Relatório - Java

Laboratórios de Informática III

Duarte Duarte, Helder Gonçalves, Fábio Gomes

13-06-2013

Conteúdo

Introdução

O projecto de Java da disciplina de LI3 tem por objectivo fundamental ajudar à consolidação experimental dos conhecimentos teóricos e práticos adquiridos na disciplina de Programação Orientada pelos Objectos. Para isto, foi requisitada a execução de uma Rede de Autores de artigos científicos

API

Classe Artigo

Classe que é utilizada para a inserção de novos artigos na estrutura da Rede de Autores. É uma classe utilizada com efeito temporário mas extremamente necessária à manipulação da informação. Na classe Artigo encontra-se a lista dos nomes dos autores bem como o ano do artigo.

- **Variaveis de Instância**
  private List<String> autores
  private int numeroAutores

- **Construtores**
  public Artigo();
  protected Artigo(Artigo art);
  public Object clone();

- **Get's**
  public int getNumeroAutores();
  public String toString();
  public boolean equals(Object obj);
  public int compareTo(Object t);
  private boolean equalsListaAutores(List<String> other);
  public boolean isSingleAuthor();

public boolean isWrittenByN(int n);
public boolean isWrittenByLessThanN(int n);
public boolean isWrittenByMoreThanN(int n);
public List<String> getAutores();

- **Set's**
  public void setNumeroAutores(int numeroAutores);
  public boolean insereAutor(String autor);
  public void setAutores(List<String> autores);

- Função auxiliar
  public int hashCode();

Classe CoAutor

Classe que se encontra dentro da estrutura do Autor. Esta classe contém a informação necessária de um coAutor. Neste caso o nome desse CoAutor bem como o número de artigos que o autor e o coAutor têm em comum.

- **Variaveis de instância**
  private String nome;
  private int artigosComum;

- **Construtores**
  public CoAutor();
  public CoAutor(String nome);
  private CoAutor(CoAutor other);
  public Object clone();

- **Get's**
  public int compareTo(Object t);
  public boolean equals(Object obj);
  public String getNome();
  public int getArtigosComum();
  public String toString();

- **Set's**
  public void setNome(String nome);
  public void setArtigosComum(int artigosComum);
  public void addCoAutoria();

- **Função auxiliar**
  public int hashCode();

Classe Autor

Classe que se encontra dentro de um AutoresAno. Esta classe contém toda a informação conveniente de um autor. O seu nome, os seus coAutores, o número total de artigos que publicou em conjunto com outros autores e ainda o número de artigos que publicou a solo.

- **Variaveis de Instância**
  private String nome;
  private HashMap<String, CoAutor> coAutores;
  private int nArtigos;
  private int nArtigosSolo;

- **Construtores**
  public Autor();
  public Autor(String nome, int nArtigos, int nArtigosSolo, Collection<CoAutor> coAutores);

- **Get’s**
  public String getNome();
  public int compareTo(Autor t);
  public String toString();
  public boolean equals(Object obj);
  public Collection<CoAutor> getCoAutores();
  public List<String> getCoAutoresNome();
  public void setnArtigos(int nArtigos);
  public int getNumCoautorias();
  public int getnArtigosSolo();
  public int getNumeroCoAutores();
  public int totalArtigos();

- **Set’s**
  public void setNome(String nome);
  public void setCoAutores(Collection<CoAutor> coAutores);
  public void setnArtigosSolo(int nArtigosSolo);
  public final boolean addCoAutores(Collection<CoAutor> coautores);
  public void incrementaContadores(int nCoAutorias, int nArtigosSolo);

- **Função auxiliar**
  public int hashCode();

Classe AutoresAno

Esta classe encontra-se dentro da rede de Autores e possui toda a informação de um determinado ano. Todos os autores que escreveram nesse ano, número total de artigos publicados nesse ano, o total de Autores que publicaram nesse ano e ainda o ano em questão.

- **Variaveis de Instância**
  private Map<String, Autor> autoresAno;
  private int numeroArtigos;
  private int totalAutores;
  private int ano;

- **Construtores**
  public AutoresAno(int ano);
  private AutoresAno(AutoresAno aa);
  public Object clone();

- **Get’s**
  public int getAno();
  public int getNumeroArtigos();
  public int getTotalAutores();
  public String toString();
  public boolean equals(Object obj);

- **Set’s**
  public void setAno(int ano);
  public void setNumeroArtigos(int numeroArtigos);
  public void setTotalAutores(int totalAutores);
  public boolean insereAutor(Autor autor);
  public void incrementaArtigos();
  public void setArtigosAno(Map<String, Autor> autoresAno);

- **Função auxiliar**
  public int hashCode();

Classe RedeAutores

Esta classe possui toda a informação retirada de um ficheiro. Possui todos os autores organizados num Mapa de <Autores, Ano>. Possui o nome do ficheiro que contém a informação e o total de autores que a estrutura possui.

- **Variaveis de Instância**
  private String nomeFicheiro;
  private int totalAutores;
  private Map<Integer, AutoresAno> autores;

- **Construtores**
  public RedeAutores(); // Vazio
  protected RedeAutores(String nomeFicheiro, Map<Integer, AutoresAno> art); //Passados
  protected RedeAutores(RedeAutores ra); //Clone
  public Object clone();

- **Get's**
  public int getTotalAutores();
  public String getNomeFicheiro();
  public AutoresAno getAutoresAno(int ano);
  public Map<String, Autor> getAutoresAno();
  public String toString();

- **Set's**
  public void setNomeFicheiro(String nomeFicheiro);
  public void addAutoresArtigo(int numAutores); //Incrementa N ao total de artigos
  public boolean insereAutoresAno(int ano, AutoresAno autores); //Insere Autores

Classe Storage

Classe onde fica guardada toda a informação, é apenas o local onde se encontra toda a informação de forma a tornar o acesso a essa mesma de uma forma mais estruturada e mais facil.

- **Variaveis de instancia**
  private RedeAutores redeAutores;

- **Construtores**
  public RedeAutores getRedeAutores();

- **Get's**
  public AutoresAno getArtigosAno(int ano);
  public String getFileName();

- **Set's**
  public void setAutoresMap(RedeAutores ra);
  public boolean addArtigo(int ano, Artigo artigo);
  public void setFileName(String fileName);

Classe Statistics

Classe onde se encontram todas as perguntas do projecto, de forma a estar bem organizado tudo que é conveniente das perguntas encontra-se nesta classe.

- **Variaveis de Instância**
  private RedeAutores ra;
  private final Collection<Autor> conjuntoAutores;

- **Construtores**
  public Statistics(RedeAutores ra);

- **Get's**
  public String pergunta11();
  public String pergunta12();
  public String pergunta13();
  public String pergunta21a(int x, int anoIni, int anoFim);
  public String pergunta21b(int x, int anoIni, int anoFim);
  public String pergunta21d(int anoIni, int anoFim);
  public int pergunta22a();
  public String pergunta22b(int x, int anoIni, int anoFim);
  ---
  public Collection<CoAutor> pergunta21c(List<String> lista, int anoIni, int anoFim);
  private void avaliaAutor(Autor autor, List<TopXEntry> topX);
  private void avaliaCoAutoria(Autor autor, List<TopXYEntry> topX);
  public int numeroArtigosLidos();
  public int totalNomes();
  private int totalAutores();
  private List<Integer> artigosUnicoAutor();

- **Set's**
  public void setRedeAutores(RedeAutores ra);

Classe FileHandling

Classe que tem os acessos ao exterior. Sempre que é necessário sair do ambiente do programa, por exemplo, aceder a ficheiros e actividades do genero, encontra-se indexado a esta classe.

- **Variaveis de Instância**
  Não tem;

- **Construtores**
  private FileHandling();

- Write's
  public static int writeToFile(Map<Integer, List<Autor>> autores);
  public void save(Storage store);

- Read's
  public static Map<Integer, List<Autor>> readFromFile();
  public Storage load();
  public static List<String> leLinhasScanner(String fichName);

Classe Util

Classe que possui métodos uteis a varios niveis do projecto. Métodos que ajudam na resolução de algumas das perguntas e até na organização da informação.

- **Variaveis de Instância**
  Não tem;

- **Construtores**
  private Util();

  public static Collection<Autor> listaAutores(RedeAutores ra);
  public static Collection<Autor> conjuntoAutores(RedeAutores todosAutores);
  public static Collection<Autor> conjuntoAutores(RedeAutores todosAutores, int anoIni, int anoFim);
  public static Collection<Autor> everythingButAutor(String nome, Collection<Autor> col);
  public static Collection<CoAutor> everythingButCoautor(String nome, Collection<CoAutor> col);

Classe Factories

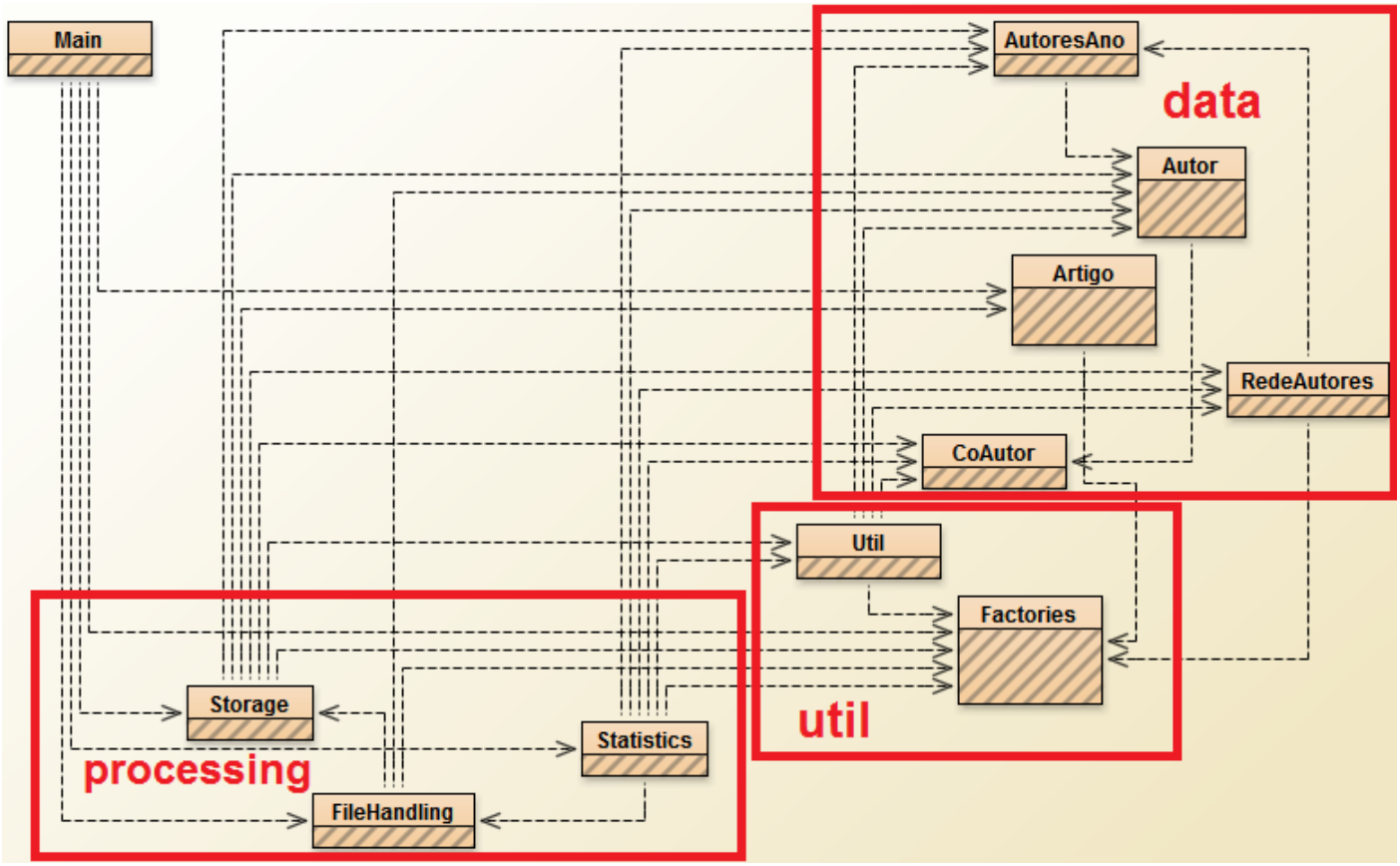
Classe que foi utilizada para facilitar os testes de estruturas. Sempre que foi necessário trocar uns tipos de Mapas por outros tipos de Mapas bastava alterar nesta classe e todo o projecto se alterava.

- **Variaveis de Instância**
  Não tem;

- **Funções auxiliares**
  public static <K, V> Map<K, V> giveMeMap();
  public static <V> Set<V> giveMeSet();
  public static <T> List<T> giveMeList();
  public static <V> Set<V> giveMeSet(Collection<V> vs);
  public static <T> List<T> giveMeList(Collection<T> ts);

Diagrama de Classes

Como é possivel observar pela figura que se segue existe uma dependencia de classes. As classes estão agrupadas em packages tal como indicam os rectângulos.



				tempos em segundos									
			Ficheiro	Alocação	P11	P12	P13	P21A	P21B	P21C	P21D	P22A	P22B	Total
TreeMap	LinkedHashSet	ArrayList	publicx	3,41696	0,22852	0,40485	0,00232	5,98595	0,58225	0,00183	0,07689	0,78218	3,71579	**15**
			publicx_x4	11,79159	0,19195	0,36048	0,00206	0,79214	0,77965	0,0022	0,08676	3,50176	5,08648	**23**
			publicx_x8	18,8237	0,20102	0,40325	0,00194	0,74421	0,72795	0,00181	0,07154	7,17743	4,03141	**33**
TreeMap	LinkedHashSet	Vector	publicx	3,99314	0,24238	0,38873	0,00187	5,86962	0,63828	0,00181	0,09345	0,7756	3,47565	**16**
			publicx_x4	11,89166	0,20831	0,34382	0,00245	0,45457	0,78323	0,00204	0,07655	3,60753	5,4646	**24**
			publicx_x8	19,75292	0,19638	0,65558	0,00179	0,45734	0,70803	0,01943	0,06486	6,33629	4,0529	**32**
TreeMap	HashSet	ArrayList	publicx	3,79494	0,21854	0,35888	0,00178	5,98443	0,54997	0,00181	0,07399	0,6587	3,74522	**14**
			publicx_x4	12,47185	0,54104	0,31325	0,00175	0,43409	0,90702	0,0018	0,07675	3,52073	5,44481	**24**
			publicx_x8	18,66095	0,19248	0,32135	0,00203	0,70452	0,83162	0,00177	0,0651	6,96748	3,73864	**31**
TreeMap	HashSet	Vector	publicx	3,86056	0,22887	0,41478	0,002	6,08893	0,5721	0,00181	0,07666	0,88437	4,48284	**16**
			publicx_x4	11,58289	0,20458	0,34526	0,00206	0,85213	0,9633	0,00189	0,10254	3,93568	5,86299	**24**
			publicx_x8	19,86162	0,19677	0,33295	0,00184	0,76637	0,80008	0,00184	0,07407	6,86045	3,65843	**33**
TreeMap	TreeSet	ArrayList	publicx	3,8856	0,22859	1,02553	0,00178	6,92202	0,64409	0,00175	0,06857	0,79143	4,20809	**18**
			publicx_x4	13,57101	0,23184	1,19517	0,00208	0,50529	0,95133	0,00186	0,06802	4,14512	5,95453	**28**
			publicx_x8	23,44296	0,20324	0,94276	0,00236	0,86546	0,92463	0,00175	0,06626	7,44101	4,31797	**39**
TreeMap	TreeSet	Vector	publicx	3,66869	0,23863	1,00691	0,00258	6,54992	0,56722	0,00182	0,06712	0,74973	3,98348	**17**
			publicx_x4	12,91605	0,19906	0,92914	0,00181	0,46431	0,9344	0,00234	0,06314	3,9367	5,66373	**26**
			publicx_x8	22,67383	0,22862	1,2395	0,00208	0,90682	0,87089	0,00227	0,08764	8,86363	4,24455	**42**
TreeMap	TreeSet	LinkedList	publicx	3,86969	0,25746	0,98514	0,00284	6,92858	0,57659	0,0018	0,06318	0,71146	3,95072	**18**
			publicx_x4	14,52326	0,22071	0,94767	0,00243	0,96697	0,6965	0,00219	0,06593	3,88895	4,54344	**26**
			publicx_x8	22,45358	0,22427	1,04766	0,00185	0,83956	0,79714	0,00178	0,07004	7,70435	4,56722	**38**

Estatística

Fizemos testes com várias combinações de Sets e Lists:

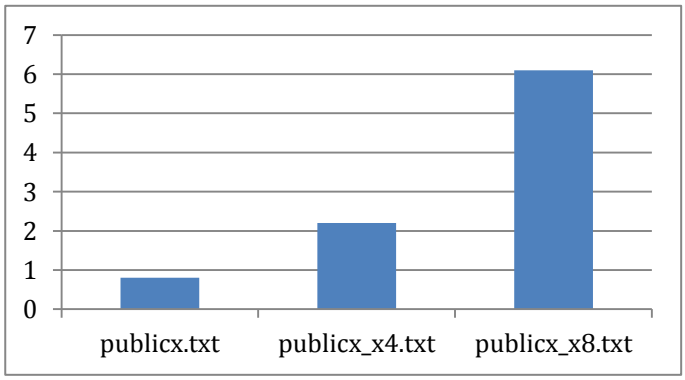
1. ***TreeMap LinkedHashSet ArrayList***
2. ***TreeMap LinkedHashSet Vector***
3. ***TreeMap HashSet ArrayList***
4. ***TreeMap HashSet Vector***
5. ***TreeMap TreeSet ArrayList***
6. ***TreeMap TreeSet Vector***
7. ***TreeMap TreeSet LinkedList***

Não podemos usar *HashMap* pois utilizamos *SortedMap* na classe *Statistics*.

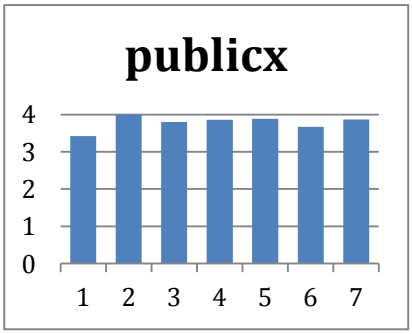
Depois de vários testes com medições de tempos conseguimos obter médias para a tabela da página anterior.

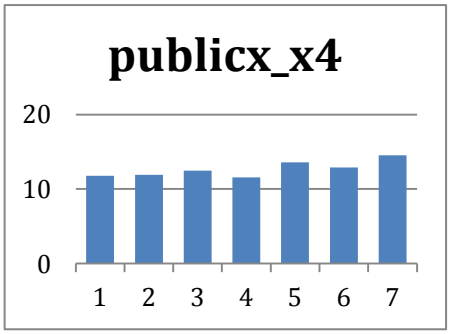
Tempos de Leitura (sem parsing) dos ficheiros em segundos:

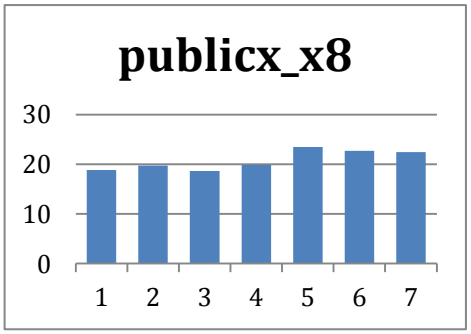
	tempo (s)
	Scanner
publicx.txt	0,8
publicx_x4.txt	2,2
publicx_x8.txt	6,1



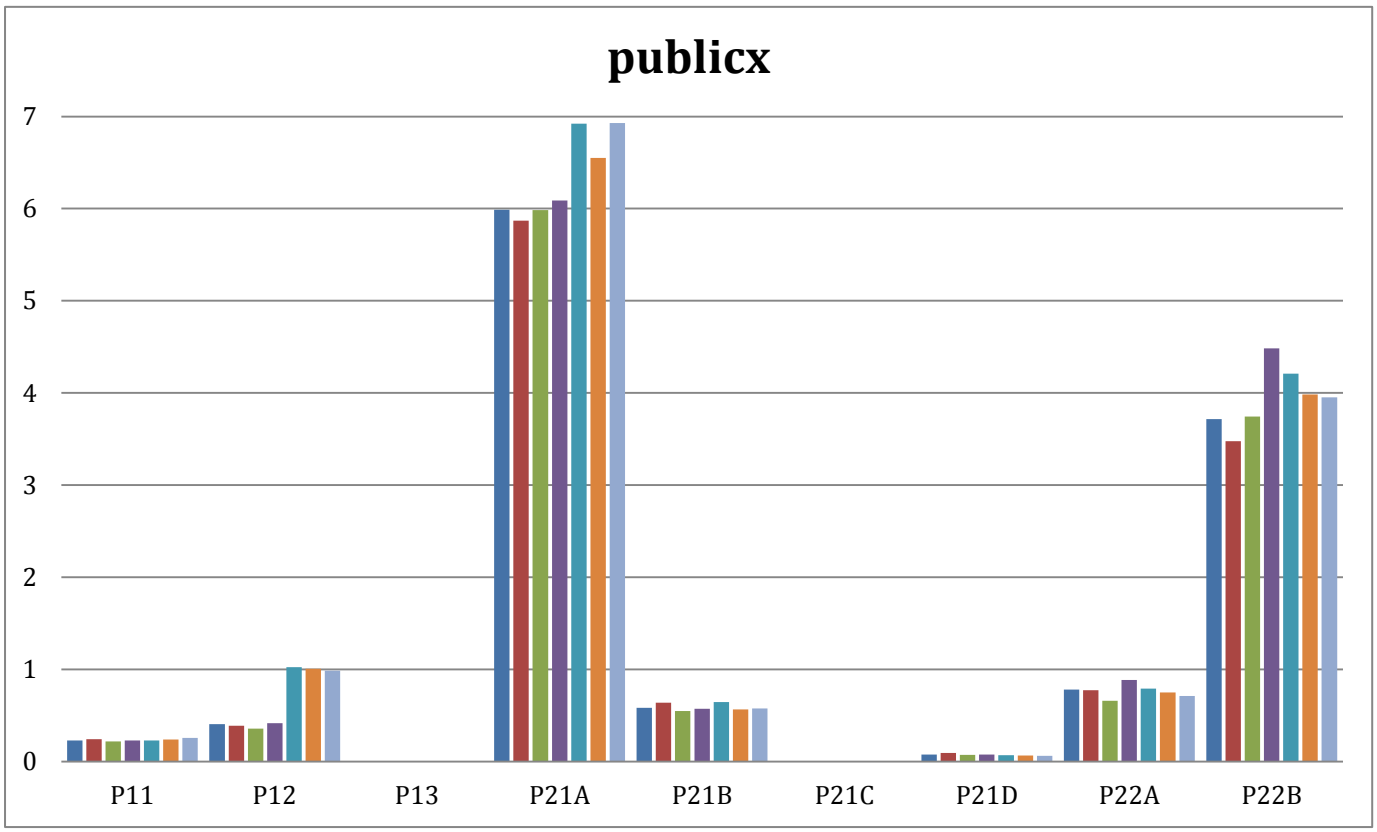
Tempos de Leitura com alocação dos ficheiros em segundos consoante a combinação de dados:

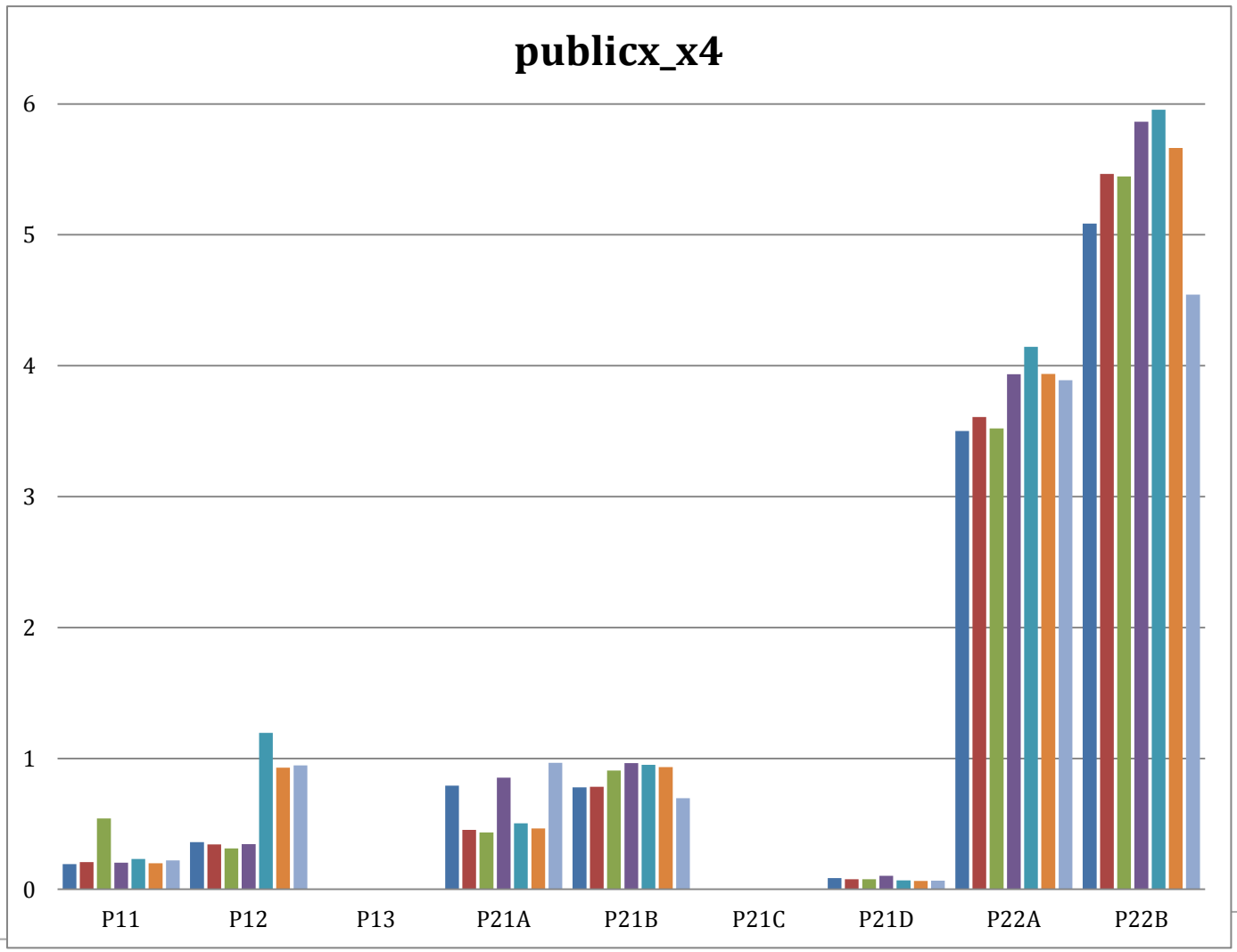


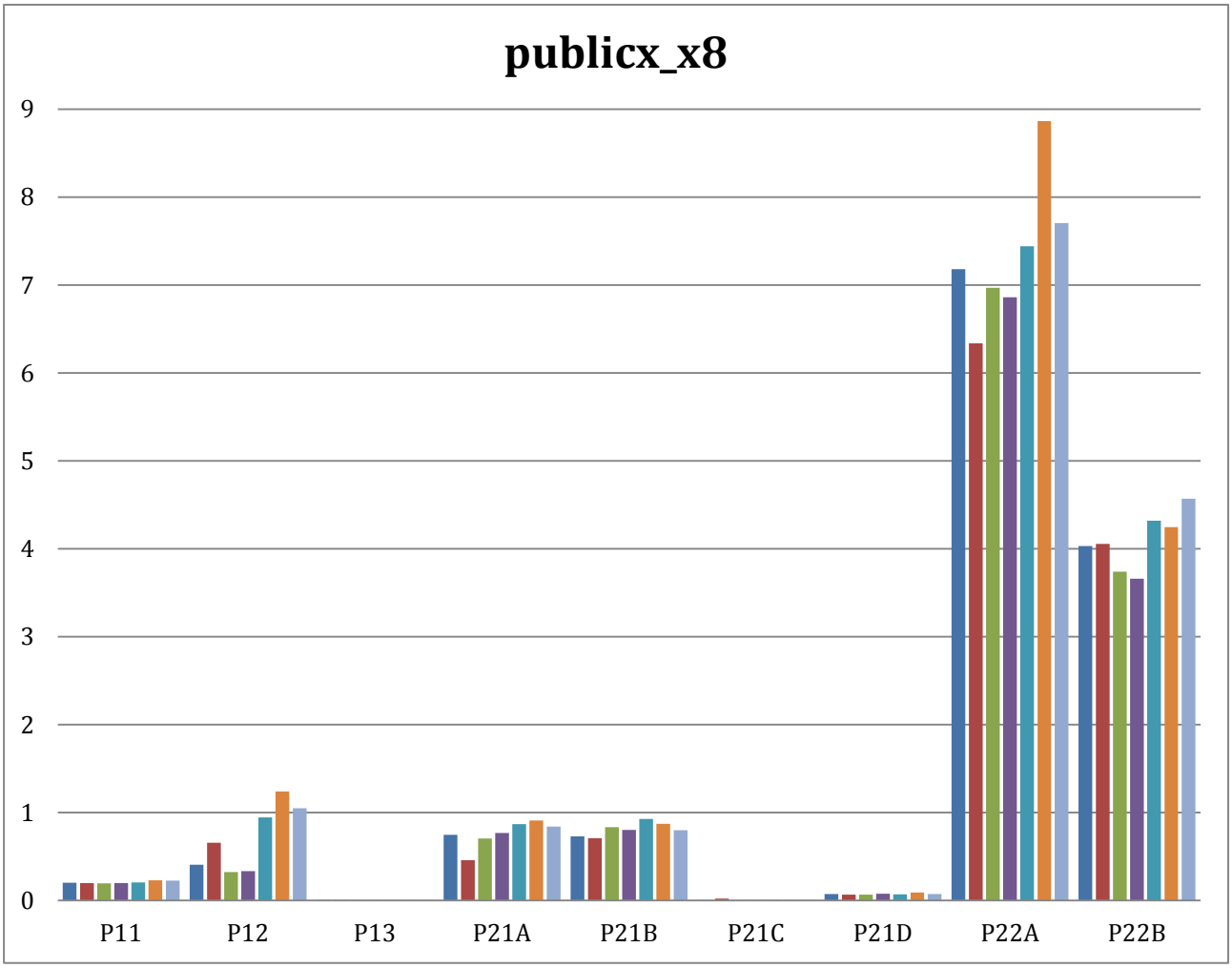




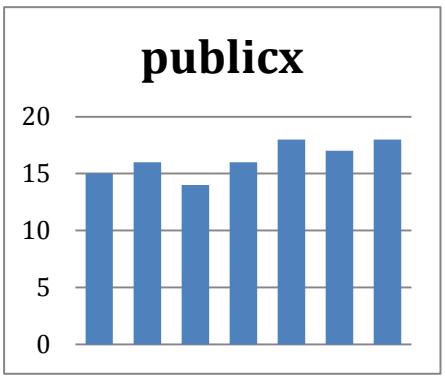
Gráficos dos Tempos Médios de cada Pergunta consoante o ficheiro:

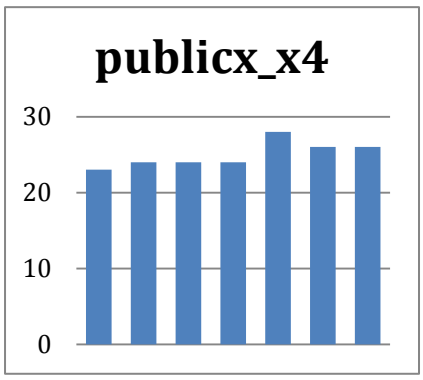


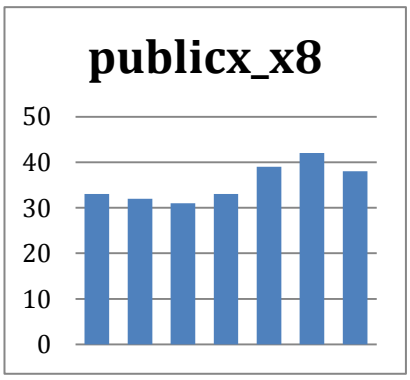




Tempos totais médios de execução:







Face a estes resultados optámos por escolher a TreeMap LinkedHashSet ArrayList como a combinação a usar. Tudo isto ficou mais fácil graças à implementação das Factories que facilitaram muito a execução dos testes. Testes estes que foram realizados numa máquina com 8Gb RAM, P6100 2GHz num Windows 7 64bit no IDE *NetBeans*, testes foram realizados em máquinas diferentes e os resultados foram todos semelhantes.

Conclusão

Como é possivel observar a forma como desenhamos e desenvolvemos a estrutura do programa permitiu-nos responder às perguntas de uma forma mais fácil ou dificil, o programa ficou de certo modo mais virado para responder às perguntas todas e um pouco mais lento na inserção na estrutura. Mas preferimos dispensar mais tempo para a inserção mas tornar as perguntas muito mais rapidas.

Assim definimos o projecto tal como foi apresentado e achamos que está num nível muito bom na óptica do utilizador pois foi desenvolvido de forma a ficar direcionado para este.