Teste

Ex1

```
public boolean ativa () {
    GregorianCalendar hoje = new GregorianCalendar ();
    return (hoje.after (this.inicio) && hoje.before (this.fim) && !terminada
}
```

Ex2

a)

```
public void addTarefa (Tarefa t) {
    this.tarefas.add(t.clone ());
}
```

b)

```
public Iterator <Tarefa> tarefasAtivas (Comparator <Tarefa> ct) {
    TreeSet <Tarefa> aux = new TreeSet <Tarefa> (ct);
    for (Tarefa t : this.tarefas)
        if (t.ativa ()
            aux.add. iterator ();
        return aux.iterator ();
```

c)

```
public class ComparatorTt implements Comparator <Tarefa> {
    public int compare (Tarefa t1, Tarefa 2) {
        if (t1.getPrioridade () > t2.getPrioridade ())
            return 1
        else { if (          <           )
            return -1;
        else {
            comparar as datas de termino
```

```
public Set<Tarefa> tarefasEntreDatas (GregorianCalendar di, Greg... df) {

HashSet out TreeSet <Tarefa> aux = new TreeSet <tarefa> (passar o comparad)

        for (Tarefa t : this.tarefas) {
            if ((t.getfim()).after(di) && (t.getfim()).before(df))
                aux.add(t.clone());
        }
        return aux;
        }
```

Ex4

```
a) public double valorTodosContratos() {
            double res = 0,0;
        for (Viculo v : this.viculo.values())
            res += v.ValorContratoVeiculo();
        return res;
        }
```

```
metodo em Veiculo
  public double ValorContratadoVeiculo() {
    double res = 0.0;
    for (Contratual c : this.servicoContratados)
        res += custoPorContrato(c);
        }
    return res;
```

```
b)
    Map<String, ArrayList<Viculo>>    VeiculoPorCliente(), {
      TreeMap <String, ArrayList <Viculo>> res = new TreeMap <String, ArrayList <Veiculo>>

        for (Veiculo v : this.Viajas.values())
            for (Contratual c ; v.getServicosContratados()) {
                String nomeCliente = c.getCliente();
                if ( res.containsKey(nomeCliente))
                    (res.get(nomeCliente)).add(v.clone());
                else
                    ArrayList<Veiculo> novo = new ArrayList<Veiculo>();
                    novo.add(v.clone()); res.put(nomeCliente, novo); }}
                return res;
```

Ex4

a)

```
public String VeiculoMaisKms(){
    String mat;
    double max=0.0;
    for (Veiculo v: this.veiculos.values()){
        if (v.kmsVeiculo() > max){
            mat = v.getMatricula();
            max = v.getKmsVeiculo();
        }
    }
    return mat;
}
```

```
metodo Veiculo
public double kmsVeiculo(){
    double res=0.0;
    for(Atividade c:this.atividades)
        res += c.getDistancia();
    return res;
```

//

Ex5

```
public Class MyMap {
    private ArrayList<String> keys;
    private ArrayList<Veiculo> values;
```

ou

```
public Class MyMap2 {
    private HashSet<Entrada> entradas;
    {
public Class Entrada{
    private String key;
    private Veiculo values;
```

Ex6

```
void GravaMyMap (String filename, double valorref) throws IOException {
    ObjectOutStream oos = new ObjectOutStream(new FileOutputStream(filename
    for (Veiculo v: this.values)
        if(v.valorEntradas() > valorref)
            oos.writeObject(v);
    oos.flush();
    oos.close();
}
```