# Conjuntos de inteiros



# Conjuntos de inteiros

- Os conjuntos de inteiros têm diversas aplicações em muitos domínios
  - Números do totoloto
  - Números de telefone
  - O ...
- Pretendemos construir diversos tipos de conjuntos de inteiros, para depois escolher o mais adequado consoante as situações
  - Em particular, queremos
    - Um conjunto simples de inteiros
    - O Um conjunto de inteiros em que seja fácil saber qual o maior
    - O Um conjunto de inteiros ordenado

# O nosso programa deve permitir

- Criar um conjunto de inteiros
  - Simples, ou com a funcionalidade extra de encontrar rapidamente o máximo
- Acrescentar números inteiros a um conjunto
- O Remover um número inteiro do conjunto
- O Testar se um número pertence ao conjunto
- Testar a relação de subconjunto
- Listar os elementos de um conjunto

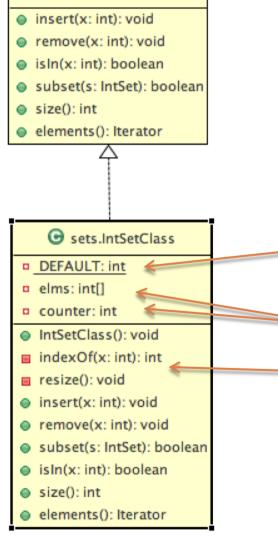
# Vamos definir uma família de conjuntos de inteiros

- IntSet fornece um conjunto adequado de métodos para conjuntos alteráveis, não limitados, de inteiros
  - o public void insert(int x)
    - O Acrescenta o inteiro x ao conjunto PRE: !isIn(x)
  - o public void remove(int x)
    - O Remove o inteiro x do conjunto PRE: isIn(x)
  - o public boolean isIn(int x)
    - O Se x pertence ao conjunto, retorna true, caso contrário, retorna false
  - o public boolean subset(IntSet s)
    - O Se this é sub-conjunto de s retorna true, caso contrário, retorna false
  - o public int size()
    - O Retorna o tamanho do conjunto
  - o public Iterator elements()
    - O Retorna um iterador de inteiros, para suportar as listagens

#### A interface IntSet

#### A classe IntSetClass

44



sets.IntSet

- Vamos definir uma classe que implemente a interface IntSet de modo a criar Conjuntos de inteiros simples
  - Esquema de implementação da interface a que já estamos habituados
    - Acrescentamos uma constante, com o tamanho por omissão
    - ODuas variáveis, com um vector acompanhado
    - ODois métodos auxiliares, privados, já nossos conhecidos...

#### Uma família de conjuntos de inteiros

sets.IntSet insert(x: int): void MaxIntSet é uma sub-classe de IntSetClass remove(x: int): void isln(x: int): boolean subset(s: IntSet): boolean O Por ser sub-classe de IntSetClass, também size(): int elements(): Iterator implementa a interface IntSet: G sets.IntSetClass Símbolo de herança DEFAULT: int elms: int[] Comportamento semelhante ao de IntSetClass, counter: int IntSetClass(): void mas com um método extra **max** que retorna o maior indexOf(x: int): int resize(): void elemento do conjunto insert(x: int): void remove(x: int): void Variável biggest acrescentada subset(s: IntSet): boolean isln(x: int): boolean Novo construtor size(): int elements(): Iteratø insert e remove redefinidos Método max acrescentado sets.MaxIntSetClas biggest: int MaxIntSetClass(): void insert(x: int): void

remove(x: int): void

max(): int

# Especificação de IntSetClass

```
public class IntSetClass implements IntSet {
                                                                                                        sets.IntSet
                                                                     Não são usados membros
                                                                                                     insert(x: int): void
                                                                                                     remove(x: int): void
                                                                     protegidos, neste exemplo.
                                                                                                     isln(x: int): boolean
                                                                                                     subset(s: IntSet): boolean
                                                                     Isto significa que as sub-
                                                                                                     a size(): int
                                                                                                     elements(): Iterator
                                                                     classes de IntSetClass apenas
                                                                     lhe podem aceder através
                                                                                                       G sets.IntSetClass
   public IntSetClass() { }
                                                                                                     DEFAULT: int
                                                                     da sua interface pública.
                                                                                                     elms: int[]
                                                                                                     a counter: int
                                                                     O nível de acesso é
                                                                                                     IntSetClass(): void
   public void insert(int x) {...}
                                                                                                     indexOf(x: int): int
                                                                     aceitável, porque o iterador
                                                                                                     resize(): void
   public void remove(int x) {...}
                                                                                                     insert(x: int): void
                                                                     permite visitar todos os
                                                                                                     remove(x: int): void
                                                                                                     subset(s: IntSet): boolean
   public boolean isIn(int x) {...}
                                                                     elementos da colecção.
                                                                                                     isln(x: int): boolean
                                                                                                     size(): int
   public boolean subset(IntSet s) {...}
                                                                                                     elements(): Iterator
   public int size() {...}
   public Iterator elements() {...}
                                                                                                      G sets.MaxIntSetClass
                                                                                                      biggest: int
                                                                                                      MaxIntSetClass(): void
                                                                                                      insert(x: int): void
```

remove(x: int): voidmax(): int

sets.IntSet

insert(x: int): voidremove(x: int): void

isln(x: int): booleansubset(s: IntSet): boolean

elements(): Iterator

DEFAULT: int

IntSetClass(): void

indexOf(x: int): int

insert(x: int): void

remove(x: int): voidsubset(s: IntSet): boolean

isln(x: int): boolean

elements(): Iterator

G sets.MaxIntSetClass
biggest: int

MaxIntSetClass(): void
 insert(x: int): void
 remove(x: int): void
 max(): int

resize(): void

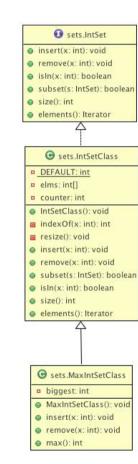
size(): int

elms: int[]
counter: int

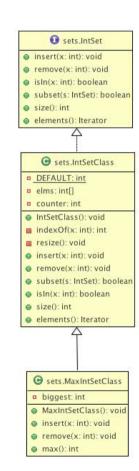
o size(): int

```
public class IntSetClass implements IntSet {
  private static final int DEFAULT = 10;
  private int[] elms;
  private int counter;
                                         Esta constante, as duas
                                         variáveis e os método
  public IntSetClass() { . . . }
                                         indexOf e resize
  private int indexOf(int x) {...}
                                         são privados.
  private void resize() {...}
                                         São inacessíveis fora
  public void insert(int x) {...}
                                         desta classe, mesmo
  public void remove(int x) {...}
                                         para as sub-classes.
  public boolean isIn(int x) {...}
  public boolean subset(IntSet s) {...}
  public int size() {...}
  public Iterator elements() {...}
```

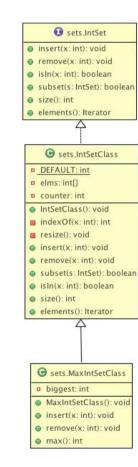
```
public class IntSetClass implements IntSet {
  private static final int DEFAULT = 10;
  private int[] elms;
  private int counter;
  public IntSetClass() {
    elms = new int[DEFAULT];
    counter = 0;
  private int indexOf(int x) {
    int i = 0;
    while (i < counter) {</pre>
      if (elms[i]==x)
        return i:
      i++;
    return -1;
```



```
public void insert(int x) {
  if (counter == elms.length)
     resize();
  elms[counter++] = x;
private void resize() {
  int[] tmp = new int[elms.length*2];
  for (int i = 0; i < counter; i++)
      tmp[i] = elms[i];
  elms = tmp;
public void remove(int x) {
  int index = indexOf(x);
  counter--;
  elms[index] = elms[counter];
```



```
public boolean subset(IntSet s) {
  if (s.size() < this.size()) return false;</pre>
  for (int i = 0; i < counter; i++)
    if (!s.isIn(elms[i]))
      return false;
  return true;
public boolean isIn(int x) {
  return (indexOf(x) !=-1);
public int size() {
  return counter;
public Iterator elements() {
  return new IteratorClass(elms, counter);
```

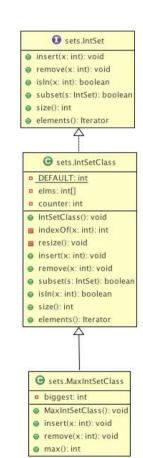


# Especificação de MaxIntSetClass

```
public class MaxIntSetClass extends IntSetClass {
   private int biggest;
   public MaxIntSet() {...}
   public void insert(int x) {...}
   public void remove(int x) {...}
   public int max() {...}
}
```

#### Repare que:

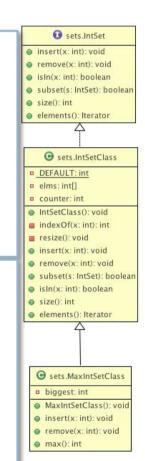
- A constante DEFAULT, por ser de classe, não é herdada
- O As duas variáveis de instância da super-classe, elms e counter, são herdadas mas não podem ser acedidas directamente por serem privadas
- O construtor da super-classe não é herdado
- O método indexOf da super-classe não é herdado, por ser privado
- Temos uma nova variável biggest
- Temos um novo construtor MaxIntSet e um novo método max
- O Temos dois métodos redefinidos insert e remove
- O Esta classe também implementa a interface IntSet



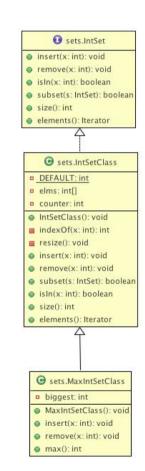
```
public class MaxIntSetClass extends IntSetClass {
  private int biggest;
  public MaxIntSetClass()
    super();
    biggest = 0;
  public void insert(int x)
    if (size() == 0 \mid \mid x > biggest)
      biggest = x;
    super.insert(x);
```

Como em todos os construtores devemos inicializar todas as variáveis de instância. Para garantir que as variáveis inacessíveis da super-classe também são inicializadas, usa-se a chamada ao construtor da super-classe!

O insert começa por tratar do caso especial levantado pela necessidade de actualizar a variável biggest. No resto, seria igual ao da super-classe, portanto, delega nela a implementação.



```
public void remove(int x) {
  super.remove(x);
  if ((size()>0) \&\& (x == biggest)) {
    Iterator it = elements();
    biggest = it.next();
    int tmp;
    while (it.hasNext()) {
      tmp = it.next();
      if (tmp > biggest)
        biggest = tmp;
public int max() { return biggest; }
```



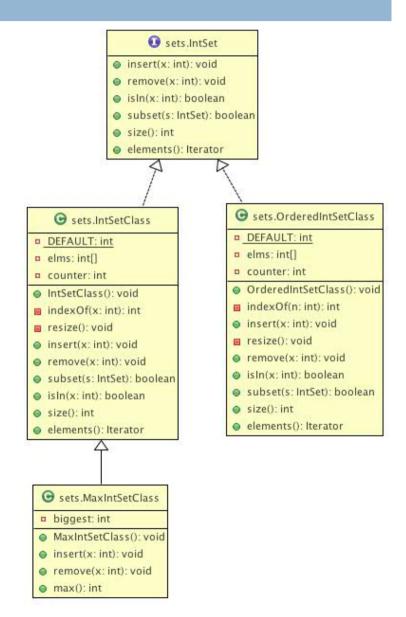
```
sets.IntSet
public class OrderedIntSetClass
                                                                                    insert(x: int): void
                                      implements IntSet {
                                                                                    remove(x: int): void
                                                                                    isln(x: int): boolean
   private static final int DEFAULT = 10;
                                                                                    subset(s: IntSet): boolean
   private int[] elms;
                                                                                    size(): int
                                                                                    elements(): Iterator
   private int counter;
   / * *
                                                                                                    G sets.OrderedIntSetClass
                                                                            G sets IntSetClass
     * Inicializa o vector acompanhado,
                                                                                                   DEFAULT: int
                                                                          DEFAULT: int
                                                                                                    p elms: int[]
                                                                          □ elms: int[]
     * de modo a representar
                                                                                                   o counter: int
                                                                          a counter; int
     * um conjunto vazio de inteiros.
                                                                                                   OrderedIntSetClass(): void
                                                                          IntSetClass(): void
                                                                                                   indexOf(n: int): int
                                                                          indexOf(x: int): int
                                                                                                   insert(x: int): void
                                                                          resize(); void
                                                                                                   resize(): void
   public OrderedIntSetClass() {
                                                                          insert(x: int): void
                                                                                                   remove(x: int): void
                                                                          remove(x: int): void
       elms = new int[DEFAULT];
                                                                                                   isln(x: int): boolean
                                                                          subset(s: IntSet): boolean
                                                                                                   subset(s: IntSet): boolean
                                                                          isln(x: int): boolean
       counter = 0;
                                                                                                   size(): int
                                                                          o size(): int
                                                                                                   elements(): Iterator
                                                                          elements(): Iterator
                                                                           biggest: int
```

MaxIntSetClass(): void
 insert(x: int): void
 remove(x: int): void

max(): int

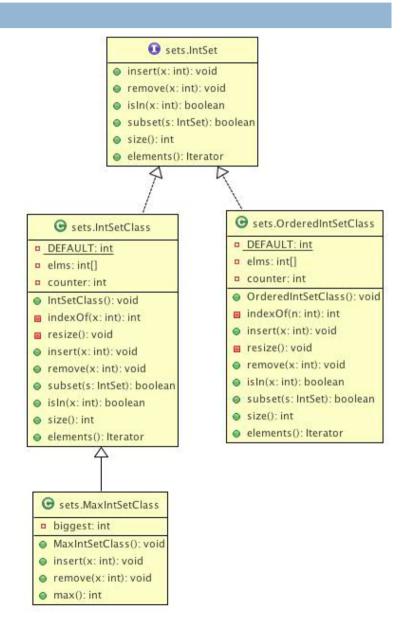
```
sets.IntSet
/ * *
                                                                                insert(x: int): void
 * Devolve o índice do elemento <code>n</code>.
                                                                                remove(x: int): void
                                                                                isln(x: int): boolean
 * @param n - o elemento a pesquisar no conjunto.
                                                                                subset(s: IntSet): boolean
 * @return - o índice com a posição do elemento,
                                                                                size(): int
                                                                                elements(): Iterator
 * ou o índice do primeiro inteiro maior se não
 * existir
                                                                                               Sets.OrderedIntSetClass
                                                                         G sets IntSetClass
private int indexOf(int n) {
                                                                                               DEFAULT: int
                                                                      DEFAULT: int
                                                                                               p elms: int[]
                                                                      a elms: int[]
   int low = 0;
                                                                                               counter: int
                                                                      a counter; int
   int high = counter-1;
                                                                                               OrderedIntSetClass(): void
                                                                      IntSetClass(): void
                                                                                               indexOf(n: int): int
                                                                      indexOf(x: int): int
   int mid = -1;
                                                                                               insert(x: int): void
                                                                      resize(): void
                                                                                               resize(): void
   while (low <= high) {</pre>
                                                                      insert(x: int): void
                                                                                               remove(x: int): void
                                                                      remove(x: int): void
      mid = (low+high)/2;
                                                                                               isln(x: int): boolean
                                                                      subset(s: IntSet): boolean
                                                                                               subset(s: IntSet): boolean
      if (elms[mid] == n) return mid;
                                                                      isln(x: int): boolean
                                                                                               size(): int
                                                                      o size(): int
      else if (n < elms[mid]) high = mid-1;</pre>
                                                                                               elements(): Iterator
                                                                      elements(): Iterator
      else low = mid+1;
                                                                       return low;
                                                                       biggest: int
                                                                       MaxIntSetClass(): void
                                                                       insert(x: int): void
                                                                       remove(x: int): void
                                                                       max(): int
```

```
public void insert(int x) {
  int pos = indexOf(x);
  if (counter == elms.length)
    resize();
  for (int i = counter; i > pos; i--)
    elms[i] = elms[i-1];
  elms[pos] = x;
  counter++;
private void resize() {
  int[] tmp = new int[elms.length*2];
  for (int i = 0; i < counter; i++)
      tmp[i] = elms[i];
  elms = tmp;
```



```
public void remove(int x) {
  int i = indexOf(x);
  while (i < counter-1) {
    elms[i] = elms[i+1];
    i++;
  }
  counter--;
}

public boolean isIn(int x) {
  int i = indexOf(x);
  if (counter == i)
    return false;
  return elms[i] == x;
}</pre>
```



```
public boolean subset(IntSet s) {
                                                                                            sets.IntSet
                                                                                       insert(x: int): void
   if (s.size() < this.size())</pre>
                                                                                       remove(x: int): void
       return false;
                                                                                       isln(x: int): boolean
                                                                                       subset(s: IntSet): boolean
   for (int i = 0; i < counter; i++)
                                                                                       size(): int
       if (!s.isIn(elms[i]))
                                                                                       elements(): Iterator
          return false:
   return true;
                                                                                                       Sets.OrderedIntSetClass
                                                                               G sets IntSetClass
                                                                                                       DEFAULT: int
                                                                             DEFAULT: int
                                                                                                       elms: int[]
                                                                             a elms: int[]
                                                                                                       counter: int
                                                                             a counter; int
public int size() {
                                                                                                       OrderedIntSetClass(): void
                                                                            IntSetClass(): void
                                                                                                       indexOf(n: int): int
   return counter;
                                                                             indexOf(x: int): int
                                                                                                       insert(x: int): void
                                                                             resize(): void
                                                                                                       resize(): void
                                                                             insert(x: int): void
                                                                                                       remove(x: int): void
                                                                             remove(x: int): void
                                                                                                       isln(x: int): boolean
                                                                             subset(s: IntSet): boolean
public Iterator elements() {
                                                                                                       subset(s: IntSet): boolean
                                                                             isln(x: int): boolean
                                                                                                       size(): int
                                                                            o size(): int
   return new IteratorClass(elms, counter);
                                                                                                       elements(): Iterator
                                                                             elements(): Iterator
                                                                              biggest: int
                                                                             MaxIntSetClass(): void
                                                                             insert(x: int): void
                                                                             remove(x: int): void
```

max(): int