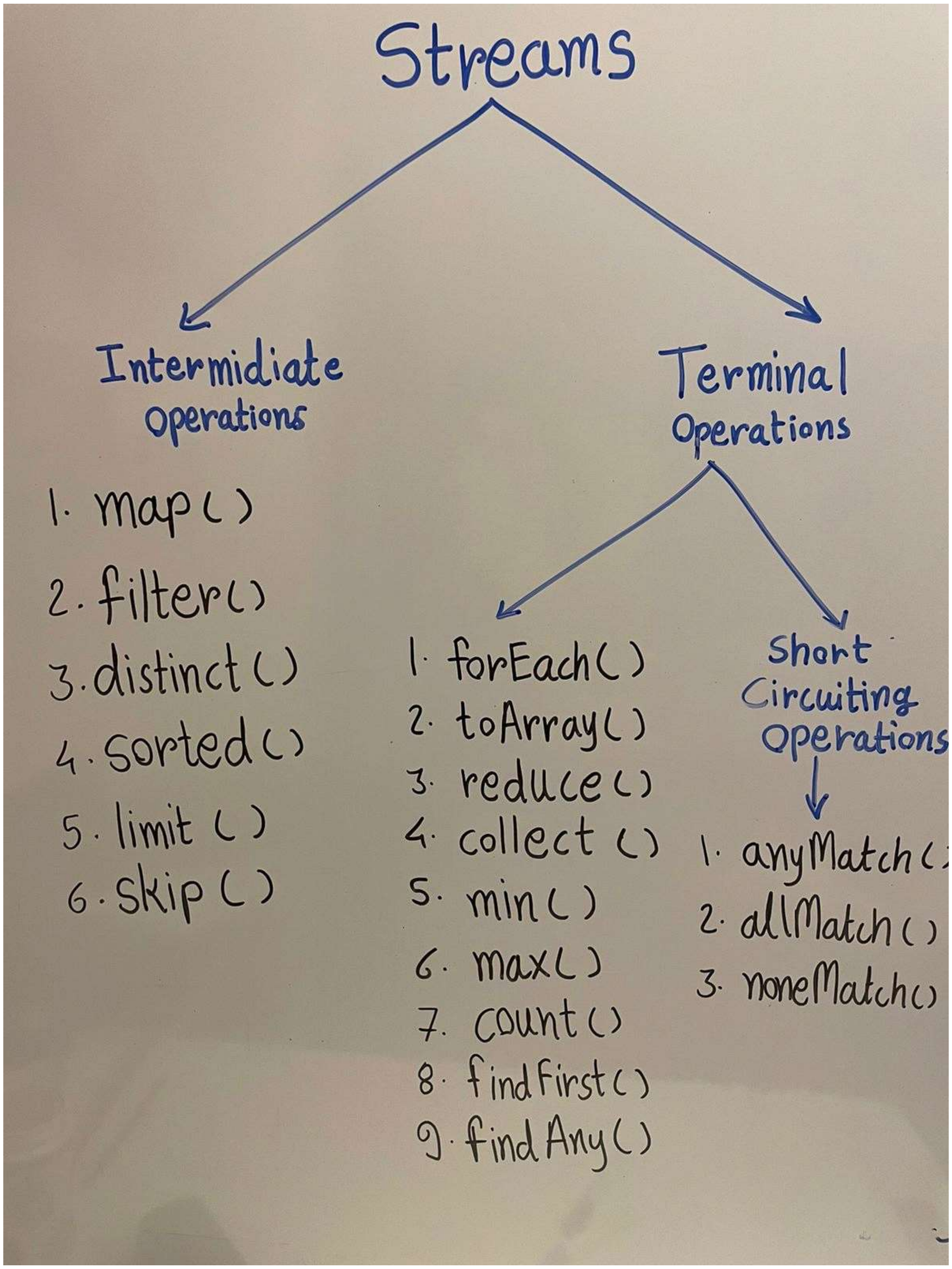

Image listing the stream operations



StreamIntermediateOperationsDemo.java

```
import java.util.List;
import java.util.stream.Collectors;

class Item {
    private String name;
    private double price;

    public Item(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }
}

public class StreamIntermediateOperationsDemo {
    public static void main(String[] args) {
        // Simulate a shopping cart with items
        List<Item> shoppingCart = List.of(
            new Item("Laptop", 1000.0),
            new Item("Phone", 500.0),
            new Item("Headphones", 100.0),
            new Item("Tablet", 800.0),
            new Item("Keyboard", 50.0)
        );

        System.out.println("Original Shopping Cart:");
        shoppingCart.forEach(item -> System.out.println(item.getName() + " - $" +
item.getPrice()));

        // Intermediate operations using streams
        List<String> selectedItems = shoppingCart.stream()
                .filter(item -> item.getPrice() > 100.0)         // Filter expensive items
                .map(Item::getName)                              // Get names of items
            .distinct()                          // Remove duplicates
            .sorted()                            // Sort item names
            .skip(1)                             // Skip the first item
            .limit(2)                            // Limit to the next 2 items
            .collect(Collectors.toList());

        System.out.println("\nIntermediate Operations Result:");
        selectedItems.forEach(System.out::println);
    }
}
```

BookStreamTerminalOperationsDemo .java

```
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

class Book {
    private String title;
    private String author;
    private double price;
    private int pageCount;
    private boolean available;

    public Book(String title, String author, double price, int pageCount, boolean available) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.pageCount = pageCount;
        this.available = available;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public double getPrice() {
        return price;
    }

    public int getPageCount() {
        return pageCount;
    }

    public boolean isAvailable() {
        return available;
    }
}

public class BookStreamTerminalOperationsDemo {
    public static void main(String[] args) {
        List<Book> books = List.of(
            new Book("The Great Gatsby", "F. Scott Fitzgerald", 12.99, 180, true),
            new Book("To Kill a Mockingbird", "Harper Lee", 10.50, 281, true),
            new Book("1984", "George Orwell", 9.99, 328, true),
            new Book("Pride and Prejudice", "Jane Austen", 7.95, 279, false),
            new Book("The Hobbit", "J.R.R. Tolkien", 14.95, 310, true)
        );

        // forEach - Print details of each available book
        System.out.println("Available Books:");
        books.stream()
            .filter(Book::isAvailable)
            .forEach(book -> System.out.println(
                "Title: " + book.getTitle() +
                " Author: " + book.getAuthor() +
                " Price: $" + book.getPrice()
            ));

        // count - Count the total number of books
        long bookCount = books.stream()
            .count();
        System.out.println("\nTotal Books: " + bookCount);

        // collect - Collect details of books by J.R.R. Tolkien into a list
        List<String> tolkienBooks = books.stream()
            .filter(book -> book.getAuthor().equals("J.R.R. Tolkien"))
            .map(Book::getTitle)
            .collect(Collectors.toList());
        System.out.println("\nTolkien Books: " + tolkienBooks);

        // min - Find the cheapest book price
        Optional<Double> cheapestPrice = books.stream()
            .map(Book::getPrice)
            .min(Double::compare);
        System.out.println("\nCheapest Book Price: $" + cheapestPrice.orElse(-1.0));

        // max - Find the highest page count
        Optional<Integer> highestPageCount = books.stream()
            .map(Book::getPageCount)
            .max(Integer::compare);
        System.out.println("\nHighest Page Count: " + highestPageCount.orElse(-1));

        // findFirst - Find the details of the first book
        Book firstBook = books.stream()
            .findFirst()
            .orElse(null);
        System.out.println("\nFirst Book: " + (firstBook != null ? firstBook.getTitle() : "None"));

        // findAny - Find details of any available book
        Book anyAvailableBook = books.stream()
            .filter(Book::isAvailable)
            .findAny()
            .orElse(null);
        System.out.println("\nAny Available Book: " + (anyAvailableBook != null ?
anyAvailableBook.getTitle() : "None"));

        // allMatch - Check if all books are available
        boolean allAvailable = books.stream()
            .allMatch(Book::isAvailable);
        System.out.println("\nAll Books Available: " + allAvailable);

        // anyMatch - Check if any book has more than 300 pages
        boolean anyLongBook = books.stream()
            .anyMatch(book -> book.getPageCount() > 300);
        System.out.println("\nAny Book with More than 300 Pages: " + anyLongBook);

        // noneMatch - Check if no book is out of stock
        boolean noneOutOfStock = books.stream()
            .noneMatch(book -> !book.isAvailable());
        System.out.println("\nNo Books Out of Stock: " + noneOutOfStock);

        // reduce - Calculate the total price of all books
        double totalPrice = books.stream()
            .map(Book::getPrice)
            .reduce(0.0, Double::sum);
        System.out.println("\nTotal Price of Books: $" + totalPrice);

        // toArray - Convert book titles to an array
        String[] bookTitlesArray = books.stream()
            .map(Book::getTitle)
            .toArray(String[]::new);
        System.out.println("\nBook Titles: " + java.util.Arrays.toString(bookTitlesArray));
    }
}
```