Task 1: **Create a Basic CRUD Operation in Java Spring Boot.**

**1. Book Entity**

import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
  
@Entity  
public class Book {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 private String title;  
 private String author;  
 private String isbn;  
  
 // Constructors (optional, but useful)  
 public Book() {  
 }  
  
 public Book(String title, String author, String isbn) {  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 }  
  
 // Getters and setters  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public String getIsbn() {  
 return isbn;  
 }  
  
 public void setIsbn(String isbn) {  
 this.isbn = isbn;  
 }  
  
 // toString() method (optional, but helpful for logging)  
 @Override  
 public String toString() {  
 return "Book{" +  
 "id=" + id +  
 ", title='" + title + '\'' +  
 ", author='" + author + '\'' +  
 ", isbn='" + isbn + '\'' +  
 '}';  
 }  
}

**2. Book Repository**

import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
 // Spring Data JPA provides basic CRUD methods for free  
 // You can add custom query methods here if needed  
}

**3. Book Controller**

import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 // Create (POST)  
 @PostMapping  
 public ResponseEntity<Book> createBook(@RequestBody Book book) {  
 Book savedBook = bookRepository.save(book);  
 return new ResponseEntity<>(savedBook, HttpStatus.CREATED);  
 }  
  
 // Read All (GET)  
 @GetMapping  
 public ResponseEntity<List<Book>> getAllBooks() {  
 List<Book> books = bookRepository.findAll();  
 return new ResponseEntity<>(books, HttpStatus.OK);  
 }  
  
 // Read One (GET)  
 @GetMapping("/{id}")  
 public ResponseEntity<Book> getBookById(@PathVariable Long id) {  
 Book book = bookRepository.findById(id).orElse(null);  
 if (book != null) {  
 return new ResponseEntity<>(book, HttpStatus.OK);  
 } else {  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
 }  
  
 // Update (PUT)  
 @PutMapping("/{id}")  
 public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book bookDetails) {  
 Book book = bookRepository.findById(id).orElse(null);  
 if (book != null) {  
 book.setTitle(bookDetails.getTitle());  
 book.setAuthor(bookDetails.getAuthor());  
 book.setIsbn(bookDetails.getIsbn());  
  
 Book updatedBook = bookRepository.save(book);  
 return new ResponseEntity<>(updatedBook, HttpStatus.OK);  
 } else {  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
 }  
  
 // Delete (DELETE)  
 @DeleteMapping("/{id}")  
 public ResponseEntity<Void> deleteBook(@PathVariable Long id) {  
 if (bookRepository.existsById(id)) {  
 bookRepository.deleteById(id);  
 return new ResponseEntity<>(HttpStatus.NO\_CONTENT);  
 } else {  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
 }  
}