



Republic of the Philippines

Department of Education

DepEd Complex, Meralco Avenue, Pasig City

STRENGTHENED SENIOR HIGH SCHOOL CURRICULUM
COMPUTER PROGRAMMING (JAVA)
Grade 11/12

Course Description:

This course enables learners to competently develop, test, and maintain software applications using Java. It enhances their career opportunities in the tech industry and allows businesses to build robust and scalable solutions. This enables them to confidently pursue roles like Java programming support staff, software developers, application developers, and user interface developers. Upon completion, learners are eligible to take assessments to earn National Certification level III in Programming (JAVA), higher education, and careers in the computer programming sector industry.

Elective: Technical Professional

Prerequisite: None

Time Allotment: In Grade 11, 320 hours for two semesters, 8 hours per week/In Grade 12, 320 hours for one semester, 16 hours per week

Schedule: First/Second Semester

QUARTER 1

CONTENT STANDARD	The learners demonstrate an understanding of Java programming, including basic structure, syntax, data types, variables, and comments. They also develop skills in creating algorithms, flowcharts, and pseudocode to plan and visualize their code before implementation.
PERFORMANCE STANDARD	The learners perform procedures in setting up the Integrated Development Environment (IDE) and in creating algorithms, flowcharts, and pseudocode to plan and visualize their code.
LEARNING COMPETENCIES	CONTENT
1. Discuss the fundamental concepts of Java	Fundamental Concepts of Java <ul style="list-style-type: none"> • History <ul style="list-style-type: none"> ○ Creation of the Oak Language and renaming from Oak to Java ○ Key contributors (James Gosling and team) ○ Initial goals of Java development ○ Sun microsystems and its role • Contributions of Java <ul style="list-style-type: none"> ○ Java Virtual Machine (JVM) and its significance ○ Impact of Java on other programming languages ○ Java's role in the rise of object-oriented programming ○ Java's place in the current programming landscape

	<ul style="list-style-type: none"> • Business and Career Opportunities • Features of Java <ul style="list-style-type: none"> ○ platform independence ○ object-oriented programming ○ security ○ multithreading
2. Perform the steps in setting up the development environment	Setting Up the Development Environment <ul style="list-style-type: none"> • Java Development Kit (JDK) • Integrated Development Environment (IDE) • Java Development Tools (JDT)
3. Discuss Java basic structures and syntax	Java Basic Structure and Syntax <ul style="list-style-type: none"> • Structure • Compiling and running the program • Class declaration • Main method • Data types • Variables • Comments
4. Develop algorithms for designing a program or system	The Algorithm Journey: From Basics to Advanced Techniques <ul style="list-style-type: none"> • Introduction to algorithms • Types of algorithms • Algorithm design techniques • Algorithm analysis • Practical examples
5. Develop flowcharts for designing a program or system	Flowcharts: A Comprehensive Guide from Basics to Advanced Applications <ul style="list-style-type: none"> • Basics of flowcharts • Flowchart symbols • Creating flowcharts • Interpreting flowcharts • Examples and exercises
6. Develop pseudocodes for designing a program or system	Pseudocode: Bridging the Gap from Concept to Code <ul style="list-style-type: none"> • Introduction to pseudocode • Writing pseudocode

- Converting pseudocode to code
- Debugging pseudocode
- Practical examples

QUARTER 2

CONTENT STANDARD	The learners demonstrate an understanding of Java Operators, Input/Output (I/O) classes, flow control structures, and arrays to solve real-world problems.
PERFORMANCE STANDARD	The learners create a well-structured program using Java Operators, input and output classes, flow control structures, and arrays.
LEARNING COMPETENCIES	CONTENT
1. Apply operators in writing a Java program	Java Operators <ul style="list-style-type: none"> • Arithmetic operators • Relational operators • Logical operators • Increment and decrement operators • Assignment operators
2. Apply Java I/O classes to read and write data files in Java programs	Java I/O Classes <ul style="list-style-type: none"> • Scanner • PrintStream
3. Apply Java Flow Control Structures in writing a Java program	Java Flow Control Structures <ul style="list-style-type: none"> • Java Conditional Statements <ul style="list-style-type: none"> ○ if ○ if then else ○ else if ○ switch case • Java Looping Statements <ul style="list-style-type: none"> ○ for loop ○ while loop ○ do-while loop ○ nested loop ○ break, continue, and return statements
4. Apply Java Arrays in writing a Java program	Arrays

	<ul style="list-style-type: none"> • Single dimensional array • Two or multi-dimensional array
5. Apply sorting algorithms in a Java program that organizes customer transaction data stored in an array	Data Structures and Algorithms <ul style="list-style-type: none"> • Bubble Sort • Insertion Sort • Selection Sort • Merge Sort

QUARTER 3

CONTENT STANDARD	The learners demonstrate an understanding of object-oriented Java programming principles, procedures to manage errors with exception handling, and the Java Collections Framework for efficient data management.
PERFORMANCE STANDARD	The learners create a well-structured and efficient program using Java flow control structures, Java arrays, and Java Object-Oriented Programming (OOP) language.
LEARNING COMPETENCIES	CONTENT
1. Discuss Object-Oriented Programming (OOP) Language in Java	Object-Oriented Programming (OOP) Language in Java <ul style="list-style-type: none"> • Introduction • Classes • Objects
2. Apply OOP principles in a Java program	OOP Principles <ul style="list-style-type: none"> • Encapsulation • Inheritance • Polymorphism • Abstraction
3. Apply exception handling in the Java program	Exception Handling <ul style="list-style-type: none"> • Introduction to exceptions • Types of exceptions handling <ul style="list-style-type: none"> ○ built-in exceptions ○ user-defined exceptions
4. Apply Java Collections Framework in writing Java programs	Java Collections Framework <ul style="list-style-type: none"> • Introduction to the Java Collections Framework

	<ul style="list-style-type: none"> ○ List Interfaces: ArrayList and LinkedList ○ Set Interfaces: HashSet and TreeSet ○ Map Interfaces: HashMap and TreeMap Iteration: Mastering for Loops and Iterator
5. Apply the java.io package and java file operations in a Java program	Java File Class (java.io package) Java file operation methods (to create, read, write, and delete file)
6. Apply reading and writing classes in writing a Java program	Reading and Writing Classes Reading Classes <ul style="list-style-type: none"> • FileReader • BufferedReader • Scanner • StringReader Writing Classes <ul style="list-style-type: none"> • FileWriter • BufferedWriter • StringWriter Appending vs. overwriting files

QUARTER 4

CONTENT STANDARD	The learners showcase their grasp of Java’s multithreading principles, graphical user interface (GUI) development, and efficient component structuring to build interactive and responsive applications. They incorporate market analysis to ensure their solutions meet industry demands and customer needs.
PERFORMANCE STANDARD	The learners will develop a creative GUI-based Java application that demonstrates innovative solutions to real-world problems.
LEARNING COMPETENCIES	CONTENT
1. Apply threads in creating a Java program using both the Thread class and Runnable interface	Multithreading <ul style="list-style-type: none"> • Definition of Thread • Difference between process and thread • Life cycle of a Thread (Thread States) • Advantages of multithreading in Java

	<ul style="list-style-type: none"> • Creating threads: extending the Thread class vs. implementing the Runnable interface
2. Create a Java program that effectively integrates CountdownLatch, CyclicBarrier, Semaphore	<p>Utilities of Concurrent Package</p> <ul style="list-style-type: none"> • Overview of java.util.concurrent package • Key concurrency utilities: CountdownLatch, CyclicBarrier, and Semaphore
3. Create simple GUI applications by using JFrame, JLabel, JButton, and JTextField while handling events with ActionListeners	<p>Java GUI with Swing</p> <ul style="list-style-type: none"> • Definition of Swing • Overview of the Java GUI framework. • Difference between Java Swing and Java AWT • The JFrame class for creating windows • Basic GUI components: JLabel, JButton, and JTextField • Event handling using ActionListeners
4. Create user interfaces in Java using layout managers such as FlowLayout, BorderLayout, and GridLayout, while effectively nesting components within multiple JPanels	<p>Layout Managers and Panels</p> <ul style="list-style-type: none"> • Nesting components using JPanel • Organizing components effectively using multiple panels and layout managers
5. Create interactive Java applications by leveraging advanced UI components and handling complex user interactions to enhance user experience and interface responsiveness	<p>Advanced Components and User Interaction</p> <ul style="list-style-type: none"> • Advanced components: JComboBox, JCheckBox, JRadioButton, JList • Handling multiple user inputs and responding with different actions • Using JOptionPane for dialog boxes (information, input, and confirmation)
6. Develop business plans for a system proposal	<p>Market analysis</p> <ul style="list-style-type: none"> • SWOT analysis • Feasibility study
7. Create designs of a GUI Java application showcasing creativity to solve real-world problems	<p>Complete GUI Application</p> <ul style="list-style-type: none"> • Combining various components and layouts in a real-world scenario • Organizing GUI components using menus (JMenuBar, JMenu, JMenuItem)

- | | |
|--|--|
| | <ul style="list-style-type: none">• Managing multiple windows or dialogs in an application |
|--|--|

GLOSSARY

algorithm	<p>a method in which a list of well-defined instructions for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state</p> <p>the transition incorporates randomness.</p>
compiler	<p>a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language, often having a binary form known as object code)</p> <p>the most common reason for wanting to transform source code is to create an executable program.</p>
computer	<p>a device that can accept data, internally store and execute a program of instructions, perform mathematical, logical, and manipulative operations on data, and report the results</p>
computer program	<p>a software program, or just a program; a sequence of instructions written to perform a specified task for a computer</p> <p>a computer requires programs to function, typically executing the program's instructions in a central processor. the program has an executable form that the computer can use directly to execute the instructions. the same program, in its human-readable source code form from which executable programs are derived (i.e., compiled), enables a programmer to study and develop its algorithms.</p>
computer programming	<p>an iterative process of writing or editing source code</p> <p>editing source code involves testing, analyzing, refining, and sometimes coordinating with other programmers on a jointly developed program. a person who practices this skill is referred to as a computer programmer, software developer, or coder. the sometimes-lengthy process of computer programming is usually referred to as software development.</p>
data	<p>objective measurements of the attributes (characteristics) of entities such as people, places, things, and events</p>
database	<p>a typically extensive compilation of data that is organized specifically for rapid search and retrieval, such as by a computer</p>

debugging	the elimination of errors, defects, deficiencies, or deviations of a computer program, typically operated at a significantly increased rate
documentation	a collection of documents or information
information	data placed in a meaningful and useful context for an end user
interpreter	<p>a program that directly executes instructions written in a high-level programming language without converting them into machine code</p> <p>this contrasts with a compiler, which translates the entire program into machine code before execution.</p>
Integrated Development Environment (IDE)	software that combines commonly used developer tools into a compact gui (graphical user interface) application; a combination of tools like a code editor, code compiler, and code debugger with an integrated terminal.
Java	a programming language and a platform; a high-level, robust, object-oriented, and secure programming language; an artificial language designed to express computations that can be performed by a machine, particularly a computer
programming language	can be used to create programs that control the behavior of a machine, to express algorithms precisely, or as a mode of human communication
software	computer programs and procedures concerned with the operation of an information system
source code	any collection of statements or declarations written in some human-readable computer programming language; the means most often used by programmers to specify the actions to be performed by a computer
standards	measure of performance developed to evaluate the progress of a system toward its objectives
system	an assembly of methods, procedures, or techniques unified by regulated interaction to form an organized whole
user-friendly	a characteristic of human-operated equipment and systems that makes them safe, comfortable, and easy to use

user interface

the system by which people (users) interact with a machine.

the user interface includes hardware (physical) and software (logical) components. user interfaces exist for various systems, and provide a means of 1) input, allowing the users to manipulate a system; and/or 2) output, allowing the system to indicate the effects of the users' manipulation.

REFERENCES

[Java Tutorial | Learn Java Programming - javatpoint](#)

<https://www.w3schools.com/java/default.asp>

<https://www.jdoodle.com/online-java-compiler>

Schildt, Herbert. *Java: A Beginner's Guide*. 8th ed. New York: McGraw-Hill Education, 2020.

Schildt, Herbert. *Java: The Complete Reference*. 11th ed. New York: McGraw-Hill Education, 2020.

Friesen, Jeff. *Beginning Java 8 Fundamentals*. New York: Apress, 2014. ISBN: 978-1484201250.

Magnucci, Chris, and Peter Mularien. *Java for Data Science*. Sebastopol, CA: O'Reilly Media, 2021. ISBN: 978-1492090884.

MATERIALS, TOOLS, AND EQUIPMENT

TOOLS	EQUIPMENT	MATERIALS
Computer Software e.g. - IDE - Libraries	Network Computer with peripherals	Learning materials/ guide
Internet access	Server	Practice materials
Application servers e.g. - database - web	Printer	Hand-outs
	White board	Reference books
	LCD Projector and screen	
	Ergonomic computer tables and chairs	