Main.java

Share Run

Output

```java
import java.util.LinkedList;
public class LettersQueue{
    private LinkedList<String> lettersQ;
    public LettersQueue(){
        lettersQ=new LinkedList<>();
    }
    public void enqueue(String letter){
        lettersQ.add(letter);
    }
    public String dequeue(){
        return lettersQ.poll();
    }
    public boolean isEmpty(){
        return lettersQ.isEmpty();
    }
    public int size(){
        return lettersQ.size();
     }
    public static void main(String[] args){
         LettersQueue queue=new LettersQueue();
         queue.enqueue("A");
         queue.enqueue("B");
         queue.enqueue("C");
         queue.enqueue("D");
         System.out.println("Queue size: "+queue.size());
         while (!queue.isEmpty()) {
            System.out.println("Dequeued: "+queue.dequeue());
```

```
java -cp /tmp/wts7Xm6sjg/LettersQueue
Queue size: 4
Dequeued: A
Dequeued: B
Dequeued: C
Dequeued: D
Queue size after dequeue: 0

=== Code Execution Successful ===
```

Main.java

Share Run

Output

```
    }
    public void enqueue(String letter){
        lettersQ.add(letter);
    }
    public String dequeue(){
        return lettersQ.poll();
    }
    public boolean isEmpty(){
        return lettersQ.isEmpty();
    }
    public int size(){
        return lettersQ.size();
     }
    public static void main(String[] args){
         LettersQueue queue=new LettersQueue();
         queue.enqueue("A");
         queue.enqueue("B");
         queue.enqueue("C");
         queue.enqueue("D");
         System.out.println("Queue size: "+queue.size());
         while (!queue.isEmpty()) {
            System.out.println("Dequeued: "+queue.dequeue());
         }
         System.out.println("Queue size after dequeue: "+queue.size());
    }
}
```

```
java -cp /tmp/wts7Xm6sjg/LettersQueue
Queue size: 4
Dequeued: A
Dequeued: B
Dequeued: C
Dequeued: D
Queue size after dequeue: 0

=== Code Execution Successful ===
```

Main.java

Share Run

Output

```java
import java.util.LinkedList;
class Stack{
    private LinkedList<Integer> list;
    public Stack(){
        list=new LinkedList<>();
    }
    public void push(int value){
        list.addLast(value);
    }
    public int pop(){
        if (list.isEmpty()){
            throw new IllegalStateException("Stack is empty");
        }
        return list.removeLast();
    }
    public int peek(){
        if (list.isEmpty()){
            throw new IllegalStateException("Stack is empty");
        }
        return list.getLast();
    }
public boolean isEmpty(){
        return list.isEmpty();
    }
    public int size(){
        return list.size();
    }
}
public class llstack{
    public static void main(String[] args){
        Stack stack=new Stack();
        stack.push(10);
        stack.push(20);
        stack.push(30);
        System.out.println("Stack size: "+stack.size());
        System.out.println("Top element: "+stack.peek());
        System.out.println("Popped element: "+stack.pop());
        System.out.println("Stack size after pop: "+stack.size());
    }
}
```

```
java -cp /tmp/z7eBrPOyfv/llstack
Stack size: 3
Top element: 30
Popped element: 30
Stack size after pop: 2

=== Code Execution Successful ===
```

Main.java

Share Run

Output

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
class Mobile{
    private String name;
    private double price;
    private int quantity;
    public Mobile(String name, double price, int quantity){
        this.name=name;
        this.price=price;
        this.quantity=quantity;
    }
    public String getName(){
        return name;
    }
    public double getPrice(){
        return price;
    }
    public int getQuantity(){
        return quantity;
    }
    public String toString(){
        return "Mobile{" +
                "name='" + name + '\'' +
                ", price=" + price +
                ", quantity=" + quantity +
                '}';
    }
    public static void main(String[] args){
        List<Mobile> mobiles=new ArrayList<>();
        mobiles.add(new Mobile("IPhone", 999.99, 10));
        mobiles.add(new Mobile("Samsung Galaxy", 799.99, 15));
        mobiles.add(new Mobile("Google Pixel", 699.99, 12));
        Collections.sort(mobiles, (m1, m2) -> m1.getName().compareTo(m2.getName()));
        System.out.println("Sorted by name ");
        mobiles.forEach(System.out::println);
        Collections.sort(mobiles, (m1, m2) -> Double.compare(m1.getPrice(), m2.getPrice()));
        System.out.println("\nSorted by price:");
        mobiles.forEach(System.out::println);
        Collections.sort(mobiles, (m1, m2) -> Integer.compare(m1.getQuantity(), m2.getQuantity()));
        System.out.println("\nSorted by quantity:");
        mobiles.forEach(System.out::println);
    }
}
```

```
java -cp /tmp/gyIwyCiDeF/Mobile
Sorted by name:
Mobile{name='Google Pixel', price=699.99, quantity=12}
Mobile{name='Samsung Galaxy', price=799.99, quantity=15}
Mobile{name='IPhone', price=999.99, quantity=10}

Sorted by price:
Mobile{name='Google Pixel', price=699.99, quantity=12}
Mobile{name='Samsung Galaxy', price=799.99, quantity=15}
Mobile{name='IPhone', price=999.99, quantity=10}

Sorted by quantity:
Mobile{name='IPhone', price=999.99, quantity=10}
Mobile{name='Google Pixel', price=699.99, quantity=12}
Mobile{name='Samsung Galaxy', price=799.99, quantity=15}

=== Code Execution Successful ===
```

Main.java

Share Run

```java
import java.util.HashMap;
import java.util.Map;
public class HashMapExample{
    public static void main(String[] args){
        Map<String, String> fruitBowl=new HashMap<>();
        fruitBowl.put("Apple", "Red");
        fruitBowl.put("Banana", "Yellow");
        fruitBowl.put("Grapes", "Green");
        fruitBowl.put("Orange", "Orange");
        System.out.println("Contents of the fruitBowl:");
        for (Map.Entry<String, String> entry : fruitBowl.entrySet()) {
            System.out.println("Fruit: " + entry.getKey() + ", Color: " + entry.getValue());
        }
    }
}
```

Output

```
java -cp /tmp/3yJ142Xzbg/HashMapExample
Contents of the fruitBowl:
Fruit: Apple, Color: Red
Fruit: Grapes, Color: Green
Fruit: Orange, Color: Orange
Fruit: Banana, Color: Yellow

=== Code Execution Successful ===
```