**Phase 1: Foundation in Java**

1. **Learn Core Java:**
   * **Duration:** 1-2 months
   * **Topics to Cover:**
     + Basics of Java (Syntax, Data Types, Variables)
     + Control Structures (Loops, Conditionals)
     + Object-Oriented Programming (Classes, Objects, Inheritance, Polymorphism, Abstraction, Encapsulation)
     + Collections Framework (List, Set, Map)
     + Exception Handling
     + Input/Output Streams
   * **Resources:**
     + [Java Programming and Software Engineering Fundamentals by Duke University on Coursera](https://www.coursera.org/specializations/java-programming)
     + [Head First Java by Kathy Sierra and Bert Bates](https://www.amazon.com/Head-First-Java-Kathy-Sierra/dp/0596009208)
   * **Project:**
     + Build a basic CRUD application (e.g., a library management system).
2. **Java Advanced Topics:**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + Multithreading and Concurrency
     + Java Generics
     + Lambda Expressions and Stream API
     + Java Network Programming
     + Java Reflection API
     + Java 8+ Features
   * **Resources:**
     + [Effective Java by Joshua Bloch](https://www.amazon.com/Effective-Java-Joshua-Bloch/dp/0134685997)
     + [Java Concurrency in Practice by Brian Goetz](https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601)
   * **Project:**
     + Develop a multithreaded web crawler.

**Phase 2: Building Expertise in Enterprise Java**

1. **Java Enterprise Edition (Java EE) and Spring Framework:**
   * **Duration:** 3-4 months
   * **Topics to Cover:**
     + Java EE Components (Servlets, JSP, JPA, EJB)
     + Introduction to Spring Framework (Spring Core, Spring MVC, Spring Boot)
     + Dependency Injection and AOP
     + Spring Data JPA and Spring Security
     + Building RESTful APIs with Spring Boot
   * **Resources:**
     + [Spring Framework Documentation](https://spring.io/projects/spring-framework)
     + [Spring in Action by Craig Walls](https://www.amazon.com/Spring-Action-Craig-Walls/dp/1617294942)
   * **Project:**
     + Create a RESTful web service for an e-commerce platform.
2. **Database and ORM:**
   * **Duration:** 1-2 months
   * **Topics to Cover:**
     + SQL and Database Design
     + JDBC
     + Hibernate ORM
     + JPA with Spring Data
   * **Resources:**
     + [Hibernate in Action by Christian Bauer and Gavin King](https://www.amazon.com/Hibernate-Action-Christian-Bauer/dp/193239415X)
   * **Project:**
     + Develop a blog application with user authentication and role-based access using Spring Security and Spring Data JPA.

**Phase 3: Cloud Computing and DevOps Integration**

1. **AWS Cloud Services:**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + AWS EC2, S3, RDS
     + AWS Lambda and API Gateway
     + AWS IAM, CloudWatch, and CloudTrail
     + Deployment and Continuous Integration/Continuous Deployment (CI/CD) with AWS CodePipeline
   * **Resources:**
     + [AWS Certified Solutions Architect – Associate Guide](https://aws.amazon.com/certification/certified-solutions-architect-associate/)
   * **Project:**
     + Deploy a Spring Boot application on AWS using EC2, S3, and RDS. Implement CI/CD pipeline with AWS CodePipeline.
2. **Docker and Kubernetes:**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + Docker Basics (Images, Containers, Dockerfile)
     + Docker Compose
     + Kubernetes Basics (Pods, Services, Deployments)
     + Kubernetes Networking and Storage
     + Kubernetes with AWS EKS
   * **Resources:**
     + Docker Documentation
     + Kubernetes Documentation
     + [Kubernetes Up & Running: Dive into the Future of Infrastructure by Kelsey Hightower, Brendan Burns, and Joe Beda](https://www.amazon.com/Kubernetes-Running-Dive-Future-Infrastructure/dp/1492046531)
   * **Project:**
     + Containerize a Spring Boot application using Docker. Deploy and manage it on a Kubernetes cluster using AWS EKS.

**Phase 4: Advanced Topics and Tech Lead Skills**

1. **Microservices Architecture:**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + Microservices Principles and Patterns
     + Spring Cloud and Netflix OSS
     + Service Discovery and Load Balancing
     + Circuit Breakers and Resilience Patterns
     + API Gateway and Security
   * **Resources:**
     + [Building Microservices by Sam Newman](https://www.amazon.com/Building-Microservices-Designing-Fine-Grained-Systems/dp/1491950358)
   * **Project:**
     + Develop a microservices-based application using Spring Cloud and deploy it on Kubernetes.
2. **System Design and Architecture:**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + System Design Principles
     + Scalability, Reliability, and Performance
     + Database Sharding and Indexing
     + Distributed Systems
     + Design Patterns and Best Practices
   * **Resources:**
     + [Designing Data-Intensive Applications by Martin Kleppmann](https://www.amazon.com/Designing-Data-Intensive-Applications-Reliable-Maintainable/dp/1449373321)
     + [System Design Interview – An Insider's Guide by Alex Xu](https://www.amazon.com/System-Design-Interview-insiders-Second/dp/B08CMF2CQF)
   * **Project:**
     + Design a scalable, distributed system for a social media platform. Include high-level and detailed design documents.

**Phase 5: Leadership and Communication Skills**

1. **Leadership and Communication:**
   * **Duration:** Ongoing
   * **Topics to Cover:**
     + Effective Communication and Collaboration
     + Team Leadership and Mentoring
     + Project Management and Agile Methodologies
     + Conflict Resolution and Decision Making
   * **Resources:**
     + [Crucial Conversations by Kerry Patterson, Joseph Grenny, Ron McMillan, and Al Switzler](https://www.amazon.com/Crucial-Conversations-Talking-Stakes-Second/dp/1469266822)
     + [The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win by Gene Kim, Kevin Behr, and George Spafford](https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business/dp/1942788290)

**Phase 6: Real-World Experience and Continuous Learning**

1. **Gain Real-World Experience:**
   * **Duration:** Ongoing
   * **Actions:**
     + Work on real projects in a professional setting.
     + Contribute to open-source projects.
     + Participate in coding competitions and hackathons.
     + Engage in continuous learning through conferences, webinars, and online courses.

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DETAILED VERSION : **Phase 1: Foundation in Java**

1. **Learn Core Java**
   * **Duration:** 1-2 months
   * **Topics to Cover:**
     + **Basics of Java:**
       - Syntax, Data Types, Variables
       - Control Structures: Loops, Conditionals
     + **Object-Oriented Programming:**
       - Classes, Objects, Inheritance, Polymorphism, Abstraction, Encapsulation
     + **Collections Framework:**
       - List, Set, Map
     + **Exception Handling:**
       - Try-Catch, Custom Exceptions
     + **Input/Output Streams:**
       - File I/O, BufferedReader/BufferedWriter
   * **Resources:**
     + [Java Programming and Software Engineering Fundamentals by Duke University on Coursera](https://www.coursera.org/specializations/java-programming)
     + [Head First Java by Kathy Sierra and Bert Bates](https://www.amazon.com/Head-First-Java-Kathy-Sierra/dp/0596009208)
   * **Project:**
     + **CRUD Application:** Build a basic CRUD application (e.g., a library management system).
2. **Java Advanced Topics**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + **Multithreading and Concurrency:**
       - Threads, Runnable, Executor Service, Synchronization
     + **Java Generics:**
       - Type Parameters, Bounded Types
     + **Lambda Expressions and Stream API:**
       - Functional Interfaces, Streams, Parallel Streams
     + **Java Network Programming:**
       - Sockets, ServerSocket, Networking API
     + **Java Reflection API:**
       - Inspecting Classes, Methods, and Fields
     + **Java 8+ Features:**
       - Optional, New Date/Time API, Nashorn JavaScript Engine
   * **Resources:**
     + [Effective Java by Joshua Bloch](https://www.amazon.com/Effective-Java-Joshua-Bloch/dp/0134685997)
     + [Java Concurrency in Practice by Brian Goetz](https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601)
   * **Project:**
     + **Web Crawler:** Develop a multithreaded web crawler that can scrape and index web pages.

**Phase 2: Building Expertise in Enterprise Java**

1. **Java Enterprise Edition (Java EE) and Spring Framework**
   * **Duration:** 3-4 months
   * **Topics to Cover:**
     + **Java EE Components:**
       - Servlets, JSP, JPA, EJB
     + **Spring Framework:**
       - Spring Core, Spring MVC, Spring Boot
       - Dependency Injection, AOP
       - Spring Data JPA, Spring Security
       - Building RESTful APIs with Spring Boot
   * **Resources:**
     + [Spring Framework Documentation](https://spring.io/projects/spring-framework)
     + [Spring in Action by Craig Walls](https://www.amazon.com/Spring-Action-Craig-Walls/dp/1617294942)
   * **Project:**
     + **E-commerce Platform:** Create a RESTful web service for an e-commerce platform with user authentication and product management.
2. **Database and ORM**
   * **Duration:** 1-2 months
   * **Topics to Cover:**
     + **SQL and Database Design:**
       - Schema Design, Indexing, Normalization
     + **JDBC:**
       - Connecting to Databases, CRUD Operations
     + **Hibernate ORM:**
       - Session Factory, HQL, Criteria API
     + **JPA with Spring Data:**
       - Repositories, Query Methods, JPQL
   * **Resources:**
     + [Hibernate in Action by Christian Bauer and Gavin King](https://www.amazon.com/Hibernate-Action-Christian-Bauer/dp/193239415X)
   * **Project:**
     + **Blog Application:** Develop a blog application with user authentication and role-based access using Spring Security and Spring Data JPA.

**Phase 3: Cloud Computing and DevOps Integration**

1. **AWS Cloud Services**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + **Core AWS Services:**
       - EC2, S3, RDS
     + **Serverless Computing:**
       - AWS Lambda, API Gateway
     + **Security and Monitoring:**
       - IAM, CloudWatch, CloudTrail
     + **Deployment and CI/CD:**
       - AWS CodePipeline, CodeDeploy
   * **Resources:**
     + [AWS Certified Solutions Architect – Associate Guide](https://aws.amazon.com/certification/certified-solutions-architect-associate/)
   * **Project:**
     + **Spring Boot on AWS:** Deploy a Spring Boot application on AWS using EC2, S3, and RDS. Implement CI/CD pipeline with AWS CodePipeline.
2. **Docker and Kubernetes**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + **Docker Basics:**
       - Images, Containers, Dockerfile
     + **Docker Compose:**
       - Multi-Container Applications
     + **Kubernetes Basics:**
       - Pods, Services, Deployments
     + **Kubernetes Networking and Storage:**
       - Ingress, Persistent Volumes
     + **Kubernetes with AWS EKS:**
       - Setting up EKS, Deploying Applications
   * **Resources:**
     + Docker Documentation
     + Kubernetes Documentation
     + [Kubernetes Up & Running: Dive into the Future of Infrastructure by Kelsey Hightower, Brendan Burns, and Joe Beda](https://www.amazon.com/Kubernetes-Running-Dive-Future-Infrastructure/dp/1492046531)
   * **Project:**
     + **Containerized Spring Boot:** Containerize a Spring Boot application using Docker. Deploy and manage it on a Kubernetes cluster using AWS EKS.

**Phase 4: Advanced Topics and Real-World Integrations**

1. **Microservices Architecture**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + **Microservices Principles and Patterns:**
       - Decomposition, Inter-Process Communication
     + **Spring Cloud and Netflix OSS:**
       - Eureka, Ribbon, Hystrix
     + **Service Discovery and Load Balancing:**
       - Config Server, Zuul
     + **Circuit Breakers and Resilience Patterns:**
       - Resilience4j
     + **API Gateway and Security:**
       - Spring Cloud Gateway, OAuth2
   * **Resources:**
     + [Building Microservices by Sam Newman](https://www.amazon.com/Building-Microservices-Designing-Fine-Grained-Systems/dp/1491950358)
   * **Project:**
     + **Microservices Application:** Develop a microservices-based application using Spring Cloud and deploy it on Kubernetes.
2. **Kafka for Event Streaming**
   * **Duration:** 1-2 months
   * **Topics to Cover:**
     + **Kafka Basics:**
       - Producers, Consumers, Topics
     + **Kafka Streams:**
       - Stream Processing, KTable
     + **Kafka Connect:**
       - Source and Sink Connectors
     + **Kafka with Spring Boot:**
       - Spring Kafka Integration
   * **Resources:**
     + [Kafka: The Definitive Guide by Neha Narkhede, Gwen Shapira, and Todd Palino](https://www.amazon.com/Kafka-Definitive-Real-Time-Streaming-Data/dp/1491936169)
   * **Project:**
     + **Event-Driven Application:** Implement an event-driven application using Kafka for asynchronous processing.
3. **Redis for Caching and Data Storage**
   * **Duration:** 1 month
   * **Topics to Cover:**
     + **Redis Basics:**
       - Data Structures, Persistence
     + **Caching Strategies:**
       - Cache Aside, Read Through, Write Through
     + **Redis with Spring Boot:**
       - Spring Data Redis
   * **Resources:**
     + Redis Documentation
   * **Project:**
     + **Caching Layer:** Integrate Redis as a caching layer in an existing Spring Boot application to improve performance.

**Phase 5: System Design and Architecture**

1. **System Design and Architecture**
   * **Duration:** 2-3 months
   * **Topics to Cover:**
     + **System Design Principles:**
       - Scalability, Reliability, Performance
     + **Database Sharding and Indexing:**
       - Sharding Strategies, Indexing Techniques
     + **Distributed Systems:**
       - CAP Theorem, Consistency Models, Replication
     + **Design Patterns and Best Practices:**
       - Singleton, Factory, Observer, Proxy
   * **Resources:**
     + [Designing Data-Intensive Applications by Martin Kleppmann](https://www.amazon.com/Designing-Data-Intensive-Applications-Reliable-Maintainable/dp/1449373321)
     + [System Design Interview – An Insider's Guide by Alex Xu](https://www.amazon.com/System-Design-Interview-insiders-Second/dp/B08CMF2CQF)
   * **Project:**
     + **Scalable System Design:** Design a scalable, distributed system for a social media platform. Include high-level and detailed design documents.

**Phase 6: Leadership and Communication Skills**

1. **Leadership and Communication**
   * **Duration:** Ongoing
   * **Topics to Cover:**
     + **Effective Communication and Collaboration:**
       - Writing clear documentation, effective meetings
     + **Team Leadership and Mentoring:**
       - Coaching, Performance Reviews
     + **Project Management and Agile Methodologies:**
       - Scrum, Kanban, Agile Principles
     + **Conflict Resolution and Decision Making:**
       - Mediation, Consensus Building
   * **Resources:**
     + [Crucial Conversations by Kerry Patterson, Joseph Grenny, Ron McMillan, and Al Switzler](https://www.amazon.com/Crucial-Conversations-Talking-Stakes-Second/dp/1469266822)
     + [The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win by Gene Kim, Kevin Behr, and George Spafford](https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business/dp/1942788290)

**Phase 7: Real-World Experience and Continuous Learning**

1. **Gain Real-World Experience**
   * **Duration:** Ongoing
   * **Actions:**
     + Work on real projects in a professional setting.
     + Contribute to open-source projects.
     + Participate in coding competitions and hackathons.
     + Engage in continuous learning through conferences, webinars, and online courses.

branch -> git branch “name\_of\_the\_branch”

Checkout -> git checkout “name\_of\_branch”

Combine creating and checking out -> git checkout -b “name\_of\_the\_branch” -> 3 is a combination of 1 and 2

Modify a file -> staged or unstaged -> commit is possible only for the staged files

Prerequisitive for committing is a staging a file

Stage -> git add “name of the file”

List the files -> git status

Add everything -> “git add .”

Commit -> git commit -m “Merge the feature”

Source branch -> feature, destination branch -> master

Git checkout master -> git merge feature

Merging

Fast forward merge

Squash merge

Normal Merge/ Three way merge

If a single line has been edited by multiple people -> Merge conflict

How do you resolve the merge conflict -> Manually

Take the change of source branch

Take the change of destination branch

Take both the changes

Stash -> Way to keep your current working directory -> save the work that you have done without need of commit

Save to the stash -> git stash save

List the stash -> git stash list

Apply the stash -> git stash apply

Pop the stash -> git stash pop -> also removes the stash from the list

Stack -> LIFO -> last would be removed by pop

Selectively -> git stash apply stack@{0}

Pop -> git stash drop stack@{0}

The Top 12 Java Projects That Will Enhance Your Portfolio

Data visualisation software

As data visualisation is concerned with the graphical or pictorial representation of data, data visualisation software makes it more efficient and convenient for users to interpret information expressed in charts or graphs rather than report pages. The data visualisation software project will use data visualisation to show node connectivity in networking.

The key objectives of this project should be to explicitly communicate information using graphical and pictorial methods. These are both useful as well as visually pleasing. And they eloquently express essential insights into complex collections of data and information, and so on.

Social Networking Platform

The social networking platform project is best suited to intermediate Java programmers. You may devise a social networking platform in Java that includes all of the essential elements, such as a chatbox, calling features, posting images, likes, comments, and sharing of posts, a friend request module, and so on. You can also choose to include any extra special features or functions in the project.

Meanwhile, you may use Java's numerous APIs to make things easier and more convenient. However, in order to complete the social networking platform project, you must have a deep understanding of several other disciplines, including computer networking, databases, and a variety of others, in addition to your Java language skills.

Number Guessing Game

This is a simple project that you can complete quickly and add to your portfolio, especially if you are a novice. In the number guessing game, you must create an app that generates a random number that the user must guess. Users will receive points if they guess correctly.

The rules of the number guessing game are completely up to you. For example, you may set a limit on the number of attempts, choose a range for number generation, and award points based on the number of attempts, among other things.

In the number guessing game, the computer must generate a random number from 1 to 100 that falls within the set range.

After that, the user is asked to guess the number and enter it in an input area. After comparing the generated number with the user's input, the user will receive a message indicating whether their guess was correct or incorrect with respect to the randomly created number.

Snake Game

Java is a sophisticated programming language with the added bonus of being able to build games with it. In fact, you might be interested to know that one of the most famous sandbox games, Minecraft, was built using Java. Nevertheless, this Java project demands that you create a classic 2D snake game similar to those you may have played as a kid.

Building the snake game will allow you to understand and use many Java principles. To develop the snake game efficiently, you'll also need to apply object-oriented programming techniques. You must make sure that the snake figure can be moved in all four directions by the player and that the snake food appears at random locations across the play area. You should additionally take care if the snake acquires or eats food, its length will increase. Also, the gaming session should end as the snake reaches the edge of the playing region or bumps into its own body.

Online Survey Tool

Surveys allow businesses to garner crucial information from customers. This information can help the business to curate sound marketing strategies and iterate products or services to resolve the pain points of clients.

For this survey tool, you'll use many Java concepts to set up a new survey system that companies can use to create and distribute surveys to consumers. Customers can be emailed the results of the surveys if the tool allows it.

You may also use it to restrict access to certain surveys to certain users or clients. When developing such a system, such as MySQL or PostgreSQL, you must utilise a database management system (DBMS) to store the data.

Surveys help companies get vital information from customers that can aid in the development of apt marketing strategies, the modification of products or services to match specific customer needs, and so on.

Traffic Controller

Beginner and intermediate Java programmers should consider working on the traffic controller system project. The goal of this project is to devise an automated system for traffic control management. This project will have multiple modules to execute various operations such as traffic light control, traffic route management, traffic police supervision for a certain area, and so on.

Furthermore, with a solid understanding of other topics such as OOPs, Servlets, and so on, you may develop the traffic controller system project in Java using JSPs and MySQL. You may even take the project to the next level by adding more complex functionality and using Spring, Hibernate, and other technologies.

Digital Clock

This is a project for amateurs that can be executed in a short amount of time. You're probably familiar with the concept of a digital clock. You only need to display the current time on the digital clock you're working on. Your digital clock can display seconds in addition to minutes and hours.

Displaying the day of the week and the date on your digital clock will give it more utility. A timer, a stopwatch, and a timezone are just a few of the other functions that may be added to the digital clock.

To build a digital clock, you'll need to be well versed in user interface design, as your digital clock will need to be aesthetically attractive. To make everything simple, you could devise a digital clock that has a simplistic style.

Temperature Convertor

If you're new to Java, creating a temperature converter is a fantastic project to get started. A temperature conversion application's principle is simple: it should convert temperatures from Fahrenheit to Celsius and vice versa.

The mathematical equation for translating temperature from Fahrenheit to Celsius and Celsius to Fahrenheit is the first thing you need to learn to construct this application. You must design an interface with one input field, a "Convert" button, a section that displays the output, and another button that allows the user to switch from Celsius to Fahrenheit.

You can also include a code to respond to an inappropriate input. If a user enters a non-numerical value, the converter should produce a message like "Invalid Input."

System for Student Management

Your CV would greatly benefit from a student management system. You will find it easier to construct this project if you have already built a few beginner-level Java projects.

A student management system, in general, allows a school administration to manage students and optimise numerous operations, such as student enrollment and course feedback.

You must divide a student management system into various sections or modules in order to make it effective. For example, the system can be divided into three main modules: administrator, student, and teacher. In addition, all these modules must have their own interface.

One or more people can use the administrator interface to take control of the entire system. In other words, an administrator has the ability to manage student and teacher accounts, monitor all data, and enable or disable services that are available to students and teachers.

The student module manages student accounts and allows students to log in, post assignments, check comments, and score, among other things. Teachers can also log in to their accounts, to monitor student attendance, check submitted assignments, and upload exam results.

Customer Relationship Management (CRM) Software

This is one of the more difficult Java projects, requiring a thorough understanding of several Java principles as well as previous experience working on a variety of Java projects. Contact management, workflow automation, lead management, and quote and order management are just a few of the capabilities of a CRM (Customer Relationship Management) system.

A CRM's primary goal is to assist a company in boosting sales, providing efficient customer service, and effectively interacting with customers.

To make the whole thing more manageable, you should develop a distinct module for each function you add to your CRM. Multiple users should be able to create accounts under a single corporate entity, while having access to customer queries and other data using the CRM.

In general, there are a number of functions that you can include in your CRM. Keep in mind, however, that the more features you add to the CRM, the more complicated it will become, so plan accordingly.

Billing System

One of the most popular projects for refining your Java programming skills is the billing system. The basic objective of this billing system project is to automatically compute bills, eliminating the need for manual calculations. You might want to start by creating a simple interface that requires the user to enter the invoice number, price per unit, and number of units.

The system will generate a bill based on these facts, including the actual amount that must be paid. The wonderful thing about creating a billing system project in Java is that you may change the project's level from amateur to advanced depending on your skills and requirements. You can, for instance, create more complicated projects such as grocery billing systems, apparel billing systems, or electricity billing systems.

Supply Chain Management Software

Clients can directly communicate their item requirements to the manufacturer, who then contacts a number of vendors to obtain the necessary materials. Vendors typically make a list of items based on the items given to them, after which the manufacturer chooses the materials that best suit the client's specifications. The chosen list of components is delivered to the stock division for processing, after which the assembly process begins. When the work is finished, the records office adds up the cost of the raw materials and the cost of gathering them to produce the total bill. Finally, the customer receives the goods along with a receipt.