Ejercicio U7_B9_E1:

En este caso más sencillo y legible es el for mejorado

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;


public class App {

    public static void main(String[] args) {
        Map<String,Double> hm= new HashMap<>();
        // repasando autoboxing. La sentencia comentada tiene problemas con autoboxing
        //El autoboxing  permite pasar double a Double pero no int a Double. 1500 es int
        //hm.put("Elías", 1500);
        hm.put("Elías", 1500.0);
        hm.put("Román", 1900.0);
        hm.put("Telma", 2400.0);

        Set<String> conjuntoDeLlaves=hm.keySet();
        System.out.println("recorrer con for mejorado");
        for(String s:conjuntoDeLlaves)
            System.out.println(s+ "  "+hm.get(s));

        System.out.println("recorrer con while e iterador");
        Iterator<String> it=conjuntoDeLlaves.iterator();
        while(it.hasNext()){
            String s=it.next();
            System.out.println(s+ "  "+hm.get(s));
        }
        System.out.println("recorrer con for e iterador");
        for(Iterator<String> it2=conjuntoDeLlaves.iterator();it2.hasNext();){
            String s=it2.next();
            System.out.println(s+ "  "+hm.get(s));
        }

    }
}
```

Ejercicio U7_B9_E2:

```
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
class App {
     public static void main(String[] args) {
        List<Integer> a= new LinkedList<>();
        a.add(1);a.add(2);a.add(3);a.add(4);
        a.add(5);a.add(6);a.add(7);a.add(8);a.add(9);
        System.out.println(a);
        for(Iterator<Integer> it= a.iterator();it.hasNext();){
            if(it.next()%2==0){
                it.remove();
            }
        }
        System.out.println(a);
     }
}
```

Ejercicio U7_B9_E3:

```
import java.awt.Point;
import java.util.ArrayList;
import java.util.Iterator;

class App{
    public static void main(String[] args) {
```

```
        ArrayList<Point> al= new ArrayList<>();

        al.add(new Point(2,56));
        al.add(new Point(22,56));
        al.add(new Point(32,5));
        al.add(new Point(99,99));
        System.out.println(al);
        for(Iterator<Point>  it= al.iterator();it.hasNext();){
            Point p=it.next();
            if(p.x < 15 || p.y<15){
                it.remove();
            }
        }
        System.out.println(al);
    }
}
```

Ejercicio U7_B9_E4:

SOLUCIÓN CON ARRAY

```
import java.util.ArrayList;

class Contacto {
    String nombre;
    String telefono;

    public Contacto(String nombre, String telefono) {
        this.nombre = nombre;
        this.telefono = telefono;
    }

    @Override
    public String toString() {
        return "nombre=" + nombre + ", teléfono="+telefono ;
    }

}

interface  MiIterator {
    boolean hasNext();
    Contacto next();
}
 class IteradorDeAgenda  implements MiIterator{
    Contacto[] contactos;
    int posSiguiente;
    IteradorDeAgenda(Agenda a){
        //suponemos que a.contactos no es null. Como mucho puede ser vacio.
        this.contactos=a.contactos;
        posSiguiente=0;
    }
    @Override
    public boolean hasNext() {
        //el && perezoso hace que no provoque excepcion el segundo operando
        return posSiguiente<contactos.length && contactos[posSiguiente]!=null;
    }

    @Override
    public Contacto next() {
       return contactos[posSiguiente++];
    }

}

 class Agenda {
    String propietario;
    Contacto[]  contactos;

    public Agenda(String propietario){
        contactos=new Contacto[100];
        this.propietario=propietario;
    }
    public String getPropietario(){
```

```
        return propietario;
    }
    public void add(String nombre, String telefono){
        for(int i=0; i<contactos.length; i++){
            if(contactos[i]==null){
                contactos[i]=new Contacto(nombre, telefono);
                break;
            }
        }
    }

    public  MiIterator crearIterador(){
         return new IteradorDeAgenda(this);
     }

}


class Cliente{
    public static void main(String[] args){
       Agenda agendaChuchi= new Agenda("Chuchi");
       agendaChuchi.add("eluno", "11111");
       agendaChuchi.add("dosi", "22222");
       agendaChuchi.add("tresi", "333333");
       MiIterator  it= agendaChuchi.crearIterador();
       System.out.println("Agenda de "+ agendaChuchi.getPropietario());
       while(it.hasNext()){
           System.out.println(it.next());
       }
    }
}
```

Ejercicio U7_B9_E5: Code runner