# La interfaz Comparable

La interfaz comparable define el método **compareTo** para comparar objetos.

* Supongamos que buscamos diseñar un método genérico para encontrar el más grande de dos objetos del mismo tipo, como dos estudiantes, dos fechas, dos cirulos, dos rectángulos o dos cuadrados. Con el fin de lograr esto, los dos objetos deben ser comparables, entonces el comportamiento en común para estos dos objetos debe ser *comparable*. Java provee la interfaz **Comparable** para este propósito. La interfaz está definida de la siguiente manera:

|  |
| --- |
| package java.lang;  import java.util.\*;  public interface **Comparable**<T> {  public int **compareTo**(T **o**);  } |

* El método **compareTo** determina el orden de este objeto con el objeto especifico **o** y retorna:

|  |  |
| --- | --- |
| *Un entero negativo* | *Si el objeto es menor que* ***o*** |
| *Cero* | *Si el objeto es igual a* ***o*** |
| *Un entero positivo* | *Si el objeto es mayor que* ***o*** |

* La interfaz comparable es una interfaz genérica. El tipo genérico **E** es remplazado por un tipo concreto cuando se implementa esta interfaz. Muchas clases en la librería de java implementan **Comparable** para definir un orden para los objetos. Las clases Byte, Short, Integer, Long, Float, Double, Character, BigInteger, BigDecimal, Calendar, String, y Date implementan la interfaz **Comparable**. Por ejemplo, las clases Integer, BigInteger, String y Date son definidas en la API de java como se muestra a continuación:

|  |
| --- |
| **public class** Integer **extends** Number **implements** Comparable<Integer> {  // class body omitted  @Override  **public int** compareTo(Integer o) {  // Implementation omitted  }  } |
| **public class** BigInteger **extends** Number **implements** Comparable<BigInteger> {  // class body omitted  @Override  **public int** compareTo(BigInteger o) {  // Implementation omitted  }  } |
| **public class** String **extends** Object **implements** Comparable<String> {  // class body omitted  @Override  **public int** compareTo(String o) {  // Implementation omitted  }  } |
| **public class** Date **extends** Object **implements** Comparable<Date> {  // class body omitted  @Override  **public int** compareTo(Date o) {  // Implementation omitted  }  } |

* Entonces, los números (Integer y BigInteger) son comparables, las cadenas (String) son comparables y lo son las fechas (String) también. Podemos usar el método **compareTo** para comparar dos números, dos cadenas o dos fechas. Por ejemplo, el siguiente código:

|  |
| --- |
| import java.util.Date;  public class **ComparaNumerosCadenasFechas** {  public static void ***main***(String[] args) {  System.*out*.println(new Integer(3).compareTo(new Integer(5)));  System.*out*.println("ABC".compareTo("ABE"));  Date fecha1 = new Date(2015, 1, 1);  Date fecha2 = new Date(2014, 1, 1);  System.*out*.println(fecha1.compareTo(fecha2));  }  }  **Salida:**  **-1**  **-2**  **1** |

* La primer línea muestra **-1** ya que 3 es menor que 5. La segunda línea imprime un valor negativo (**-2**) ya que **ABC** es menor que **ABE**. La última línea muestra un numero positivo (**1**) porque la **fecha1** es mayor que la **fecha2**.
* Suponiendo que **n** es un objeto Integer, **s** un objeto String y **d** un objeto Date. Todas las siguientes expresiones son verdaderas, true:

|  |  |  |
| --- | --- | --- |
| n **instanceof** Integer  n **instanceof** Object  n **instanceof** Comparable | n **instanceof** Integer  n **instanceof** Object  n **instanceof** Comparable | s **instanceof** String  s **instanceof** Object  s **instanceof** Comparable |

* Ya que todos los objetos **Comparable** tienen el método **compareTo**, el método **java.util.Arrays.sort**(**Object[]**) en la API de java usa el método **compareTo** para comparar y ordenar objetos de un array a condición de que los objetos sean instancias de la interfaz **Comparable**. El siguiente código ofrece un ejemplo de ordenar un array de objetos String y un array de objetos BigInteger.

|  |
| --- |
| import java.math.BigInteger;  import java.util.Arrays;  public class **OrdenarCadenasNumeros** {  public static void ***main***(String[] args) {  String[] ciudades = {"Mexico", "Guadalajara","Acapulco", "Monterrey"};    Arrays.*sort*(ciudades);  for (String ciudad : ciudades) {  System.*out*.println(ciudad);  }    BigInteger[] numeros = {  new BigInteger("2323231092923992"),  new BigInteger("432232323239292"),  new BigInteger("54623239292")  };    Arrays.*sort*(numeros);  for (BigInteger numero : numeros) {  System.*out*.println(numero);  }  }  } |

* El programa crea un array de String e invoca al método **sort** para ordenar las ciudades. El programa crea un array de BigInteger e invoca al método **sort** para ordenar los números.
* No podemos usar el método **sort** para ordenar un array de objetos **Rectangulo**, porque esta clase no implementa la interfaz **Comparable**. Sin embargo, podemos definir una nueva clase que implemente la interfaz **Comparable**. Las instancias de esta nueva clase serán comparables. Nombraremos **RectanguloComparable** a esta nueva clase, el código se muestra a continuación:

|  |
| --- |
| public class **RectanguloComparable** extends Rectangulo implements Comparable<RectanguloComparable> {  public **RectanguloComparable**(double ancho, double alto) {  super(ancho, alto);  }    @Override  public int **compareTo**(RectanguloComparable o){  if (getArea() > o.getArea()){  return 1;  } else if (getArea() < o.getArea()){  return -1;  } else {  return 0;  }  }    @Override  public String **toString**(){  String info = "";  info += "Ancho: " + getAncho();  info += "**,** Alto: " + getAlto();  info += "**,** Area: " + getArea();  return info;  }  } |

* La clase **RectanguloComparable** extiende a la clase **Rectangulo** e implementa la interfaz **Comparable**. La palabra implements indica que **RectanguloComparable** hereda todas las constantes de la interfaz **Comparable** e implementa el método en la interfaz. El método **compareTo** compara el área de dos Rectángulos. Una instancia de **RectanguloComparable** también es una instancia de **Rectangulo**, **ObjetoGeometrico**, **Object** y **Comparable**:

|  |
| --- |
|  |

* Ahora podemos usar el método **sort** para ordenar un array de objetos **RectanguloComparable** como se muestra a continuación:

|  |
| --- |
| import java.util.Arrays;  public class **RectangulosOrdenar** {  public static void ***main***(String[] args) {    RectanguloComparable[] rectangulos = {  new RectanguloComparable(3.4, 5.4),  new RectanguloComparable(13.24, 55.4),  new RectanguloComparable(7.4, 35.4),  new RectanguloComparable(1.4, 25.4)  };    Arrays.*sort*(rectangulos);    for (RectanguloComparable rectangulo : rectangulos) {  System.*out*.println(rectangulo);  }    }  }  Salida:  Ancho: 3.4, Alto: 5.4, Area: 18.36  Ancho: 1.4, Alto: 25.4, Area: 35.559999999999995  Ancho: 7.4, Alto: 35.4, Area: 261.96  Ancho: 13.24, Alto: 55.4, Area: 733.496 |