# Phonebook Application Java Code

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Contact {
  String name;
  String phone;
  String email;

  public Contact(String name, String phone, String email) {
    this.name = name;
    this.phone = phone;
    this.email = email;
  }
}

public class Phonebook {
  private ArrayList<Contact> contacts;

  public Phonebook() {
    this.contacts = new ArrayList<>();
  }

  // 1. Insert Contact
  public void insertContact(String name, String phone, String email) {
        Contact newContact = new Contact(name, phone, email);
        contacts.add(newContact);
    }

    // 2. Search Contact
    public Contact searchContact(String name) {
        for (Contact contact : contacts) {
            if (contact.name.equalsIgnoreCase(name)) {
                return contact;
            }
        }
        return null; // Not found
    }

    // 3. Display All Contacts
    public void displayContacts() {
        for (Contact contact : contacts) {
            System.out.println("Name: " + contact.name + ", Phone: " + contact.phone + ", Email: " + contact.email);
        }
    }

    // 4. Delete Contact
    public boolean deleteContact(String name) {
        for (int i = 0; i < contacts.size(); i++) {
            if (contacts.get(i).name.equalsIgnoreCase(name)) {
                contacts.remove(i);
                return true; // Successfully deleted
            }
        }
        return false; // Not found
    }

    // 5. Update Contact
    public boolean updateContact(String name, String newPhone, String newEmail) {
        Contact contact = searchContact(name);
        if (contact != null) {
            contact.phone = newPhone;
            contact.email = newEmail;
            return true; // Successfully updated
        }
        return false; // Not found
    }

    // 6. Sort Contacts
    public void sortContacts() {
        Collections.sort(contacts, Comparator.comparing(contact -> contact.name));
    }

    // 7. Analyze Search Efficiency
    public String analyzeSearchEfficiency() {
        return "The search algorithm is a linear search with time complexity O(n). " +
               "Best Case: O(1) (if the contact is the first element). " +
               "Worst Case: O(n) (if the contact is the last element or not present).";
    }
}
```