```
import java.util.*;

// ===================================================
// 1️⃣ Basic Collection Example (Add & Remove)
// ===================================================
class CollectionDemo {
    public static void main(String[] args) {
        // Creating a Collection (ArrayList by default)
        Collection a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);

        System.out.println(a); // Prints entire collection

        // Removing an element
        a.remove(123);
        System.out.println(a);
    }
}

// ===================================================
// 2️⃣ Merging Two Collections (addAll)
// ===================================================
class CollectionDemo1 {
    public static void main(String[] args) {
        Collection a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);
        System.out.println(a);

        Collection a1 = new ArrayList();
        a1.add(1234);
        a1.add("Raviyadav");
        a1.add(123.45456);
        a1.add("Raviyadav");
        System.out.println(a1);

        // Adding all elements of a1 into a
        a.addAll(a1);
        System.out.println(a);   // Both lists merged
        System.out.println(a1);  // a1 unchanged
    }
}

// ===================================================
// 3️⃣ Add and Remove All Elements
// ===================================================
class CollectionDemo2 {
    public static void main(String[] args) {
        Collection a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);
        System.out.println(a);

        Collection a1 = new ArrayList();
        a1.add(1234);
        a1.add("Raviyadav");
        a1.add(123.45456);
        a1.add("Raviyadav");
        System.out.println(a1);

        System.out.println("-------------");
        // Add all elements of a1 to a
        a.addAll(a1);
        System.out.println(a);
        System.out.println(a1);

        System.out.println("-------------");
        // Remove all elements of a1 from a
        a.removeAll(a1);
        System.out.println(a);
        System.out.println(a1);
    }
}

// ===================================================
// 4️⃣ Retaining Common Elements (retainAll)
// ===================================================
class CollectionDemo3 {
    public static void main(String[] args) {
        Collection a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);
        System.out.println(a);

        Collection a1 = new ArrayList();
        a1.add(1234);
        a1.add("Raviyadav");
        a1.add(123.45456);
        a1.add("Raviyadav");
        a1.add(123.45);
        System.out.println(a1);

        System.out.println("-------------");
        // Retains only common elements
        a.retainAll(a1);
        System.out.println(a);
        System.out.println(a1);
```

```java
    }
}

// ====================================================
// 5️⃣ For-Each Loop Traversal
// ====================================================
class CollectionDemo4 {
    public static void main(String[] args) {
        Collection a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);

        System.out.println(a);

        // Traversing using for-each loop
        for (Object obj : a) {
            System.out.println(obj);
        }
    }
}

// ====================================================
// 6️⃣ Iterator Traversal
// ====================================================
class CollectionDemo5 {
    public static void main(String[] args) {
        Collection a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);

        System.out.println(a);

        // Traversing using Iterator
        Iterator it = a.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        System.out.println(a);
    }
}

// ====================================================
// 7️⃣ ListIterator Traversal (Forward + Backward)
// ====================================================
class CollectionDemo6 {
    public static void main(String[] args) {
        List a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);

        System.out.println(a);

        // Traversing with ListIterator
        ListIterator lit = a.listIterator();

        // Forward direction
        while (lit.hasNext()) {
            System.out.println(lit.next());
        }

        // Backward direction
        while (lit.hasPrevious()) {
            System.out.println(lit.previous());
        }

        System.out.println(a);
    }
}

// ==================================================
// 8️⃣ ListIterator with Modification (set)
// ==================================================
class CollectionDemo7 {
    public static void main(String[] args) {
        List a = new ArrayList();
        a.add(123);
        a.add("ragini yadav");
        a.add(123.45);
        a.add(123.45);

        System.out.println(a);

        // Access element by index
        Object obj = a.get(1);
        System.out.println("Element at index 1: " + obj);

        // Traversing with ListIterator
        ListIterator lit = a.listIterator();

        // Forward direction traversal
        while (lit.hasNext()) {
            Object obj1 = lit.next();
            System.out.println(obj1);

            // Replacing current element with "111"
            lit.set("111");
        }
        System.out.println();

        // Backward direction traversal
        while (lit.hasPrevious()) {
            System.out.println(lit.previous());
        }

        // Final modified list
        System.out.println(a);
    }
}

===================================================================================



package com.coll;

import java.util.*;
import java.util.Map.Entry;

// ===================================================
// 1️⃣ HashSet Demo → No Duplicates, Unordered
// ===================================================
class CollectionDemo9 {
    public static void main(String[] args) {
        // HashSet → does not allow duplicates, order not guaranteed
        Set a = new HashSet();
        a.add(123);
        a.add("Rama");
        a.add(1234.678);
        a.add("Rama"); // duplicate → ignored

        System.out.println(a); // prints unique elements
    }
}

// ===================================================
// 2️⃣ TreeSet Demo → Sorted Set, No Duplicates
// ===================================================
class CollectionDemo11 {
    public static void main(String[] args) {
        // TreeSet → sorts elements in ascending order, removes duplicates
        Set<Integer> a = new TreeSet<>();
        a.add(123);
        a.add(111);
        a.add(123); // duplicate → ignored
        a.add(456);
        a.add(56);

        System.out.println(a); // Sorted: [56, 111, 123, 456]
    }
}

// ===================================================
// 3️⃣ HashMap Demo (entrySet with Iterator + For-each)
// ===================================================
class CollectionDemo10 {
    public static void main(String[] args) {
        // HashMap stores key-value pairs
        HashMap<String, Object> a = new HashMap<>();
        a.put("Rollno", 123);
        a.put("Name", "Rama");
        a.put("Fee", 123.56);

        System.out.println(a);

        // Get all entries as Set
        Set<Entry<String, Object>> s = a.entrySet();

        // Traverse with Iterator
        Iterator<Entry<String, Object>> it = s.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        // Traverse with for-each loop
        for (Entry<String, Object> a1 : s) {
            System.out.println(a1);
        }
    }
}

// =====================================================
// 4️⃣ HashMap Demo (keySet + Iterator)
// =====================================================
class CollectionDemo12 {
    public static void main(String[] args) {
        HashMap<String, Object> a = new HashMap<>();
        a.put("Rollno", 123);
        a.put("Name", "Rama");
        a.put("Fee", 123.56);
        a.put(" ", 123.11);
        a.put("   ", 1111.11);
        a.put("Rollno", 555); // overwrites old "Rollno"
        a.put("   ", 123);    // overwrites old "   "
        a.put("Name1", " ");
        a.put("Name2", " ");
        a.put("Name3", " ");

        System.out.println(a);

        // Get value by key
        Object obj = a.get("Fee");
        System.out.println("Fee: " + obj);

        // Get all keys
        Set<String> keys = a.keySet();

        // Traverse keys
        Iterator<String> it = keys.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}

// ===================================================
// 5️⃣ HashMap Demo (values() Collection)
// ===================================================
class CollectionDemo13 {
    public static void main(String[] args) {
        HashMap<String, Object> a = new HashMap<>();
        a.put("Rollno", 123);
        a.put("Name", "Rama");
        a.put("Fee", 123.56);

        System.out.println(a);

        // Get only values
        Collection<Object> col = a.values();

        // Traverse values
        Iterator<Object> it = col.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```