POO - revisões exame

Resolução do teste 2017
PARTE 1

1)
```
public class Faixa {
    private String nome;
    private String autor;
    private double duracao;
    private int classificacao
    private ArrayList<String> letra; //letra da música
    private int numeroVezesTocada; //numero de vezes que foi tocada
    private LocalDateTime // regista quando foi tocada a última
                          // vez
...
}
```

a)
```
public Faixa (Faixa f) {
    nome = f.getNome();
    autor = f.getAutor();
    duracao = f.getDuracao();
    duracao = f
    classificacao = f.getClassificacao();
    letra = f.getLetra();
    numeroVezesTocada = f.getNumeroVezes();
    ultimaVez = f.getUltimaVez();
```

b)
```
boolean equals (Object o) {
    if (this == o) return true
    if (o == null || this.getClass() != o.getClass())
        return false

    Faixa f = (Faixa) o;

    return (nome.equals(f.getNome()) &&
            autor.equals(f.getAutor()) &&
            duracao == f.getDuracao() &&
            classificacao == f.getClassificacao &&
            letra.equals(f.getLetra()) &&
            numeroVezesTocada == f.getNumeroVezes()) &&
            ultimaVez.equals(f.getUltimaVez());
```

c)
```
public int compareTo (Faixa f) {
    int n = f.getNumeroVezes();
    int res;
    if (this.numero == n)
        res = 0;
    else if (this.numero > n)
            res = 1;
        else res = -1;
    return res
}
```

d) public class CompasatoUltimaVez implements Comparator<Faixa>{

public int compare(Faixa f1, Faixa f2){
return f1.getUltimaVez().compareTo(f2.getUltimaVez());
}
}

// assumo que [tipo] tem compareTo definido

2) public class PlayList {
private String nome;
private Map<String, List<Faixa>> musicas;
...

a) public List<Faixa> getFaixas(String autor) throws AutorInexistenteEx...
if(musicas.containsKey(autor)){
return musicas.get(autor).stream().map(Faixa::clone)
.collect(Collectors.toList());
}
else throw new AutorInexistenteException();

b) public double tempoTotal(String autor) throws AutorInexistenteEx...
if(musicas.containsKey(autor)){
return musicas.get(autor).stream()
.mapToDouble(Faixa::getDuracao)
.sum();
}
else throw new AutorInexistenteException();

PARTE 2

3) a) public List<Faixa> todasAsFaixas(){
List<Faixa> res = new ArrayList<>();
for(List<Faixa> lista : musicas.values()){
for(Faixa f : lista){
res.add(f.clone());
}
}
return res;
}

b) public Map<Integer, List<Faixa>> faixasPorClass() {
List<Faixa> res = new ArrayList<>();
~~[illegible]~~
for (List<Faixa> lista : musicas.values()) {
for (Faixa f : lista) {
res.add(f.clone());
}
}
return res.stream().collect(Collectors.groupingBy(Faixa::
getClassif
icacao));

c) public class Faixa implements Playable {
...
public void play() {
System.audio.play(letra);
}
}

4) a) public class MusicaComVideo extends Faixa {
private String video;
...
}

b) public MusicaComVideo(String nome, String autor,
double duracao, (...), String video) {
super(nome, autor, duracao, (...));
this.video = video;
}

c) ... implements Playable {
public void play() {
super.play();
System.audio.play(video);
}

PARTE 3

5) public abstract class Hotel implements Comparable <Hotel> {
    private String codigo;
    private String nome;
    private String localidade;
    private double precoBaseQuarto;
    private int numeroQuartos;
    private int estrelas;

a) public class AgenciaViagens {
    private String nome;
    private Map <[illegible] Hotel, List<String>> alojadoHotel;
    // mapeia um hotel a uma lista de NIFs
    ...
}

b) public void escreveHotelTxt (String nomeficheiro) throws IOException {
    try {
        PrintWriter fich = new PrintWriter (nomeficheiro);
        fich.println ("..... Hotels Inc ....");
        fich.println (this.toString());
        fich.flush();
        fich.close();
    }
    catch (IOException e) {
        System.out.println (e.getMessage());
    }
}

c) public void gravarHotel (String fich) throws IOException, FileNotFoundException {
    try {
        FileOutputStream fos = new FileOutputStream (fich);
        ObjectOutputStream oos = new ObjectOutputStream (fos);
        oos.writeObject (this);
        oos.flush();
        oos.close();
    }
    catch (IOException | FileNotFoundException e) {
        System.out.println (e.getMessage());
    }
}

d) Ambas as classes tinham de implementar Serializable.