer 1 Exam 1Z0-803 taken at a certified testing facility.

Discount coupons for exams may be available from Oracle Academy. Third party, supplemental practice exams and materials are recommended and may be available at discounted prices.

Certification requirements and pathways are subject to change and should be confirmed prior to exam registration.

COLLEGE BOARD AP COMPUTER SCIENCE A EXAM:

Students who successfully master all requirements of this course should be prepared to take the College Board Advanced Placement Computer Science A Exam.

A. SAFETY AND ETHICS

1. Identify major causes of work-related accidents in offices.
2. Describe the threat of viruses to a computer network, methods of avoiding attacks, and options in dealing with virus attacks.
3. Identify potential abuse and unethical uses of computers and networks.
4. Explain the consequences of illegal, social, and unethical uses of information technologies, e.g., piracy; illegal downloading; licensing infringement; and inappropriate uses of software, hardware, mobile devices.
5. Differentiate between freeware, shareware, and public domain software copyrights.
6. Discuss computer crimes, terms of use, and legal issues such as copyright laws, fair use laws, and ethics pertaining to scanned and downloaded clip art images, photographs, documents, video, recorded sounds and music, trademarks, and other elements for use in Web publications.
7. Identify netiquette including the use of email, social networking, blogs, texting, and chatting.
8. Describe ethical practices in business professions such as safeguarding the confidentiality of business-related information.
9. Discuss the importance of cyber safety and the impact of cyber bullying.

B. EMPLOYABILITY SKILLS

1. Identify positive work practices, e.g., appropriate dress code for the workplace, personal grooming, punctuality, time management, organization.
2. Demonstrate positive interpersonal skills, e.g., communication, respect, teamwork.

C. STUDENT ORGANIZATIONS

1. Explain how related student organizations are integral parts of career and technology education courses.
2. Explain the goals and objectives of related student organizations.
3. List opportunities available to students through participation in related student organization conferences/competitions, community service, philanthropy, and other activities.

4. Explain how participation in career and technology education student organizations can promote lifelong responsibility for community service and professional development.

JAVA FUNDAMENTALS

D. INTRODUCTION TO PROGRAMMING

1. Identify the software and accounts used in this course.
2. Describe the purpose for using intermediary tools (Alice 3, Greenfoot, etc.) to learn Java.
3. Describe the characteristics and importance of teamwork.
4. Describe the skills used to generate an animation.
5. Describe the components of a team project.
6. Create a teamwork assessment rubric.
7. Describe the purpose for creating a journal to document programming projects.
8. Describe the code of ethics and cyber security issues required by this course.

E. OBJECT-ORIENTED PROGRAMMING

Using Alice 3 or an equivalent intermediary programming environment:

1. Compare the definitions for animation and scenario.
2. Write an example of using four problem-solving steps to storyboard an animation.
3. Create visual and textual storyboards for an animation.
4. Flowchart a storyboard.
5. Describe an algorithm.
6. Identify all components of a scene for a given scenario.
7. Define a gallery using Java programming terminology.
8. Define classes and instances.
9. Describe three-dimensional positioning axes.
10. Use a procedural method to precisely position an object in a scene.
11. Describe the value of saving multiple versions of an animation scene.
12. Correlate storyboard statements with program execution tasks.
13. Describe actor orientation in 3D modeling.
14. Create programming comments.
15. Use procedures to move objects in a scene.
16. Write Java programming procedures.
17. Demonstrate how procedure values can be altered.
18. Define multiple control statements to control animation timing.
19. Recognize programming constructs to invoke simultaneous movement.
20. Explain the purpose of variables in programming.
21. Create functions to control movement based on a return value.
22. Define the value of a variable based on a math calculation.
23. Create control structures to effect execution of instructions.
24. Create an expression to perform a math operation.

25. Use keyboard controls to manipulate an animation.
26. Create a conditional loop for repetitive behavior.
27. Use random numbers to randomize motion.
28. Complete a simple animation.
29. Describe Java simple types.
30. Define arithmetic, relational, logical, and assignment operators.
31. Describe a method, class, and instance.
32. Identify when to use IF control structures.
33. Identify when to use WHILE control structures.
34. Identify the syntax for methods, classes, functions, and procedures.
35. Describe Java input and output.

F. JAVA SYNTAX

Using Greenfoot or an equivalent intermediary programming environment:

1. Describe the components of the programming environment.
2. Create an instance of a class.
3. Describe classes and subclasses.
4. Recognize Java syntax used to create a subclass.
5. Define parameters.
6. Define how parameters are used in methods.
7. Describe how inheritance is impacted by subclass and superclass.
8. Describe properties of an object.
9. Describe the purpose of a variable.
10. Demonstrate syntax to invoke methods.
11. Demonstrate syntax for an IF decision statement.
12. Describe a procedure to display object documentation.
13. Demonstrate program-testing strategies.
14. Describe phases for developing a software application.
15. Create randomized behaviors.
16. Define comparison operators.
17. Create IF ELSE control statements.
18. Describe dot notation.
19. Describe effective placement of methods in a super or subclass.
20. Create defined methods.
21. Create code to call defined methods.
22. Write programming statements to include sound.
23. Write programming statements to include keyboard input.
24. Construct a world object using a constructor method.
25. Write programming statements to use the NEW keyword.
26. Describe the purpose of a defined variable.
27. Identify the syntax to create a defined variable.
28. Identify the syntax to test variables.
29. Write programming statements to switch between two images.
30. Write programming statements to end a game.

31. Define abstraction.
32. Identify when abstraction is used.
33. Create a WHILE loop in a constructor to build a world.
34. Describe an infinite loop.
35. Describe how to prevent infinite loops.
36. Use an array to store multiple variables used to create a world.
37. Create an expression using logic operators.
38. Describe the scope of a local variable within a method.
39. Use string variables to concatenate strings.
40. Identify Java constructs (classes, methods, constructors, etc.).
41. Correct errors in Java syntax.
42. List programming tasks to design and create a game using Java.
43. List five variations to include in quality-assurance testing.

G. USING AN INTEGRATED DEVELOPMENT ENVIRONMENT

1. Identify components of Eclipse.
2. Install Eclipse.
3. Configure Eclipse.
4. Compile a sample application.
5. Enter the code for a simple program (GalToLit).
6. Test the program.
7. Modify the program.
8. Describe the general form of a Java program.
9. Use variables in a Java program.
10. Use the IF statement in a Java program.
11. Use the FOR statement in a Java program.
12. Specify literals for simple data types and strings.
13. Demonstrate multiple ways to initialize variables.
14. Describe the scope rules of a method.
15. Identify type conversion in expressions.
16. Apply casting in Java code.
17. Use arithmetic operators.
18. Use relational operators.
19. Use logical operators.
20. Use the assignment operator.
21. Use a Math method in a simple program.
22. Identify elements of the Java API.
23. Identify common Java classes.
24. Identify common Java methods.

H. CONTROL STATEMENTS, CLASSES, OBJECTS, AND METHODS

1. Create a WHILE loop.
2. Create a DO-WHILE loop.
3. Write code to accept keyboard input during program execution.
4. Write code using the full form of the IF statement.
5. Write code using the full form of the FOR statement.
6. Use switch syntax in a control statement.
7. Use break syntax in a control statement.
8. Identify the general form of a class.
9. Create an object of a class.
10. Describe object references.
11. Create a variety of methods.
12. Write code that returns a value from a method.
13. Use parameters in a method.
14. Add a constructor to a class.
15. Describe garbage collection.
16. Describe finalizers.
17. Write code using the THIS reference.
18. Modify an existing constructor.

I. ARRAYS AND STRINGS

1. Write a single-dimensional array using primitive data types.
2. Write a single-dimensional array using reference types.
3. Write a 2-dimensional array using primitive data types.
4. Write a 2-dimensional array using reference types.
5. Declare an array.
6. Initialize an array.
7. Traverse an array.
8. Describe array initialization.
9. Use command-line arguments.
10. Write a Java program to store integers in an array.
11. Write code to perform a Bubble Sort of integers.
12. Use alternative array declaration syntax.
13. Instantiate a String.
14. Create a reference to a String.
15. Use the + and += operators for concatenating Strings.
16. Identify escape sequences in String literals.
17. Explain the difference between a String and a primitive char data type.
18. Write code to test Strings using the compareTo method.
19. Write code to test Strings using the equals method.
20. Describe why the == operator does not always work with Strings.
21. Write code to use String methods including length, substring, indexOf, and charAt.
22. Distinguish between the String method length() and an array's length value.

23. Create a program that will take in five Strings and store them in an array of Strings, sort the array in alphabetical order, and display the Strings back to the user.
24. Write code to use the Ternary operator.
25. Use the appletviewer executable to run an applet.
26. Describe different types of programming errors.
27. Describe the exceptions used for in Java.
28. Identify the exceptions thrown by any foundation class.
29. Write code to handle an exception thrown by a method of a foundation class.

J. RECURSION, ABSTRACTION, AND INHERITANCE

1. Use public access specifiers.
2. Use private access specifiers.
3. Pass objects to methods.
4. Return objects from methods.
5. Write overloaded methods.
6. Use variable argument methods.
7. Write overloaded constructors.
8. Create a class with specified arrays, constructors, and methods.
9. Create static variables.
10. Use static variables.
11. Create static methods.
12. Use static methods.
13. Create static classes.
14. Use static classes.
15. Create linear recursive methods.
16. Create non-linear recursive methods.
17. Explain UML (Unified Modeling Language) class diagrams.
18. Use the extends keyword to inherit a class.
19. Explain the differences between superclass and subclass.
20. Explain how inheritance affects member access.
21. Use super to call a superclass constructor.
22. User super to access superclass members.
23. Create a multilevel class hierarchy.
24. Identify when constructors are called in a class hierarchy.
25. Apply superclass references to subclass objects.
26. Demonstrate the concept of inheritance through the use of applets.
27. Modify an existing applet to change its parameters.
28. Write code to override methods.
29. Use dynamic method dispatch to support polymorphism.
30. Create abstract methods.
31. Create abstract classes.
32. Write code to override methods from the object class.
33. Demonstrate the use of final.

34. Explain the purpose of the Object class.
35. Write code for an applet that displays two triangles of different colors.

JAVA PROGRAMMING

K. JAVA APPLICATION DEPLOYMENT

1. Describe the concept of packages.
2. Describe how to deploy an application.
3. Describe a Java application that includes a database back end.
4. Identify the essential components of creating an application.
5. Identify multiple-tier architectures.

L. GRIDWORLD CASE STUDY

1. Install the GridWorld environment.
2. Test the GridWorld program.
3. Observe the GridWorld program results.
4. Modify the GridWorld program to create bug variations.
5. Create code to use GridWorld classes.
6. Create code to use GridWorld interfaces.
7. Create code to interact with objects.
8. Create code to use grid data structures.
9. Test the modified GridWorld program.
10. Observe the modified GridWorld program results.

M. CLASS DESIGN, GENERICS, STRINGS, AND EXCEPTIONS

1. Create Java subclasses.
2. Use subclassing to extend another class.
3. Identify class design considerations.
4. Create variable argument methods.
5. Model business problems using Java classes.
6. Explain the concept of immutability.
7. Create immutable classes.
8. Compare default and public access levels.
9. Use the instanceof operator to compare object types.
10. Use virtual method invocation.
11. Create code to demonstrate upward casts.
12. Create code to demonstrate downward casts.
13. Create a custom generic class.
14. Use the concept of generics to create type-safe data structures.
15. Use the type inference diamond to create an object.
16. Create a collection without using generics.
17. Create a collection by using generics.
18. Implement an ArrayList.

19. Implement a Set.
20. Implement a HashMap.
21. Implement a stack by using a deque.
22. Demonstrate the use of enumerated types.
23. Create code to read, search, and parse strings.
24. Use StringBuilder to create strings.
25. Use regular expressions to search, parse, and replace strings.
26. Use exception handling syntax to create reliable applications.
27. Identify common exception classes and categories.
28. Create custom exception and auto-closeable resources.
29. Test invariants by using assertions.
30. Describe the basics of input and output.
31. Read data from the console.
32. Write data to the console.
33. Use streams to read and write files.
34. Read and write objects by using serialization.

N. CAPSTONE APPLICATION PROJECT

1. Develop a final project that demonstrates the software development elements including analyze, design, develop, implement, test, and document.
2. Present the final project.

OPTIONAL OR SELF STUDY UNITS

Career Planning – see Oracle Java Fundamentals Appendix A

Introduction to SQL – see Oracle Java Programming (JP) Appendix B

Preparing for AP Computer Science A Exam – see Oracle (JP) Appendix C