# Java Learning Projects: Step-by-Step Growth Path

## Introduction

To help you build your Java programming skills from the ground up, this learning path contains smaller projects that gradually increase in complexity. Each project focuses on specific concepts and prepares you for the next level until you are ready to tackle the full To-Do List project.

### Project 1: Print Even Numbers from 1 to 100

\*\*Objective\*\*: Learn how to use `for` loops and `if` conditions.

\*\*Description\*\*: Write a program that prints all even numbers from 1 to 100.

\*\*Key Concepts\*\*:

* - Using loops (`for`).
* - Using conditional statements (`if`).

### Project 2: Simple Calculator

\*\*Objective\*\*: Get familiar with user input and the `switch` statement.

\*\*Description\*\*: Write a simple calculator that performs addition, subtraction, multiplication, and division between two numbers input by the user.

\*\*Key Concepts\*\*:

* - Using `Scanner` for user input.
* - Using `switch-case` to select operations.
* - Basic arithmetic operations.

### Project 3: Character Frequency Counter

\*\*Objective\*\*: Work with strings and loops.

\*\*Description\*\*: Input a string and a character, then count and print the number of times the character appears in the string.

\*\*Key Concepts\*\*:

* - Working with strings (`String`).
* - Using loops (`for`) to iterate through a string.

### Project 4: Number Guessing Game

\*\*Objective\*\*: Learn about `while` loops and user input handling.

\*\*Description\*\*: Create a number guessing game where the user guesses a number between 1 and 100 until they get it right. Provide hints like "Higher" or "Lower" to guide the user.

\*\*Key Concepts\*\*:

* - Using `while` loops.
* - Comparing values (`if`).
* - Generating random numbers with `Math.random()`.

### Project 5: Simple Phonebook

\*\*Objective\*\*: Get familiar with lists (`ArrayList`) and managing multiple items.

\*\*Description\*\*: Create a program that allows the user to store, delete, and display phonebook entries with names and phone numbers.

\*\*Key Concepts\*\*:

* - Using `ArrayList` for storing data.
* - Adding (`add`), removing (`remove`), and displaying entries.

### Project 6: ATM Simulator

\*\*Objective\*\*: Practice handling user input and using multiple classes.

\*\*Description\*\*: Write a program that simulates an ATM with functions like checking balance, depositing, and withdrawing money.

\*\*Key Concepts\*\*:

* - Using multiple classes (`BankAccount`, `ATM`).
* - Understanding methods and objects.
* - Practicing encapsulation.

### Project 7: Library Book Management

\*\*Objective\*\*: Build a larger application with multiple methods and objects.

\*\*Description\*\*: Create an application that allows the user to manage a library of books, including adding, deleting, and searching for books.

\*\*Key Concepts\*\*:

* - Creating and using objects from classes.
* - Using `ArrayList` to manage a list of books.
* - Implementing search and sort methods.

### Project 8: Simple File I/O for User Data

\*\*Objective\*\*: Learn how to save and read data from files.

\*\*Description\*\*: Write a program that saves user information (name and age) to a file and can load it back.

\*\*Key Concepts\*\*:

* - Using `FileWriter` and `FileReader` for file operations.
* - Handling exceptions (`IOException`) while working with files.

### Project 9: Simple Task Manager

\*\*Objective\*\*: Combine learned concepts to build a basic task manager.

\*\*Description\*\*: Write a simple task manager that allows users to add, edit, and delete tasks. This version doesn't require file storage and only keeps data in memory.

\*\*Key Concepts\*\*:

* - Using multiple classes (`Task`, `TaskManager`).
* - Managing tasks using `ArrayList`.
* - Implementing basic CRUD (Create, Read, Update, Delete) operations.

### Project 10: Basic To-Do List

\*\*Objective\*\*: Use all previously learned concepts to create a complete application.

\*\*Description\*\*: Develop a To-Do List application with functions such as adding, deleting, editing tasks, and saving tasks to a file.

\*\*Key Concepts\*\*:

* - Object-Oriented Programming (OOP).
* - Managing tasks with `ArrayList`.
* - File I/O to save and load data.
* - Exception handling (`try-catch`).