### **Week 1: Welcome to the HotelKey Internship Program!**

Welcome, interns! This week is all about building a strong foundation. We'll be diving into the core technologies and tools you'll be using throughout your time at HotelKey. Our goal is to get you comfortable with the essentials of **Java development**, understand key **data structures**, and introduce you to the **cloud services** that power our applications. Let's get started! 🚀

### **Objectives for Week 1**

By the end of this week, you will be able to:

* **Set up** a complete Java development environment on your machine.
* **Write and run** basic Java programs using Visual Studio Code or IntelliJ IDEA.
* **Explain** the fundamental concepts of Java, including the distinction between methods and functions.
* **Describe** the use cases for common data structures like Linked Lists and Trees.
* **Understand** the basics of garbage collection in Java.
* **Simulate** AWS services on your local machine using LocalStack.

### **📚 Readings: Building Your Knowledge Base (Day 1)**

This week's readings will guide you through the fundamentals of Java and introduce you to important concepts in data management and cloud services.

* [**Getting Started with Java**](https://dev.java/learn/getting-started/)
  + **Description:** An official guide from Oracle that provides a comprehensive introduction to the Java programming language. It's the perfect starting point if you're new to Java or need a refresher.
  + **Focus:** Core syntax, object-oriented principles, and setting up your first project.
* [**Methods vs. Functions in Java for Beginners**](https://medium.com/@AlexanderObregon/methods-vs-functions-in-java-for-beginners-0a22c5893aef)
  + **Description:** A clear explanation of a common point of confusion for new Java developers. Understanding this distinction is key to writing clean Java code.
  + **Focus:** The conceptual difference and practical application of methods and functions in an object-oriented context.
* [**Efficient Data Management with Linked Lists in Java**](https://medium.com/javarevisited/efficient-data-management-with-linked-lists-in-java-a-beginners-guide-b77af1b3c7c5)
  + **Description:** This article introduces you to Linked Lists, a fundamental data structure. You'll learn how they work and when to use them.
  + **Focus:** Implementation and advantages of using Linked Lists for dynamic data.
* [**Garbage Collections in Java for Beginners**](https://code.likeagirl.io/garbage-collections-in-java-for-beginners-27750b0ee1f7)
  + **Description:** An introduction to how Java manages memory automatically through garbage collection. This is a crucial concept for writing efficient and stable applications.
  + **Focus:** Understanding the process of automatic memory management in the Java Virtual Machine (JVM).
* [**A Comprehensive Guide to Tree Data Structures in Java**](https://medium.com/@YodgorbekKomilo/a-comprehensive-guide-to-tree-data-structures-in-java-from-beginner-to-advanced-2232d0902208)
  + **Description:** This guide covers another essential data structure: Trees. You'll explore different types of trees and their applications.
  + **Focus:** Understanding hierarchical data representation and traversal algorithms.
* [**Getting Started with AWS Services using LocalStack**](https://dev.to/tnz5/getting-started-with-aws-services-using-localstack-573d)
  + **Description:** An introduction to LocalStack, a tool that lets you simulate Amazon Web Services (AWS) on your own computer. This is invaluable for development and testing.
  + **Focus:** Learning how to mimic cloud services locally to build and test applications without needing an AWS account.

### **🛠️ Tools: Your Development Toolkit (Day 1)**

Below are the essential tools you'll need to install this week. These will be your primary environments for writing and running code at HotelKey.

* [**IntelliJ IDEA**](https://www.jetbrains.com/idea/download/?section=mac)
  + **Description:** A powerful and intelligent Integrated Development Environment (IDE) for Java. It provides advanced features for code completion, debugging, and analysis that will boost your productivity.
  + **Objective:** Install and configure IntelliJ IDEA to get a feel for a professional-grade Java development environment.
* [**Java in Visual Studio Code**](https://code.visualstudio.com/docs/java/java-tutorial)
  + - **Description:** This tutorial walks you through setting up and using Visual Studio Code for Java development. VS Code is a popular, lightweight editor you'll find useful.
    - **Focus:** Installing necessary extensions, running, and debugging Java code within VS Code.
* [**LocalStack**](https://docs.localstack.cloud/getting-started/installation/)
  + **Description:** A cloud service emulator that runs in a single container on your laptop or in your CI environment. It allows you to develop and test your cloud and serverless apps entirely on your local machine.
  + **Objective:** Install LocalStack to begin working with AWS services in a local, offline environment.

**Submission Folder:** [Week 1 Java Exercises](https://drive.google.com/drive/folders/1Dg-wVfaK7D6OXAzj2-afom9eFK6gn_Ue?usp=drive_link)

Please create a separate project or file for each exercise and upload your source code to the shared Google Drive folder.

### **Part 1: Java Fundamentals & Object-Oriented Programming**

These exercises focus on core Java syntax and OOP principles.

* **Exercise 1: Room Class:** Create a Room class with properties for roomNumber (int), roomType (String, e.g., "Single", "Suite"), and isOccupied (boolean). Include a constructor and a method to display the room's status.
* **Exercise 2: Hotel Static Properties:** Create a Hotel class. Use **static** variables for hotelName and address. Add a static method to print these details without creating an instance of the class.
* **Exercise 3: Method Overloading:** In your Room class, create two assignGuest methods. One should just take a Guest object. The other should take a Guest object and an int for the number of nights.
* **Exercise 4: Encapsulation:** Make all properties in your Guest and Room classes **private**. Create public "getter" and "setter" methods to access and modify them safely.
* **Exercise 5: Inheritance:** Create a VIPGuest class that **extends** the Guest class. Add a new property like membershipLevel (String). Override the method that displays guest info to include this new detail.
* **Exercise 6: Abstract Class:** Design an abstract class Person with properties for name and an abstract method getRole(). Have the Guest class and a new Staff class extend Person and implement the method.
* **Exercise 7: Interface:** Create an interface named Bookable with a method signature boolean isAvailable(). Have your Room class implement this interface.

### **Part 2: Java Data Structures**

These exercises will test your ability to use Java's built-in data structures.

* **Exercise 8: LinkedList for Housekeeping:** Use a LinkedList to manage a queue of Room objects that need cleaning. Write a program to add rooms to the queue and "clean" them by removing them from the front of the list.
* **Exercise 9: ArrayList of Guests:** Create an ArrayList to hold multiple Guest objects. Write code to add 5 guests, then remove the second guest from the list, and finally iterate through the list to display info for the remaining guests.
* **Exercise 10: HashMap for a Room Directory:** Use a HashMap<Integer, Guest> to map a roomNumber (key) to a Guest object (value). Implement a function that allows you to find which guest is in a specific room.
* **Exercise 11: HashSet for Hotel Amenities:** A hotel offers a list of unique amenities. Use a HashSet<String> to store them. Add several amenities, including a few duplicates, and print the set to demonstrate that only unique items are stored.
* **Exercise 12: Stack for Luggage:** Simulate luggage storage using a Stack. The last bag checked in is the first one to be retrieved (LIFO). Implement push (store luggage) and pop (retrieve luggage) operations for a stack of luggage tags (Strings).
* **Exercise 13: Binary Search Tree (BST) Insertion:** Create a basic Node class. Write a program that creates a BST of reservation confirmation numbers (integers). Implement an insert method that adds new numbers to the tree correctly.
* **Exercise 14: BST In-Order Traversal:** Using the tree from the previous exercise, write a method that performs an "in-order" traversal to print all reservation numbers in ascending order.
* **Exercise 15: BST Search:** Write a search method for your BST that takes a reservation number and returns true if it exists in the tree and false otherwise.

### **Part 3: Combining Concepts & Error Handling**

These exercises require you to integrate multiple concepts.

* **Exercise 16: Basic Check-In System:** Create a Hotel class that contains an ArrayList<Room>. Write a checkIn method that finds the first available room in the list and assigns a Guest to it by setting the room's isOccupied status to true.
* **Exercise 17: Null Pointer Safety:** Modify your checkIn method from the previous exercise. If no rooms are available, it should print a "Hotel is full" message instead of causing a NullPointerException or other error.
* **Exercise 18: Overriding equals():** Override the equals() and hashCode() methods in your Guest class. Two Guest objects should be considered equal if they have the same name.
* **Exercise 19: Custom Exception:** Create a custom checked exception named RoomNotAvailableException. In your Hotel class, make the checkIn method throw this exception if the desired room is occupied. Catch this exception in your main method.
* **Exercise 20: Simple Waitlist:** Modify your Hotel class. If a checkIn fails because no rooms are available, add the Guest to a LinkedList that serves as a waitlist. Write a method to display all guests currently on the waitlist.

### **📚 Readings: Building Your Knowledge Base (Day 2)**

This week's readings will guide you through the fundamentals of Java and introduce you to important concepts in databases

🔗 [**Java 8 Streams Tutorial**](https://www.baeldung.com/java-8-streams)

* ✨ **What it offers:** A comprehensive guide to understanding and utilizing Java 8 Streams API, a powerful feature for processing collections of data in a functional and expressive way. Essential for modern Java development.

🔗 [**NoSQL Explained (MongoDB Basics)**](https://www.mongodb.com/resources/basics/databases/nosql-explained)

* ✨ **What it offers:** An excellent introduction to the world of NoSQL databases. This resource explains what NoSQL is, why it emerged, and its fundamental principles, perfect for getting started with non-relational data stores.

🔗 [**NoSQL vs. SQL: Key Differences**](https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql#differences-between-sql-and-nosql)

* ✨ **What it offers:** A focused comparison highlighting the core differences between traditional SQL (relational) databases and modern NoSQL databases. This is crucial for understanding when to choose one paradigm over the other for your specific application needs.

**Employees Table:**

* employee\_id (INT, Primary Key)
* first\_name (VARCHAR)
* last\_name (VARCHAR)
* email (VARCHAR, UNIQUE)
* phone\_number (VARCHAR)
* hire\_date (DATE)
* job\_id (INT)
* salary (DECIMAL)
* department\_id (INT)

**Jobs Table:**

* job\_id (INT, Primary Key)
* job\_title (VARCHAR)
* min\_salary (DECIMAL)
* max\_salary (DECIMAL)

**Departments Table:**

* department\_id (INT, Primary Key)
* department\_name (VARCHAR)
* location\_id (INT)

**Locations Table:**

* location\_id (INT, Primary Key)
* street\_address (VARCHAR)
* city (VARCHAR)
* state\_province (VARCHAR)
* postal\_code (VARCHAR)
* country\_id (VARCHAR)

### **Basic SQL Queries (SELECT, WHERE, ORDER BY, LIMIT)**

**Exercise 1: Employees in Department 90** List all employees belonging to department\_id 90.

**Exercise 2: Employees Sorted by Last Name** Retrieve first\_name, last\_name, and salary for all employees, sorted in ascending order by last\_name.

**Exercise 3: Top 5 Highest Paid Employees** Find the top 5 employees with the highest salary.

**Exercise 4: Employees with "IT" in Job Title** Select employee\_id, first\_name, and job\_title for employees whose job title contains the word "IT". (Hint: Use LIKE and %).

### **Intermediate SQL Queries (JOINs, Aggregate Functions, GROUP BY, HAVING)**

**Exercise 9: Employee Name and Job Title** Get the first\_name, last\_name, and job\_title for all employees. (Hint: Use JOIN between Employees and Jobs tables).

**Exercise 12: Average Salary by Department** Calculate the average salary for each department\_id.

**Exercise 13: Count Employees per Job Title** Count how many employees are in each job\_title.

**Exercise 14: Departments with More Than 5 Employees** List the department\_name and the count of employees for departments that have more than 5 employees.

**Exercise 15: Employees Not in Any Department** Find employees who do not have a department\_id assigned. (Hint: Use IS NULL).

**Exercise 16: Employees and Their Location City** Get the first\_name, last\_name, and the city where their department is located.

**Exercise 17: Employees Who Earn More Than Their Job's Minimum Salary** List employee\_id, first\_name, salary, and job\_title for employees whose salary is greater than the min\_salary for their job\_id.

### **Advanced SQL Queries (Subqueries, EXISTS, UNION, CASE, Window Functions - Optional)**

**Exercise 18: Employees Who Earn the Maximum Salary in Their Department** Find the first\_name, last\_name, department\_id, and salary for employees who earn the maximum salary within their respective department. (Hint: Use a subquery).

**Exercise 19: Departments with No Employees** Identify department\_names that currently have no employees.

**Exercise 20: Employees Whose Salary is Greater Than the Average Salary of All Employees** List employee\_id, first\_name, last\_name, and salary for employees whose salary is higher than the overall average salary of all employees.

**Exercise 21: Employees Who Work in 'Sales' or 'IT' Departments** Retrieve first\_name, last\_name, and department\_name for employees working in the 'Sales' or 'IT' departments. (Hint: You can use IN with a subquery or multiple JOINs and OR).

**Exercise 22: Job Titles with Average Salary Above 7000** Find job\_titles where the average salary of employees holding that job is greater than 7000.

**Exercise 23: Employees with Duplicate Emails (If Applicable)** Assume your email column might not be UNIQUE (for this exercise only, if you strictly enforced it before). Find any email addresses that appear more than once in the Employees table.

**Exercise 24: Employee Salary Rank (Window Function)** *This is an advanced exercise using window functions. If you're new to them, research RANK() or DENSE\_RANK().* For each department, list employees along with their salary rank within that department, ordered by salary descending.

**Exercise 25: Case Statement for Salary Grades** Categorize employees into salary grades based on their salary:

* 'Low' for salary < 5000
* 'Medium' for salary >= 5000 and < 10000
* 'High' for salary >= 10000 Display first\_name, last\_name, salary, and the salary\_grade.