UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC0604 – Programação Orientada a Objetos

Prof. Robson L. F. Cordeiro

Lista 7 - Generics-Collections

1) Escreva uma classe genérica que recebe um tipo, necessariamente, possuindo as funcionalidades da classe java.lang.Number. A classe terá como atributos dois valores que definirão um par, e possuirá os seguintes métodos:
- getMax
- getMin
- toString: retorna uma string com a concatenação dos dois valores

2)
a) Escreva o método de ordenação insertion-sort* de maneira genérica. Use o seguinte algoritmo em pseudo-código:

```
Insertion-sort(TIPO[] A)        /*Não use ArrayList, apenas array simples*/
  Laço j← 1 até (A.tamanho-1)
    temp ← A[ j ]
    i ← j-1
    Enquanto i > -1 E A[i] > temp
      A[i+1] ← A[i]
      i ← i-1
    Fim Enquanto
    A[i+1] ← temp
  Fim Laço
Fim
```

➔ Modifique o pseudo algoritmo para que ele use o método compareTo ao invés do operador >. Teste seu algoritmo com uma classe boxing, como Integer ou Float.
b) Escreva uma classe que implementa a interface Comparable implementando o método compareTo.
c) Escreva um método principal que usa a classe do item b) com o algoritmo do item a). Execute o método sobre um array com 10 dados, imprimindo os dados antes e depois da ordenação.

*O insertion-sort é um algoritmo de ordenação por comparação que tem péssimo desempenho esperado de $O(n^2)$; ele só deve ser usado em razão de sua simplicidade, e em operações de ordenação com poucos elementos.

3) Escreva uma classe que defina uma árvore binária de busca genérica, a qual recebe um parâmetro T que implementa a interface Comparable. Sua árvore deverá ter métodos de inserção, remoção, busca e impressão ordenada de seus valores. Não é necessário mantê-la balanceada.

➔ Para os próximos exercícios, modifique a quantidade de dados (para menos ou para mais) sempre que julgar necessário, ou viável.

4) Usando a classe Math.Random para criar valores aleatórios do tipo Double.
a) Insira 10^8 dados em um ArrayList; em seguida, acesse 10^7 destes dados usando o método get(i) – use um índice i aleatório. Meça o tempo gasto usando:

```
long tempoInicial = System.currentTimeMillis();
...
long tempoFinal = System.currentTimeMillis();
System.out.println( tempoFinal - tempoInicial ); /*converta para minutos!*/
```

b) Agora delete 10^7 elementos cujos índices devem ser escolhidos aleatoriamente, e insira novos 10^7 valores aleatórios. Meça o tempo.
c) Repita o exercício dos itens a) e b) usando um LinkedList.
d) Compare os tempos em uma tabela (um arquivo texto ou Excel), e discuta os resultados de acordo com o que foi visto em aula.
➔ Enquanto estiver realizando o experimento, não realize outras operações que possam comprometer o processamento, e influenciar no tempo.

Para entrega: código dos projetos NetBeans referentes aos exercícios acima em um arquivo zip ➔ entregar via Tidia→Atividades

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

SCC0604 – Programação Orientada a Objetos

Prof. Robson L. F. Cordeiro

5) Usando a classe Math.Random para criar valores aleatórios do tipo Double.
a) Insira 10^8 dados em um ArrayList; dos dados inseridos, guarde referências a 10^7 destes elementos (escolhidos aleatoriamente) em um segundo array auxiliar (não um ArrayList).
b) Em seguida, execute o método `boolean contains(Object o)` passando como parâmetro os 10^7 elementos guardados no item a). Meça o tempo.
c) Agora insira os elementos do item a) em um HashSet e repita os itens a) e b) usando o mesmo array auxiliar.
d) Compare os tempos em uma tabela (um arquivo texto ou Excel), e discuta os resultados de acordo com o que foi visto em aula.
➔ Enquanto estiver realizando o experimento, não realize outras operações que possam comprometer o processamento, e influenciar no tempo.

6) a) Crie uma HashMap cujas chaves são inteiros escolhidos aleatoriamente (int inteiro = random.nextInt()), e cujos valores são dados por:

```
Math.hypot(inteiro*2/3, Math.sqrt(inteiro*2/3));
```

Insira 10^7 pares e guarde 10^6 chaves em um array auxiliar.
b) Use o array auxiliar do item a) para executar o método `V get(Object key)` 10^6 vezes. Meça o tempo. Isto é, ao invés de fazer o cálculo, você recuperar o resultado a partir de um mapa.
c) Agora repita o cálculo do item a) para os 10^6 valores guardados no array auxiliar. Meça o tempo.
➔ Caso os tempos colhidos sejam muito pequenos (abaixo de uma dezena de segundos), aumente a escala para 10^8, 10^9, e assim por diante.
d) Compare os tempos em uma tabela (um arquivo texto ou Excel), e discuta os resultados de acordo com o que foi visto em aula. Agora responda: se você tem memória, mas não tem tempo, o que você pode fazer?
e) Refaça os itens a) e b) usando o TreeMap. Compare os tempos em uma tabela (um arquivo texto ou Excel), e discuta os resultados de acordo com o que foi visto em aula.
➔ Enquanto estiver realizando o experimento, não realize outras operações que possam comprometer o processamento, e influenciar no tempo.

7. Implemente uma classe chamada ArraySet utilizando um ArrayList. Compare o desempenho desta classe com qualquer uma das implementações Set da API Java.
➔ Enquanto estiver realizando o experimento, não realize outras operações que possam comprometer o processamento, e influenciar no tempo.