Teste POO 2016
Parte I

1. a)

```
public ListaEleitoral(String partido, Collection<Candidato> candi){
    this.partidoPolitico = partido;
    this.porEleger = new ArrayList<Candidato>(candi); //lixo
                    ou
    this.porEleger = candi.stream().collect(Collector.toList());
}
```

b)

```
public Candidato aEleger(){
    Candidato c = porEleger.get(0);
    porEleger.remove(0);

    return c;
}
```

c)

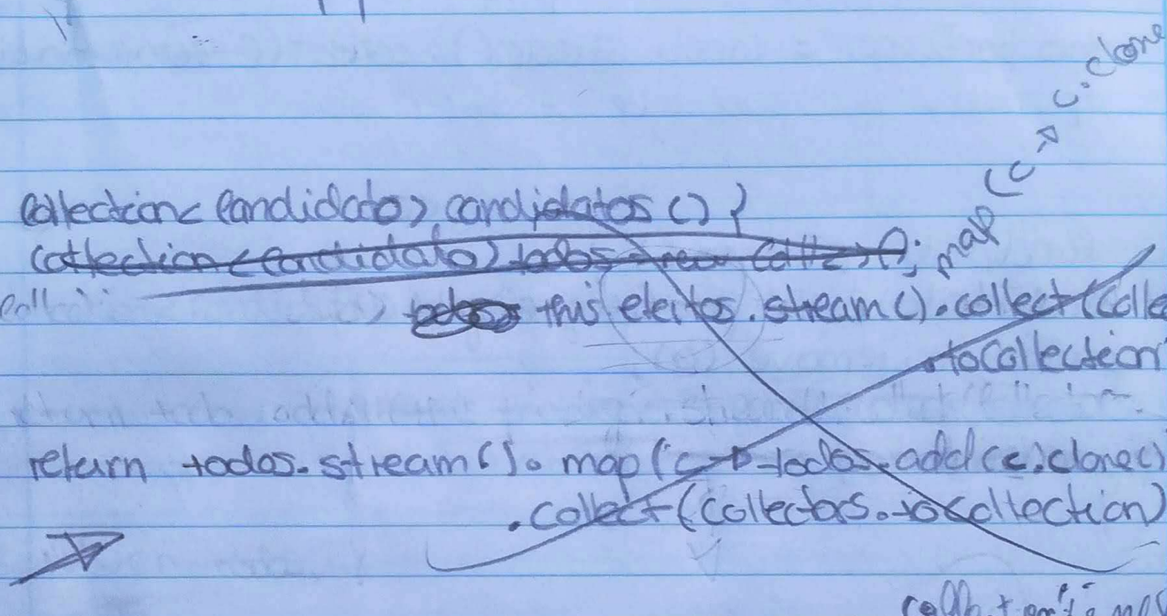
```
public void elege() throws NoCandidatosException{
    if (porEleger.isEmpty()) throw new NCE("Não há candidatos
    Candidato c = porEleger.get(0);
    this.porEleger.remove(0);
    this.eleitos.add(c);
}
```

d) public void elege (int n)
if (porEleger.size() >= n) {

for (Candidato c : porEleger && n > 0) {
this.elegidos.add(c)
This.porEleger.remove(c);
n--;
}}

e)
public Collection<Candidato> candidatos () {
~~Collection<Candidato> todos = new Coll~~;
~~todos~~ this.elegidos.stream().collect(Collectors
.toCollection());

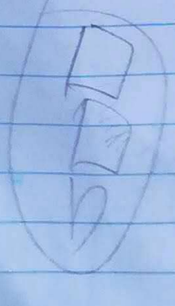
return todos.stream().map(c -> todos.add(c.clone()))
.collect(Collectors.toCollection);



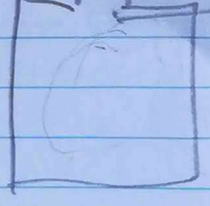
ver map

TreeSet<Candidato> todos = new
this.elegidos.stream().~~forEach~~(e->~~todos.add~~
todos.add(e))

Polym



return todos super.emtd



(TreeSet :: new)

2. ▷ Ver se está certo

a) public TreeSet<Candidato>> eleitos(){

~~TreeSet<Candidato> cand = new TreeSet<Candidato>();~~

return this.eleitos.stream().map(c -> c.clone()).
collect(Collectors.toSet());

}

b) public class ComparatorNome implements Comparator<Candidato>{

public int compara(Candidato c1, Candidato c2){
return c1.getNome().compareTo(c2.getNome);
}
}

c) public class ComparatorIdadeNome implements Comparator<Candidato>{

public int compara(Candidato c1, Candidato c2){
if ((c1.getIdade - c2.getIdade) == 0)
return c1.getNome().compareTo(c2.getNome());

return c1.getIdade() - c2.getIdade();
}

3)

a) private class Nodo {
    private Candidato c;
    private Nodo prox;
}

public class LCandidato {
    private Nodo prox;
    private int tamanho;
}

a) public LCandidato() {
    this.prox = new Nodo();
}

b) public int size() {
    return this.tamanho;
}

c) public void add(Candidato c) {
    Nodo n = new Nodo(c, null);

    if (this.prox == null) {
        this.prox = n;
        this.tamanho++; }

    Nodo n1 = ~~this.prox; new Nodo();~~

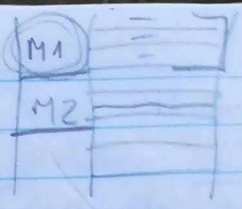
for(; n1; n1=n1.getProx());
    n1 = n;

    while(n1) {
        n1 = n1.getProx();
    }
    n1 = n;
}

d)
```
public Candidato get(int i) throws CandidatoException {
    if (i > this.tamanho) throw new CandidatoException
                                ("Não existe!");
    Nodo n = new Nodo();
    while (i > 0) {
        n = n.getProx();
        i--;
    }
    return n.getCandidato();
}
```

e)
```
public boolean equals(Object o) {
    if (this == o) return true;
    if ((o == null) || (this.getClass() != o.getClass()))
        return false;
    LLCandidato c = (LLCandidato) o;
    return this.prox.equals(c.getProx()) &&
           this.tamanho == c.getTamanho();
```

Parte IV



4.

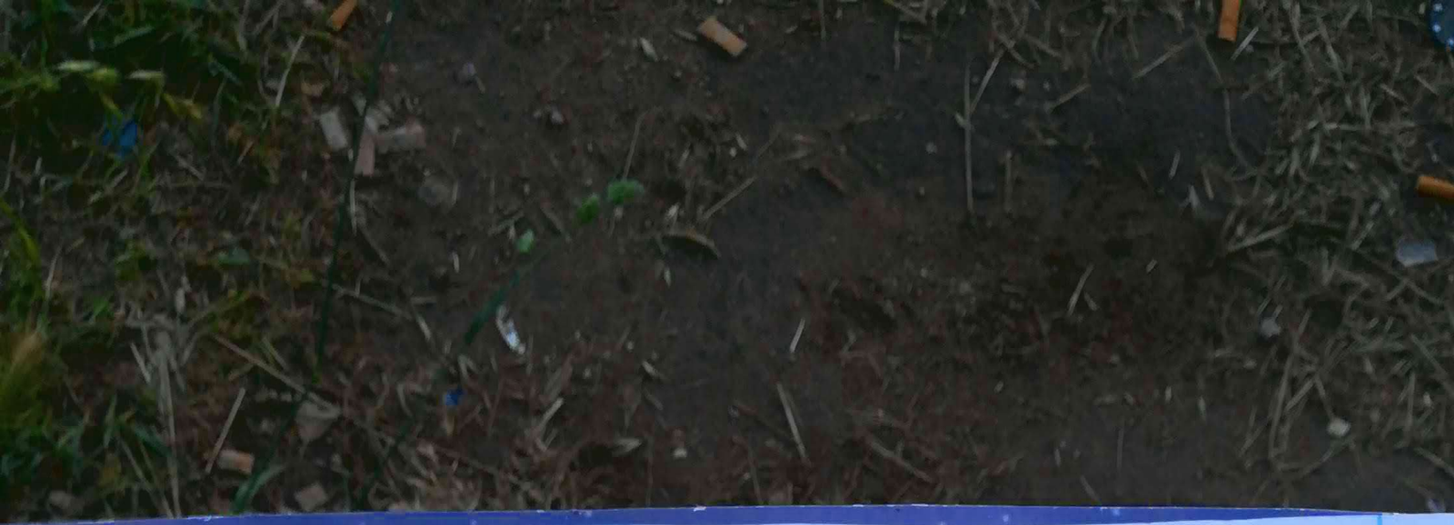
```
public class ParqueComReservados extends Parque implements Parque
    private Map<String, Set<String>> reservados;
                        K       ,   V

public void entra (String cartao, String Matrícula) throws ... {
    try {
        super.entra(cartao, matricula);
    }
    catch (SemPermissaoException exc) {

        if (reservados.containsKey(matricula)) {
        this.reservados.get(matricula).add(cartao);
        }
        else {
            TreeSet<String> novo = new TreeSet<String>();
            this.reservados.put(matricula, novo.add(cartao));
            }

        throw new SemPermissaoException(matricula);
        }
        }

Map<String, Set<String>> getReservas () {
            return this.reservados;
            }
```



Map<K, V>

5. public double getKmsTotal(){
    double total = 0;

    for(Empregado e : empregados.values()){
        if(e instanceof Motorista)
            total += (double) e.getNrKms();
    }}

Teste 2015
Parte II

2.

a) public class VOR {

    private Map<String, Equipa> listaEquipas;
    // private Map<String, Map<String, Barco>> barcosEmAgua;

    // construtores? gets? sets?

b) Barco getBarco(String idEquipa, String idBarco) throws ...

    if(listaEquipas.get(idEquipa).getBarcos().get(idBarco) ==
        throw InvalidBoatException("barco não existe");

    return listaEquipas.get(idEquipa).getBarcos().get(idBarco