

## Part 1: The Set Interface – For Unique Elements

### Exercise 1.1: Understanding Sets vs. Lists

```
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.Set;

public class Exercise1_1 {
    public static void main(String[] args) {
        ArrayList<String> nameList = new ArrayList<>();
        nameList.add("Alice");
        nameList.add("Bob");
        nameList.add("Charlie");
        nameList.add("Alice");
        nameList.add("David");
        nameList.add("Bob");

        nameList.forEach(System.out::println);

        System.out.println("LinkedHashSet");
        LinkedHashSet<String> nameSet = new LinkedHashSet<>(nameList);
        nameSet.forEach(System.out::println);
    }
}
```

### Exercise 1.2: Comparing Set Implementation with Custom Objects

```
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.TreeSet;

public class Book implements Comparable<Book>{
    private String title;
    private int publishYear;
    public Book(String title, int publishYear) {
        this.title = title;
        this.publishYear = publishYear;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
```

```
        this.title = title;
    }
    public int getPublishYear() {
        return publishYear;
    }
    public void setPublishYear(int publishYear) {
        this.publishYear = publishYear;
    }
    @Override
    public String toString() {
        return "Book [title=" + title + ", publishYear=" + publishYear + "]";
    }

    public static void main(String[] args) {
        HashSet<Book> bookSet = new HashSet<>();
        LinkedHashSet<Book> bookLHS = new LinkedHashSet<>();
        TreeSet<Book> bookTS = new TreeSet<>();

        bookSet.add(new Book("Assassination Classroom", 2009));
        bookSet.add(new Book("Mushoku Tensei", 2011));
        bookSet.add(new Book("Distractible", 2020));
        System.out.println("HashSet");
        bookSet.forEach(System.out::println);
        System.out.println();

        bookLHS.add(new Book("Assassination Classroom", 2009));
        bookLHS.add(new Book("Mushoku Tensei", 2011));
        bookLHS.add(new Book("Distractible", 2020));
        System.out.println("LinkedHashSet");
        bookLHS.forEach(System.out::println);
        System.out.println();

        bookTS.add(new Book("Assassination Classroom", 2009));
        bookTS.add(new Book("Mushoku Tensei", 2011));
        bookTS.add(new Book("Distractible", 2020));
        System.out.println("TreeSet");
        bookTS.forEach(System.out::println);
        System.out.println();
    }
    @Override
    public int compareTo(Book o) {
        return this.getTitle().compareTo(o.getTitle());
    }
}
```

## Part 2: Performance Comparison – Set VS. List

### Exercise 2: Analyze the Performance Results

- (a) Which data structure was faster for the membership test (the contains method)? By how much?

The HashSet by around 1400 ms

- (b) Based on what you know about how a HashSet works (using hash codes), why do you think it is so much faster for searching?

Because there is only one key with the element that we are looking for, therefore it is faster to find it, whereas the ArrayList may contain multiple elements with the same key

- (c) If Sets are so much faster for searching, why would a programmer ever choose to use a List? (Hint: Think about what a List can do that a Set cannot).

Because the ArrayList iterates through the list in the right order, whereas the set will iterate and print elements in random order

## Part 3: The Map Interface – For Key-Value Pairs

```
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.TreeMap;

public class Product{
    private String name;
    private double price;
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    @Override
    public String toString() {
        return "Product [name=" + name + ", price=" + price + "]";
    }
    public static void main(String[] args) {
        HashMap<Integer, Product> hmList = new HashMap<>();
        LinkedHashMap<Integer, Product> lhmList = new LinkedHashMap<>();
        TreeMap<Integer, Product> tmList = new TreeMap<>();

        //HashMap
        hmList.put(67, new Product("Book", 21));
        hmList.put(69, new Product("Laptop", 1360));
        hmList.put(21, new Product("Pencil", 4.20));
        hmList.put(727, new Product("Phone", 700));

        //LinkedHashMap
        lhmList.put(67, new Product("Book", 21));
        lhmList.put(69, new Product("Laptop", 1360));
        lhmList.put(21, new Product("Pencil", 4.20));
```

```
lhmList.put(727, new Product("Phone", 700));

//TreeMap
tmList.put(67, new Product("Book", 21));
tmList.put(69, new Product("Laptop", 1360));
tmList.put(21, new Product("Pencil", 4.20));
tmList.put(727, new Product("Phone", 700));

System.out.println(hmList.get(67));

//prints the products randomly
System.out.println("\nHashMap");
for(Map.Entry<Integer, Product> entry : hmList.entrySet()){
    System.out.println("ID: " + entry.getKey() + " " + entry.toString());
}

//prints the product based on insertion order
System.out.println("\nLinkedHashMap");
for(Map.Entry<Integer, Product> entry : lhmList.entrySet()){
    System.out.println("ID: " + entry.getKey() + " " + entry.toString());
}

//prints the product based on the key
System.out.println("\nTreeMap");
for(Map.Entry<Integer, Product> entry : tmList.entrySet()){
    System.out.println("ID: " + entry.getKey() + " " + entry.toString());
}
// System.out.println(tmList);

}
```

## Part 4: Practical Application – Aggregating and Grouping Data

```
import java.util.ArrayList;
import java.util.Map;
import java.util.TreeMap;

public class Transaction{
    private String category;
    private double amount;
    public Transaction(String category, double amount) {
        this.category = category;
        this.amount = amount;
    }
    public String getCategory() {
        return category;
    }
    public double getAmount() {
        return amount;
    }

    public static void main(String[] args) {
        ArrayList<Transaction> list = new ArrayList<>();
        list.add(new Transaction("Groceries", 55.75));
        list.add(new Transaction("Transport", 22.50));
        list.add(new Transaction("Utilities", 120.00));
        list.add(new Transaction("Groceries", 34.25));

        TreeMap<String, Double> tmList = new TreeMap<>();

        for(Transaction t : list){
            System.out.println("Category: " + t.getCategory() + " Amount: " +
t.getAmount());

            if(!tmList.containsKey(t.getCategory())){//If the category is NOT in
the map: This is the first expense for this category. put the category and its
amount into the map.
                tmList.put(t.getCategory(), t.getAmount());
            }
            else if(tmList.containsKey(t.getCategory())){
                tmList.put(t.getCategory(), tmList.get(t.getCategory()) +
t.getAmount());
            }
        }
    }
}
```

```
    }
    System.out.println("\nFinal Summary");
    for (Map.Entry<String, Double> t : tmList.entrySet()) {
        System.out.println("Category: " + t.getKey() + " Amount: " +
t.getValue());
    }
}
```